Аннотация

Данное учебное пособие предлагается для студентов механико-математического факультета всех специальностей, поскольку оно основано на материале научно-популярных текстов. Его можно использовать при обучении студентов бакалавриата. Тексты небольшие и по содержанию не превышают требований к учебному пособию данного уровня, легкие и по сложности доступные как для начинающих, так и для продолжающих изучать язык, как под руководством преподавателя, так и самостоятельно.

Основная цель пособия – развитие умения читать и понимать научно-техническую литературу для извлечения необходимой информации в пределах изучаемой специальности. Пособие состоит из четырёх частей.

Первая часть содержит тексты из научно-популярной литературы, лексико-грамматические упражнения. Тексты адаптированы.

Вторая часть данного учебного пособия содержит тексты для технического перевода. Преподавтель может варьировать объём текста для перевода каждому студенту в соотсветствии с его уровнем знания языка.

Третья часть пособия состоит из грамматических пояснений. Она также содержит дополнительный материал, который можно периодически брать на уроках в случае, если студенты забыли что-либо из того, что в этой части сoдержится.

Четвёртая часть данного пособия содержит англо-русско-казах- ский словарь-минимум с транскрипцией, что облегчает проблему работы как в русских, так и в казахских группах.

Contents

**Part I**

Lesson I The Purpose of a Computer ………………………………. 2

Lesson II The Raw Material ………………………………………… 4

Lesson III The Product …………………………………………………6

Lesson IV The Device; Distinguishing features ………………………..8

Lesson V Instructions ………………………………………………….10

Lesson VI Hardware ……………………………………………………13

Lesson VII The Power and Speed of a Computer ……………………….15

Lesson VIII Types of Computer ………………………………………….18

Lesson IX Analog Computers …………………………………………..19

Lesson X Prossessing – Programmes …………………………………..21

Lesson XI Computer's Processor ………………………………………..23

Lesson XII Software ……………………………………………………..25

Lesson XIII Languages ……………………………………………………27

Lesson XIV People ………………………………………………………..28

Lesson XV The Future ……………………………………………………30

**Part II**

Supplementary Reading – 32

**Part III**

Часто встречающиеся грамматические термины – 99

Grammar Explanations – 102

Irregular verbs – 129

Additional Materials – 132

**Part IV**

Vocabulary – 138

**PART I**

**INTRODUCTION TO COMPUTERS**

**Lesson I**

**The Purpose of a Computer**

A computer is a tool. A tool operates by taking raw material and converting it into a product by means of a device, which performs a process. The process is determined by people. To take an analogy from everyday life: a blunt pencil (raw material) is converted into a sharp pencil (product) by means of a penknife performing the process of sharpening, as determined by a schoolboy. The device – the penknife – can of course be turned to several other uses, such as extracting stones from horses’ hooves, or carving initials on a desk. But its scope is basically limited to one type of process and one type of material. In all essentials, computers are tools, and in describing them we can consider each of the functions of any tool: the raw material, the product, the device, the process and the people.

Ex. 1 Read and memorise international words:

computer *n*; operate *v*; material *n*; convert *v*; product *n*; process *n, v*; analogy *n*; extract *n, v*; initial *n, a*; basic *n, a*; limit *v,n*; type *n, v*; function *n,v*.

Ex. 2 Read the words and learn their meanings:

**vocabulary**

tool *n* – инструмент(арий); метод; аппарат raw *a* – сырой, необработанный

raw material – необработанный материал by means (of) *prep* – посредством any *pron* – любой, всякий, какой-либо (чего-либо); с помощью (ч-л) на

a device *n* – устройство, прибор, механизм; русский не переводится) by *prep*  элемент;метод, схема; выражает отношения

to perform *v*  – to do, to make, твор. падежа (кем? чем?

to fulfill, to carry out – выполнять, исполнять, to determine *v* – to define,

blunt (pencil) *a* – тупой, затупившийся to specify *v* – производить,определять, a penknife *n* – перочиный нож устанавливать решать,

sharp (pencil) (product) *a* – острый; резкий, extracting stones – добыча камня

чёткий; символ # to turn to *v* – обращаться к

to sharpen *v* – заострять, чинить essential *a* – основной

to turn (to) *v* – обратиться (к ч-л), заняться (ч-л) to convert *v* – превращать

other *pron* – другой, иной horses’ hooves *n* – копыта лошадей to operate *v* – работать, оперировать to perform *v* – выполнять

carving *n* – гравирование, вырезание узора по a scope *n* – область (сфера)

дереву,металлу действия(видимости); контекст; horse *n* – лошадь масштаб; hooves *pl* (hoof *sg*) *n* – (копыта)

to carve *v* – вырезать, гравировать размах; область применения

essentials *n* – основы to describe *v* – описывать

to consider *v* – рассматривать, полагать, считать each (of) *pron* – каждый (из)

Ex. 3 Learn the word combinations:

1. **raw material** – необработанная информация 2.**convert smth. into** – преобразовывать (превращать) что-либо в 3. **by means of** –посредством, с помощью 4. **of course** – конечно 5. **such as** – такой (такие, такая, такое) как 6. **to be limited to** – быть ограниченным чем-либо

Ex. 4 Answer the questions on the content of the text:

1. What is a computer? 2. How does a tool operate? 3. What is a blunt pencil? 4. What is it converted into? 5. What is the penknife limited to? 6. Is a penknife a device? 7. What can we consider describing computers?

Ex. 5 Read and translate the text:

**Types of Computers**

There are three main categories of computers, namely, digital computers, analog computers and hybrid computers.

Digital computers are those used for commercial applications, and also by scientists and engineers engaged in the processing of large numbers of mathematical calculations.

Analog computers are such computers which can accept data as a quantity varying over a length of time, rather than as a series of distinct items, each with a unique value. Analog computers are used for scientific purposes, to measure the results of experiments, or simulate processes which can be described in terms of quantities varying according to known rules over lengths of time. They are also used for process control of industrial operations.

Hybrid computers are another type of machines combining some of the properties of both digital and analog computers.

Digital processing is usually faster and more sophisticated than the processing which can be done by analog machines, so hybrid processing gives the advantages of both types of devices.

Ex. 6 Read and translate the sentence and put as many questions as possible to its parts:

**I** **like to read interesting books at home.** (9 questions)

1. Who likes … ? (Special question to the **subject**, Специальный вопрос к **подлежащему, бастауышқа** арнайы сұрақ)
2. Do I like … ? (**General** question, **общий** вопрос, **жалпы** сұрақ, )
3. What do I like to do? (Question to the **predicative**, вопрос к **предикативу** (именной части составного именного сказуемого), **баяндауыщқа** қойылатын сұрақ)
4. Do I like to read or … ? (**Alternative** question, **альтернативный** вопрос,

**балама** сұрақ)

1. What do I like to read … ? (Special question to **the** **direct object**, специальный вопрос к **прямому дополнению**, **тура толықтауышқа** қойылатын арнайы сұрақ)
2. What books do I like … (Special question to the **attribute**, специальный вопрос к **определению**, қойылатын арнайы сұрақ **анықтауышқа**)
3. I like to read interesting books, …? (**Disjunctive** question or **tag**-question, **tail**-question, **разделительный** вопрос (с хвостиком), **анықтаушы** сұрақ)
4. Don’t I like … ? (**Negative general** question, **вопросительно-отрицательное** предложение, **сұраулы болымсыз жалпы** сұрақ)
5. Where do I like to read books? (Special question to the **adverbial modifier**, специальный вопрос к **обстоятельству**, арнайы сұрақ **мекен** **пысықтауышқа** қойылатын)

Ex. 7 Put as many questions as possible to the parts of the sentence:

**We study at the al-Farabi National University**. (8 questions)

**Lesson II**

**The Raw Material**

First, the raw material: facts (or data). A fact is a thing or event known to exist or have happened: something, which can be described in precise, measurable terms. A fact may be an amount of £13 as written on a cheque; a line on an architect's plan; a measurement in a scientific experiment; a membership number; an address; an item on an invoice; a forecast in a plan. An individual fact, on its own, does nothing more than provides the single piece of information (which) it represents.

Ex. 1 Read and memorise international words:

**fact** *n* – ; **term** *n* – , **cheque** *n* – ; **line** *n* – ; **architect** *n* – ; **experiment** *n* – ; **address** *n* – ; **individual** *a* – ; **information** *n* – ; **plan** *n* – ; **number** *n* – ;

Ex. 2 Read the words and learn their meanings:

**vocabulary**

data *n, pl,* (datum *n, sl*) – данные event *n* – событие to exist *v* – существовать, жить to happen *v* – случаться to describe *v* – описывать precise *a* – точный measurable *a* – измеримый amount *n* – количество measurement *n* – измерение scientific *a* – научный membership *n* – членство item *n* – элемент, блок invoice *n* – расчёт, вычисление to forecast *v* – предсказывать to provide *v* – обеспечивать single *a* – один (-очный) piece *n* – кусок, часть to represent *v* – представлять

Ex. 3 Learn the word combinations:

1. raw material – необработанный материал;

2. on its own – сам по себе; в отдельности;

3. nothing more than – ничего более … чем (как);

4. a scientific experiment – научный эксперимент

5. measurable terms – измеримые сроки, периоды, элементы, составляющие

6. a piece of information – одна инормация

Ex. 4 Answer the questions on the content of the text:

1. What is the raw material?

2. What is a fact?

3. What can be described in precise, measurable terms? 4. What may a fact be?

5. What can be written on an architect’s plan?

6. Does an individual fact provide а piece of information? 7. What represents information?

Ex. 5 Match the beginnings of the sentences with their endings (*see text* “Types of

Сomputers” on page 2):

1. Hybride computers …
2. ‍Digital processing …
3. ‍Analog computers …
4. ‍There are three main categories of computers …
5. ‍Digital computers namely digital computers …

**a**. … analog computers and hybride computers.

**b**. … are those used for commercial applications, and also by scientists and engineers.

**c**. …engaged in the processing of large numbers of mathematical calculations.

**d**. … are such computers which can accept data as a quantity varying over a length of time, rather than as a series of distinct items, each with a unique value.

**e**. … are another type of machines combining some of the properties of both digital and analog computers.

**f**. … is usually faster and more sophisticated than the processing which can be done by analog machines.

Ex. 6 Study the way questions are made up:

**I am a student at the al-Farabi Kazakh National University.**

1. Who is a student …?
2. What educational Institution do you study at?
3. Are you a student at …?
4. Do you study at …?
5. Are you a student or …?
6. You are a student at the al-Farabi Kazakh National University, aren’t …?
7. Aren’t you a student…?

Ex. 7 Read and translate the sentence and put as many questions as possible about its content:

**He is the fifth kid in the family of his parents.** (7 questions)

**Lesson III**

**The Product**

The product that a computer generates from this raw material is information. By the relation of facts of the same type or different types, something useful is obtained. In isolation, a cheque for $13 is of little interest to the bank manager: related to an overdraft of $88 and an agreed maximum of $100 for Mr. Smith, the isolated fact becomes informative, with unfortunate consequences for Mr. Smith. The single line on the architect's drawing is meaningless: related to all the other lines on the drawing, it plays part as representing an outside wall, or a room divider or a drain. The single experimental measurement may be insignificant: related to hundreds of other measurements, the fact may become crucial in the proof of a scientific theory. The process carried out by the computer is the converting of isolated facts into information by relating them to each other. This is a process with which we are all familiar in daily life - figuring out how much has been spent on a shopping spree by jotting down facts, isolated purchases, and adding them up to form a grand total. Looking up an address in a street atlas is, again, the conversion of an isolated fact, an address, into information: the relationship of the address to a locality, and thus to a way of getting there. We are also familiar with tools used to process facts: the cash register, the adding machine, the library catalogue, the slide rule, even a dictionary, all are used to process facts into information.

Ex. 1 Read and memorise the international words:

**generate** *v* – вырабатывать, производить **information** *n* – **relation** *n* – отношение **isolation** *n* – **interest** *n* – **bank** *n* – хранилище **manager** *n* – управляющая программа **maximum** *n* – **drain** *n,v* – утечка, течь **crucial** *a* – решающий **theory** *n* – **figure** *n* – цифра **form** *v* – **total** *a* – общий **atlas** *n* – **locality** *n* – **register** *n,v* – , заносить в регистр **machine** *n* – **catalogue** *n* –

Ex. 2 Read the words and learn their meanings:

**vocabulary**

relation *n* – отношение (the) same *pron* – тот же самый different *a* – различный useful *a* – полезный to obtain *v* – получать overdraft *n* – превышение своего to slide *v* – скользить кредита в банке dictionary *n* – словарь (книга) to agree *v* – соглашаться to become *v* – становиться unfortunate *a* – неудачный consequence *n* – следствие drawing *n* – рисование meaningless *a* – бессмысленный to play *v* – играть part *n* – часть represent *v* – представлять outside *a* – внешний, вне wall *n* – стена room *n* – комната, место divider *n* – драйвер, устройство cash *n* – наличные деньги привода

to drain *v* – осушать, просачиваться insignificant *a* – незначительный hundred *num, n* – сто, сотня crucial *a* – критический proof *n* – доказательство to carry *v* – нести familiar *a* – знакомый, известный daily *a* – ежедневный shopping – делать покупки spree *n* – веселье, резвость to jot *v* – кратко записать, набросать purchase *v* – покупать

to add *v* – добавлять, скалдывать to look *v* – смотреть

street *n* – улица conversation *n* – разговор locality *n* – место нахождения thus – и так, таким образом way *n* – путь, способ to get *v* – получать, добираться, even *interjc* – даже доставать

Ex. 3 Learn the word combinations:

**in isolation** – изолированно **related to** – связанный с **to play a part** – играть роль **to carry out** – выполнять **each other** – друг друга **to be familiar with** – быть знакомым **to figure out** – выражаться (в цифрах; **at**) **how much (many)** – сколько **to jot down** – набросать, кратко записать **to add up** – складывать (числа) **a grand total** – общий итог **to look up smth. in** – смотреть (искать) что-либо в (словаре)

Ex. 4 Learn to translate the noun chains:

**the cash register** – регистр оперативной памяти **the adding machine** – суммирующая машина **the library catalogue** – библиотечный каталог **the slide rule** – логарифмическая линейка **a street atlas** – схема улиц города **the bank manager** – банковский менеджер, управляющий банком **a room divider** – межкомнатная перегородка **a car driver** – водитель автомобиля (легкового)

Ex. 5 а) Study the irregular verbs:

to be – am, is , are – was, were – been – being; to get – got – got – getting

to become – became – become – becoming; to spend – spent – spent – spending;

b) Study the modal verbs:

can – could; may – might

Ex. 6 Translate the sentences:

1. You already know the grammar and the words. So you can translate the text. 2. You have already written the test, so you can go. 3. You could do it. You had so much time to do it! 4. You could have done it much better if you hadn’t been in a hurry. 5. Oh, she couldn’t have said such a foolish thing!

1. You may go home now. 2. They may be a bit late! 3. He might come soon.

Ex. 7 Answer the questions on the content of the text “**The Product**”:

1. Does a computer generate a product? 2. What is a product that a computer generates? 3. What is of little interest to the bank manager?

4. Why is a single line on an architect’s drawing meaningless? 5. May a single experimental measurement be insignificant? 6. When may the fact (the previous question) become crucial? 7. How does the computer convert isolated facts into information? 8. Who are familiar with the process of converting the isolated facts?

Ex. 8 Put the questions to the words in bold type:

1. **A computer** is really a very specific kind of accounting machine.
2. It can do **arithmetic problems** faster than any person alive.
3. By means of **electric circuits** it can find the answer to a very difficult and complicated problem in a few seconds.
4. A computer can **“remember”** information you give it.
5. It stores the information **in its “memory”** until it is needed.
6. It works the problem with **lightning** speed.
7. Then it checks its work **to make sure** there are no mistakes.

**Lesson IV**

**The Device: Distinguishing Features**

How does the computer compare with other information-processing tools? The three main distinguishing features of a computer are speed, capacity and versatility. An example of speed: an average person might take about a minute to add up ten 7- digit numbers. In the same time, some computers could have added up 1000 million or more numbers. Further, the average man would feel some unease about the accuracy of his addition of 1000 million numbers: the computer sum would be correct. An example of storage capacity: the contents of a hundred volumes of the London telephone directory could be accommodated in one typical storage unit: in one second, 15,000 entries could be looked up. Examples of versatility: the same computer could be used to print bank statements; draw perspective drawings from outline sketches; calculate betting odds; calculate satellite orbits; match aspiring lovers; calculate population statistics. There are few fields of human activity in which the computer cannot be applied.

Ex. 1 Read and memorize the international words:

**person** – лицо, человек **minute** – **million** – **accuracy** – точность **correct** – правильный **telephone** – **directory** – справочник **typical** – обычный **second** – второй **print** – печатать **perspective** – **sketch** – набросок **calculate** – вычислять **satellite** – спутник **orbit** – **match** – согласовывать **population** – население **statistics** – **human** – человек **activity** – деятельность **application** – применение

Ex. 2 Read the words and learn their meanings:

**vocabulary**

distinguishing *a* – отличающийся feature *n* – характеристика to compare *v* – сравнивать main *a*  – главный speed *n* – скорость capacity *n* – возможность versatility n – многосторонность average *a* – средний

digit *n* – цифра further *adv* – далее to feel *v* – чувствовать unease *n* – неудобство sum *n* – сумма storage *n* – память content *n* – содержание volume *n* – объём to accommodate *v* – размещать unit *n* – блок entry *n* – запись to print *v* – печатать statement *n* – оператор to draw *v* – чертить, рисовать (карандашом) outline *n* – контур, эскиз to bet *v* – держать пари, биться об заклад odd *a* – нечётный to match *v* – согласовывать to aspire *v* – стремиться, домогаться few *adv* – мало field *n* – поле, область

Ex. 3 Learn the word combinations:

**information processing tools** – инструментальные программные средства обработки инфоормации **main distinguishing features** – основные отличительные характеристики **in the same time** – в то же самое время **storage capacity** – объём (вместимость) памяти **London telephone directory** – справочник (директория) лондонской телефонной системы **one typical storage unit** – одна обычная единица памяти **bank statements** – операторы банка данных **outline sketches** – общие очертания, набросок **population statistics** – статистика (переписи) населения (популяции) **human activity** – человеческая деятельность

Ex. 4 a) Study the irregular verbs; give the rest forms of the verbs:

do –, take –, have –, feel –, be –, draw –

b) Let us recall modal verbs:

Ex. 5 Answer the questions on the content of the text:

1. What are three main distinguishing features of a computer? 2. Can you give an example of speed? 3. How many million numbers could some computers have added up? 4. What would the average man feel while adding 1000 million numbers? 5. Give an example of storage capacity, please. 6. How many entries could be looked up in one second? 7. What can one say about versatility? 8. Can the computer be applied in all fields of human activity? Why?

Ex. 6 Complete the sentences of the text with a suitable word: (***computer, general-purpose,***

***kinds, specific, special-purpose, different, the most difficult, purpose***)

**What Is a Computer?**

There are different … of computers. Some do only one job over and over again. These are … computers. Each specific application requires a … computer. One kind of … can help us build a space – craft. A special-purpose computer is built for this … alone and cannot do

anything else. But there are some computers that can do many … jobs. They are called … computers. These are the "big" brains that solve … problems of science. They answer questions about rockets and planes, bridges and ships long before these things are even built.

**Lesson V**

**Instructions**

The computer achieves such prodigious feats not because of any superhuman powers of intelligence, but because it is able to carry out a few extremely simple operations accurately and very quickly. We are all familiar with the principle of doing something complex by performing a set of simple instructions: a knitting pattern tells one how to make a garment by following a set of simple stitches; a do-it-yourself construction kit works in the same way. Just as the knitting pattern instructions consist of a few basic steps written in a special code (K 1, P I mean knit one stitch, purl one stitch) so the computer has a repertoire of basic operations which it can carry out on data. These operations can be loosely grouped into control operations which enable the computer to operate various devices which feed data into it for processing, and which accept results after processing; arithmetic operations, enabling the computer to add and subtract, and

thus multiply and divide as well; and logical operations, which enable the computer to select different sets of instructions as a result of tests made on items of data.

Ex. 1 Read and memorise the international words:

operation *n*, familiar *a*, principle *n*, instruction *n*, data *n*, special *a*, repertoire *n*, group *n*, control *v,n* processing *n*, result *n*, arithmetic *a*, logical *a*, select *v*, complex *a*, construction *n*, test *n,v* basic *a*, operate *v*, code*n*,

Ex. 2 Read the words and learn their meanings:

**vocabulary**

prodigious *a* – oгромный, изумительный; knitting *n* – вязание, вязка;

pattern *n* – узор, образец; одеяние; garment *n* – предмет одежды, покров,

stitch *n, v* – стежок, шов, петля; kit *n* – набор инструментов; шить; вышивать; purl *n* – журчать, журчание;

loosely *adv* – свободно; to feed *v* – подавать, кормить (fed, fed);

item *n* – элемент, часть.

Ex. 3 Learn the word combinations:

**a do-it-yourself** – самостоятельное выполнение; («Cделай сам») **a construction kit** – конструкторский набор (жесткий) **in the same way** – тем же самым способом **just as** – именно так, как **a few** – несколько **few** – немного **as well** – также, тоже (в конце отрицательного предложения) **as a result** – в результате

Ex. 4 Learn to translate the noun chains:

**super-human powers** – сверхчеловеческие силы  **the knitting pattern instructions** – указания по вязанию узора **control operation** – операция управления **arithmetic operations** – арифметические операции

**a set of simple stitches** – набор простых стежков

Ex. 5 a) Study the irregular verbs:

to be –, to do –, to tell –, to make –, to write –, to have –, can –, to feed –

b) Give some sentences with the modal verbs:

can –; may –; must – ;

c) Now, give their Past Indefinite (Simple) forms, if any:‍

Ex. 6 Answer the questions on the content of the text “**Instructions”**:

1. Why does the computer achieve such prodigious feats? 2. What are we all familiar with? 3. How many basic steps do the knitting pattern instructions consist of? 4. What enable the computer to operate various devices? 5. What do arithmetic operations do? 6. What do logic operations enable the computer to do?

Ex.7 Read the text.

**The Second Industrial Revolution**

The first industrial revolution involved the replacement of human and animal muscle power by the power of machines. It released mankind from a lot of physical work. The application of computers to industry is causing a second industrial revolution. Computers have been described as machines that think. That is an oversimplification. Computers are machines that are capable of very rapid and accurate calculations, but they need instructions from human beings.

However within the past few years great advances have been made in the techniques of programming computer. The latter acts in remarkably intelligent ways. Moreover, having received instruction the computer is able to analyse them in a very short period of time. The results can be used to control some apparatus or plant directly. In either case the computation is done much faster than a human could do it.

Ex. 8 Match the beginnings of the sentences to their endings:

1. The first industrial revolution involved …
2. The machines released mankind …
3. The application of computers to industry …
4. Computers have been described …
5. Computers are machines that are capable of …
6. Having received instructions …
7. The results can be used …

а.… from a lot of physical work.

b.… to control some apparatus or plants directly.

c.… is causing a second industrial revolution.

d.… the replacement of human and animal muscles power by the power of machines.

e.… the computer is able to anolise them (instructions) in a very short period of time.

f.… very rapid and accurate calculations, but they need instructions from people.

g.… as machines that think.

Ex. 9 Choose the correct words to complete the questions to the parts of the sentence and

remember their structures:

**They have very many interesting books in their collection at home.**

1. Who (**have, has, had, having**) very many interesting books in (**their, his, her, theirs**) collection at home?
2. (**have, has, had, having**) they very many interesting books in (**them, their, his, hers, theirs**) collection at home?
3. (**does, do, did, doing**) they have very many interesting books in (**theirs, their, them, hers**) collection at home?
4. (**did, doing, do, done**) they have very (**much, many, a lot of, little**) interesting books in their collection at home or in the library?
5. Do they (**has, have, having, had**) very many or very (**a few, few, a little**) interesting books in their collection at home?
6. They have very many interesting books in their collection at home, (**don’t, haven’t, hadn’t, didn’t**) they?

**Lesson VI**

**Hardware**

Hardware is the term given to all the electronic and mechanical gadgetry which together forms a computer system, as distinct from the sets of instructions which are used to operate them. A computer system consists of three basic types of machine. First, there is that part of the system, which can perform operations as a result of instructions. This part is known as the central processor. Second, the unit used to store the raw material on which the processor is to

set to work: this is known simply as a storage device, although the term memory is also used, as an analogy in human terms.

Finally, there are devices (a) to place data into storage and (b) to extract the finished product from the system. These are known collectively as input devices and output devices. A computer is thus a collective noun used for a group of devices: a central processor, storage, input and output devices. In fact, the picture is not quite as clear-cut as this. Central processors contain a special type of storage device known as high-speed or immediate access store.

Ex. 1 Read the words and learn their meanings:

**vocabulary**

hardware *n* – мат.-тех. обеспечение gadgetry *n* – приспособление to distinct (from) *v* – отличаться (от) set *n* – ряд, набор, множество to consist (of) *v* – состоять из) to store *v* – хранить, запоминать storage *n* – хранилище, память although – хотя, несмотря на human *a* – человеческий to extract *v* – извлекать, выделять collectively *adv* –всё это вкупе, вместе input *n* – ввод output *n* – вывод thus *adv* – таким образом, так, и так quite – *adv* вполне, совсем, почти clear-cut *a* – чётко выполненный to contain *v* – содержать immediate *a* – немедленный, быстрый access *n* – доступ store *n* – память

Ex. 2 Read and learn the international words:

electronic – , mechanical – , finish – , collectively – , group –, central –, processor – ,

system – , special – , type – , machine – , to form – , term – , group – , storage – .

Ex. 3 Read the words and learn their meanings:

**gadgetry** – приспособление (в механизме), полезная мелочь **distinct (from)** – отличный (от) **to consist of** – состоять из **to perform** – выполнять **central – processor** – центральный процессор **unit** – блок **storage device** – устройство хранения данных, память **memory** – память, хранилище инфоромации **data** – информация, данные

Ex. 4 Learn the word combinations:

**basic types** – основные, базовые типы **as a result of** – в результате (ч.-л.) **to place data into** – поместить, записать данные в **to set to work** – приступить к работе **finished product** – законченный, конечный продукт **a collective noun** – собирательное существительное **in fact** – на самом деле, в самом деле, фактически **immediate access store** – память с быстрым доступом **collective noun** – собирательное существительное

Ex. 5 Learn to translate noun chains:

**storage** **device** – записывающее устройство, память **in** **human terms** – применительно к людям (на человеческом языке) **input** **devices** – устойства ввода **output** **devices** – устройства вывода **software** – программное обеспечение (программы) **peripheral** **unit** – периферийное (внешнее) устройство **hardware** – материально-техническое обеспечение компьютера

Ex. 6.

a) Study the irregular verbs, give the three forms of them:

to be –, to give –, to know –, to set –, to cut –.

b) Study the modal verbs’ equivalents:

**can** – to be **able** to (do)

**must** – to **be** to (do), to **have** to (do)

**may** – to be **allowed** to (do) = to be **permitted** to (do)

c) What other modal verbs do you know?

Ex. 7. Learn the nouns borrowed from other languages:

dat**um** *sngl* – dat**a** *pl* advice: (**a** (**one**) **piece** of advice, **3** **pieces** of advice)

information: (**a** **piece** of information, **pieces** of information)

Ex. 8. Do all of these nouns have a plural form? What are they?

g**oo**se – g**ee**se

f**oo**t – f**ee**t

m**ous**e – m**ic**e

fish – no plural form

fruit – no plural form

money – no plural form

Ex. 9. Answer the questions on the content of the text “**Hardware”**:

1. What term is given to all the electronic and mechanical gadgetry? 2. What forms a computer system? 3. What are distinct from the sets of instructions? 4. How many types of machine does a computer system consist of? 5. What can the central processor perform? 6. What is a storage device used for? 7. What device is used to place data into storage? 8. What is used to extract the finished product from the system? 9. What is immediate access store?

Ex. 10. Put the questions to the words in bold type:

1. The results can be used to control some apparatus or plants **directly.**
2. The **computation** is done much faster than a human could do it.
3. Automatic computing opened **the way** to new and much more effective methods of engineering design and management.
4. The computer may never make **possible** entirely new way of conducting business.
5. However, science and engineering are **the first to experience** the impact of
6. We live in a century **with the intimate independence of science and engineering**.

**Lesson VII**

**The Power and Speed of a Computer**

The power and speed of a computer depends on the amount of this storage available, since all instructions are held in this storage device, as well as that part of the data which is being immediately affected by the instructions. As high-speed store is expensive (even after the astonishingly sharp declines in price ever since the introduction of integrated circuits on silicon), large quantities of data are held on backing stores which are cheaper, but from which it takes longer to extract individual items of data.

Basic data still has to be placed into the backing store, so a common sequence of events is for this raw material to be put onto a backing store in one operation, and then be processed from the backing store before becoming the final product. Indeed, several stages of processing may take place, the intermediate results of each being transcribed to backing store, before the final product is available.

The central processor is usually placed in one box or cabinet, and input, output and storage devices each in their own boxes; for this reason, the latter being all collectively known as peripheral units. Nowadays a computer system is much like a child's construction kit in that the user can buy the “Mark 1” version and then improve his kit by buying new peripheral units and plugging them into his system. Or he can keep his peripheral units and change the central processor for a bigger and faster one. 'Bigger' will mean a larger immediate access store with the ability to hold larger and more complex sets of instructions; 'Faster’ will mean that the instructions themselves will be carried out more quickly.

Ex. 1 Read and memorise international words:

data *n* –, affect *n* –, instruction *n* –, introduction *n* –, integrate *v* –, silicon *n* –, extract *n* –, individal *adv* –, basic *adv* –, material *n* –, operation *n* –, final *n*, *adv* –, process *n* –, product *n* –, stage *n* –, result *n* –, transcribe *v* –, central *v* –, cabinet *n* –, reason *n* –,

collective *adv* –, peripheral *n, adv* –, construction *n* –, version *n* –, complex *n* –.

Ex. 2 Read the words and learn their meanings:

**vocabulary**

power *n* – мощность speed *n* – скорость

to depend (on) *v*  – зависить (от) amount *n* – количество

available *a*– имеющийся since *conj* – так как, поскольку, с (c тех пор как)

to carried out *v* – выполнять to mean *v* – значить, означать

to hold (held) *v* – держать as well as – так же как и

to affect *v* – влиять (на) as *adv, conj* – как, поскольку, так как

expensive *a*– дорогой even *a*– даже, чётный

astonishingly *adv* – удивительно sharp *a*– резкий, острый

to decline *v* – отклоняться price *n* – цена

ever since – с того времени integrated *a*– интегрированный

circuit *n* – (электро-) схема, цепь quantity *n* – количество

cheap *a*– дешёвый backing stores – вспомогательная,

ability *n* – способность внешняя память

individual *a*– отдельный item *n* – элемент, часть

basic *a*– базовый, основной still *adv* – всё ещё, всё-таки, всё же

so *adv* – и так, поэтому common *a* – обычный, простой, общий

sequence *n* – последовательность event *n* – событие. случай

then – затем, потом, тогда to process *v ,n* – обрабатывать, процесс

before – до, ранее, прежде чем indeed *adv* – в самом деле,

fast *a* – быстрый действительно

several – несколько each *pron* – каждый

to transcribe *v* – преобразовывать usually *adv* – обычно

box or cabinet *n* – коробка, отделение own *a* – собственный

reason *n* – причина the latter *n* – второй (последний)

peripheral *n* – периферийное устр-во unit *n* – блок, единица

nowadays *adv* – сейчас, сегодня like *a,adv*, *v* – похожий на, подобно,

quickly *adv* – быстро нравиться

kit *n* – набор, комплект to buy *v* – покупать

to improve *v* – улчшать plugging *n* – включение (электричество)

to keep *v* – держать, хранить to change *v* – (из-) менять

Ex. 3 a) Read the words and learn their meanings, paying attention to the degrees of

comparison of adjectives:

**short** – short**er** – (the) short**est** **strong** – strong**er** – (the) strong**est** **thin** – thin**ner** – (the) thin**nest** **beautiful** – **more** beautiful – (the) **most** beautiful **good** – better – (the) best

b) Now, find all the forms of the adjectives used in the text.

c) Give the rest of the forms of the following adjectives:

bad –; little –; far –; good –.

Ex. 4 Learn the word combinations:

**as well as** – также как и **high-speed store** – быстродействующая память **integrated circuits** – интегрированные схемы **backing store** – дополнительный накопитель **items of data** – единицы (элементы) данных **a common sequence** – стандартная последовательность **to take place** – происходить, иметь место **intermediate results** – промежуточные результаты **for this reason** – по этой причине (поэтому) **peripheral units** – периферийные блоки (устройства) **а child’s construction kit** – детский конструкторский набор **the “Mark 1” version** – версия Марк 1 **by buying** – купив, покупая, при покурке **to carry out** – выполнять

Ex. 5 a) Recall the modal verb’s equivalents:

to **be** to –; to **have** to– ;

b) What other irregular verbs are there in the text? Find them in the text and give all their forms.

c) Review the modal verbs:

d) What modal verbs and their equivalents are there in the text? Find them in the text.

Ex. 6 Answer the questions on the content of the text “**The Power and Speed of a Computer**”

(page 15):

1. What do the speed and power of a computer depend on? 2. Why do they depend on the amount of this storage available? 3. Why are large quantities of data held on backing stores? 4. What does it take longer to extract individual items of data from? 5. When do basic data have to be processed from the backing store? 6. Where does basic data have to be placed?

**The Main Types of Programming Languages**

The programming languages fall into two categories: procedure-oriented languages or high-level languages and machinery-oriented languages or low-level ones. High-level languages are: ALGOL, BASIC, COBOL, FORTRAN and PL/1. Low-level languages are: an assembler or a compiler and autocodes.

The basic languages where the program written by the programmer is similar to the machine code version are known as low level programming languages or autocodes. Here each instruction has a corresponding machine code equivalent. However, all such basic languages are closely connected to the machine code into which the source program can alternatively be converted. Programs written in a basic language can normally be used on a particular machine or a range of machines.

High level languages differ from low level languages in that the instructions in high level languages take the form of rather complex statements. A high level statement is generally translated into several machine code instructions.

**Lesson VIII**

**Types of Computers**

What we have described so far has been mostly concerned with computers used for commercial applications, and also by scientists and engineers engaged in the processing of large numbers of mathematical calculations. This sort of computer is known as a digital computer. It works on data held as a large number of distinct items, and operates in a series of steps or instructions. But 'raw material' or unprocessed data is not always in the form of individual and separate items – cheques, bills, bookings. In some cases this raw material is something which lasts over a length of time and changes its value over that time. For example, the speed of a car over a journey is in a sense an item of data, which lasts as long as the journey takes, and varies continuously over that journey. We may need to process this data item to obtain information, for example to detect a speed in excess of 70 m.p.h. and slow down. We are all familiar with a machine, which does just this: the car speedometer displays the speed at any time, and by watching this we can control our vehicle. ( to be continued)

Ex. 1 Read and memorise international words:

concern *n* –; computer *n* –; process *n* –; number *n* –; calculation *n* –; material *n* –; engineer *n* –; to separate *v* –; machine *n* –; control *n* –; commercial *a* –; to operate *v* –; familiar *a* –; display *n* –; individual *n* –; series *n* –; to speed *v* –;cheque *n* –; application *n* –; data *n* –;information *n* –; form *n* –; speedometer *n* –;step *n* – sort *n* –; instruction *n* –; mathematical *a* –;

Ex. 2 Read the words and learn their meanings:

**vocabulary**

scientist *n* –учёный calculation *n* – вычисление sort *n* – сорт, тип, вид to hold *v* – держать distinct *a* – отличный, чёткий bill *n* – счёт cheque *n* – чек a series *n* – серия, ряд book *n* – заказать, взять (билет to last *v* – продолжаться

length *n* – длина, продолжительность to change *v* – менять speed *n* – скорость value *n* – значение, величина journey *n* – путешествие, поездка to obtain *v* – получать to detect *v* – обнаруживать to excess *v* – превышать to display *v* – выводить на экран to watch *v* – наблюдать, vehicle *n* – средство

Ex. 3 Learn the word combinations:

**to be concerned with** – быть связанным с **to be engaged in** – быть занятым **distinct items** – отдельные элементы **so far** – до сих пор

**to be familiar with** – быть знакомым с **unprocessed data** – необработанные

данные

Ex. 4

a) Revise the irregular verbs:

to know –; to hold –; to take – ; to think – ; to draw – ;

b) What other irregular verbs are there in the text?

Ex. 5 Answer the questions on the content of the text “**Types of Computers**” (page18):

1. What are used for commercial application? 2. B**y** what scientists and engineers are computers used? 3. What is known as a digital computer? 4. What is held as a large number of distinct items? 5. What operates as a series of steps or instructions? 6. Are unprocessed data always in the form of individual and separate items? 7. What are these items? 8. Does this raw material last over a length of time? 9. How long does the speed of a car over a journey last? 10. How does the speed vary? 11. Why may we need to process this data item? 12. What may we need to obtain information for? 13. What machine are we all familiar with?

Ex. 6 Choose the correct alternative:

1. The programming languages fall into (**two, three**) categories: procedure-oriented languages or high-level languages and machinery-oreinted languages or low-level ones.
2. (**High-level, low-level**,) languages are: ALGOL, BASIC, COBOL, FORTRAN and PL/1.
3. (**Low-level, high-level**) languages are: assembler or compiler and autocades.
4. The basic languages, where the programme written by the programmer is similar to the machine code version are known as (**low-level, high-level**) programming languages or autocodes.
5. All such (**basic languages, procedure-oriented languages**) are closely connected to the machine code into which the source program can alternatively be converted.
6. Programs written in a basic language (**can, can’t**) normaly be used on a particular machine or a range of machines.
7. High-level languages differ from low-level languages because the instructions in high-level languages take the form of rather (**complex, simple**) statements.
8. A high level statement is generally translated into (**only one, several**) machine code instructions.
9. The program produced from a high-level language is rather (**easier, more cumbersome**) and (**longer, shorter**) than that produced from a low-level language.
10. A high-level language (**has, has no**) such close correspondence to instructions for a particular machine.

**Lesson IX**

(Give the title to the text after having read it)

Analog computers are a type of computer, which can accept data as a quantity varying over a length of time, rather than as a series of distinct items each with a unique value. Analog computers are used for scientific purposes, to measure the results of experiments, or to simulate processes, which can be described in terms of quantities varying according to known rules over lengths of time. They are also used for process control of industrial operations, constantly measuring some varying quantity, for example voltage or gas pressure, and causing operations

such as switching off circuits or opening valves when the quantity being measured reaches pre-determined values. Hybrid computers are yet another type of machine, combining some of the properties of both digital and analog computers. A varying input accepted by the analog computer can be converted into a series of distinct values which can then be processed by the digital part of the machine. Digital processing is usually faster and more sophisticated than the processing, which can be done by analog machines, so hybrid processing gives the advantages of both types of device.

Ex. 1 Read and memorise international words:

industrial *v* –; computer *n* –; unique *a* –; measure *n* –; series *n* –; result *n* –; type *n* –;

experiment *n* –; to simulate *v* –; process *n* –; control *n* –; constant *a* –; term *n* –; gas – *n;* operation *n* –; pressure *n* –; hybrid *n* –; machine *n* –; to convert *v* –; analog *n* –;

Ex. 2 Read the words and learn their meanings:

**vocabulary**

analog *a* – аналоговый to accept *v* – принимать

quantity *n* – величина, количество to vary *v* – изменяться

distinct *a* – чёткий, отдельный length *n* – длина

unique *a* – единственный to simulate *v* – моделировать

to describe *v* – описывать constantly *adv* – постоянно

Ex. 3 Learn the word combinations:

**in terms of** – в терминах, на языке **according to** – согласно (ч.-л.) **over lengths of time** – в промежутки времени **a varying quantity** – изменяющаяся величина **such as** – такой (такая, такое, такие) как **a** **quantity being measured** – измеряемая величина **pre-determined values** – заранее определённые величины **both … and …--** как …, так и … **both types** – оба типа **rather than** – скорее …, чем; а не

Ex. 4 a) Learn to translate noun chains:

process control – gas pressure –

information-producing equipment –

programme instructions –

cash registers –

vacuum tubes –

memory size –

reference number –

computer capacity –

table legs –

b) Give some other similar noun chains you can think of.

Ex. 5 a) Study irregular verbs:

b) Find all the irregular verbs in the text and give all their forms.

c) Recall the modal verbs:

d) Pick out all the modal verbs from the text.

Ex. 6 Answer the questions on the content of the text:

1. What can analog computers accept? 2. How does a quantity of data vary? 3. Has a quantity a unique value? 4. What are analogue computers used for? 5. What else are they used for? 6. What do hybrid computers combine? 7. Can a varying input be converted into a series of distinct values? 8. What is faster and more sophisticated?

Ex. 7 Read and translate the text:

**The Digital Computer**

The first electronic digital computer was the ENIAC (from Electronic Numerical Integrator and Computer) built at the University of Pennsylvania between 1943 and 1949. Early computers such as this were large enough physically to fill a room the size of a small house.

Yet, with all their size, their computational capability was rather primitive by present standards. The ENIAC computer contained over 18,000 vacuum tubes, and the overall reliability of this device was rather low owing to failures in these electronic tubes. Finding and replacing malfunctioning tubes took countless hours. The tubes also generated considerable heat and consumed large amounts of electric power. Even with its disadvantages, however, the ENIAC performed well enough to demonstrate the utility of the digital computer and to encourage later development of improved devices.

**Lesson X**

**Processing – Programs**

We have said that the computer is a tool, and the job it does is processing facts to obtain information. But the same could be said of adding machines, cash registers and a number of other accounting and calculating machines. The great difference between a computer and other information-producing equipment is versatility: unlike the other machines, which in general perform one fixed type of operation, the computer's operations can be varied without limit. How does this happen? We have already explained how the computer performs a complex task by a series of relatively simple steps, and how the heart of the machine is a central processor in which the instructions are stored and obeyed.

The storage unit of the processor can be likened to a large filing cabinet, with quantities of empty files or pockets each with a reference number. To start things off, one instruction is placed in each 'pocket". In the computer system this is done by means of an input device 'reading' the set of instructions or program. The computer is then directed to start working. The first instruction is extracted from the first pocket and whatever this instruction says is obeyed. Then the next instruction is extracted from the next pocket, and so on until the sequence of instructions is completed. Data which is to be processed is placed in pockets in the same way; one instruction might be to fill up a number of pockets with data; another might be to empty pockets containing results. ( *to be continued*)

Ex. 1 Read and memorise international words:

information *n;* register *n, v;* computer *n;* general *a;* complex *a;* to fix *v;* type *v,n;* instruction *n*; machine *n;* central *a;* cabinet *n;* extract *v,n;* fact *n;* start *v,n;* to contain *v;* result *v,n*; limit *a;*

Ex. 2 Read the words and learn their meanings:

**vocabuilary**

tool *n* – инструмент job *n* – задание to obtain *v* – получить (the) same *pron* – тот же самый versatility *n* –многосторонность unlike *prep* – в отличие от vary *v* – изменяться to happen *v* — случаться to explain *v* – объяснять relatively *adv* — относительно;

соответственно

heart *n* — сердце to obey *v* – подчиняться to liken *v* – сравнивать cabinet *n* – кабинет, ящик, бюро

empty *a* – пустой pocket *n* – карман to place *v* – помещать to start off *v* – запустить to extract *v* – извлечь whatever *a* , *pron* – когда бы ни; что бы ни to complete *v* – завершать to contain *v* – содержать‍

Ex. 3 Learn the word combinations:

adding machines, cache registers, accounting and calculating machines, information- producing equipment, unlike the other machines, a large filing cabinet, a reference number, to start things off, an input device, and so on, until the set of instructions is completed, data to be processed, in the same way, in general.

Ex. 4 Learn to translate **noun** **chains**:

**a storage unit** – устройство памяти; записывающее устройство **an input device** – устройство ввода **a computer system** – компьютерная система

Ex. 5 a) Write out all the irregular verbs contained in the text and give all their forms:

b) Pick out all the modal verbs which are used in the text:

Ex. 6 Answer the questions on the content of the text “**Processing – Programs**” (page 21):

1. What is the computer? 2. What is the job it does? 3. What else can do the same? 4. How can the computer’s operations be varied? 5. Where are the instructions stored? 6. How may things be started off?

7. And how is the same to be done in the computer system? 8. What is the first instruction extracted from? 9. When is the next instruction extracted from the next “pocket”? 10. How is the data which is to be processed placed in pockets?

Ex. 7 Put the questions to the words in **bold** type;

1. The **first electronic** computer was the ENIAC (from Electronic Numerical Integrator and Computer) built at the University of Pensilvania between 1943 and 1949.
2. Early computers such as this were large enouph physically **to fill** a room the size of a small house.
3. **Their** **computational capability** was rather primitive by present standards.
4. The ENIAC computer contained **over 18.000 vacuum tubes**, and the overall reliability of this device was rather low owing to failures in these electronic tubes.
5. **Finding and replacing** malfunctioning tubes took **countless hours**.
6. The tubes also generated considerable heat and consumed large amounts **of electric power**.
7. Even with its disadvantages, however, the ENIAC performed well enouph to demonstrate the utility of the digital computer and **to encourage** later development of improved devices.

**Lesson XI** (Give the title to the text)

In the computer's processor, pockets or storage locations hold characters, words or bytes; they are the smallest unit of information with which the machine can operate. If we were to go to any storage location and extract its contents, we could not tell whether the 'number' was an instruction, a data item or the address of another storage location, for these are all held in exactly the same way in store. Since instructions are thus the same sort of thing to the computer as numbers, they can be modified. This means that once a program of instructions is placed in store, it can be changed by its own action: a program of instructions can thus be made to react to each problem it has to face, and this enables extremely sophisticated and complex operations to be performed. However, there is one difficulty - and it is a very significant one.

The computer can only do what it is told: it slavishly obeys each instruction it extracts from successive storage locations, without any discrimination: it cannot say to itself 'that doesn't seem sensible', or 'I wonder what that means': it just obeys. So, instructions must be carefully worked out to make sure that in no circumstances will the computer be asked to do anything illogical. Therefore, when a computer issues a final demand for payment of a bill for Ј0, it is not the machine that has done something stupid: it is correctly obeying a stupid instruction. The illogicality has been caused by the human operator who has failed to foresee the situation and give the computer appropriate instructions. The art of programming is thus an exacting one, for every combination of circumstance must be considered by the programmer and taken care of by an appropriate set of instructions.

Ex. 1 Looking through the text read out all the international words from it.

Ex. 2 Read the words and learn their meanings:

**vocabulary**

storage *n* – память to hold *v* – удерживать to tell (differ) *v* – отличать another *pron* – другой, ещё один exactly *adv* – точно to modify *v* – изменять to mean *v* – значить once (if) *conj* – если to place *v* – помещать to face *v* – встречать; сталкиваться

to enable *v* – давть возможность extremely *adv* – чрезвычайно to sophisticate *v* – усложнять however *adv, conj* – как бы ни; тем неменее significant *a* – значительный slavishly *adv* – зависимо successive *a* – последовательный to seem *v* – казаться sensible *a* – чувствительный to wonder *v* – удивляться, xотеть знать circumstance *n* – обстоятельство carefully *adv* – внимательно illogicality *n* – алогичность issue *v,n* – выпускать; проблема demand *v,n* – требовать; спрос a bill *n* – счёт stupid *a* – глупый, тупой cause *v,n* – заставлять, вызывать; причина to fail *v* – не удаваться to foresee *v* – предусмотреть appropriate *a* – надлежащий exacting *a* – изнуряющий once *conj* – после того, как; если

Ex. 3 Learn the word combinations:

**the same way** – тот же способ, путь

**to work out** – вырабатывать

**to make sure that** – удостовериться в том, что

**in no circumstances** – ни при каких обстоятельствах **to take care of** – заботиться о

Ex. 4 a) Learn to translate **noun chains**:

**storage location** – адрес; ячейка памяти **data item** – элемент данных, информации **human operator** – оператор-человек

b) Give some noun chains of your own.

Ex. 5 a) Study all the irregular verbs in the text:

b) Write out them into your copybooks.

c) Give the equivalents of the modal verbs in the text:

Ex. 6 Answer the questions on the content of the text:

1. Where are pockets or storage locations located? 2. What are words or bytes? 3. What can the machine operate with? 4. Why can instructions be modified? 5. Can instructions be made to react to each problem it has to face?

6. What difficulty is there in this case?

7. When must instructions be carefully worked out?

8. Does a machine obey a stupid instruction?

9. Who causes illogicality?

10. What kind of art is a programming one?

Ex. 7 Put the questions to the words in **bold** type:

1. An **automatic** computer vbst be able to perform arithmetic operations.
2. An automatic **computer** must be designed to change signals in accordance eith certain arithmetic rules for interpreting the signals.
3. There is no need for each of the basic arithmetic processes **to be carried out** in a single operation: some operations can be built up from others.
4. **We** shall use the form “elementary operation” to denote an operation which is performed when the machine obeys a single instruction.
5. Compound **operations**, on the other hand, need **several instructions**.
6. It is inconvenient **for the programmar** **if his repertoir of instructions is too restricted**.
7. A compound operation takes **longer time** to execute than an elementary one.
8. A compromise must be reached between **the demands of the designer and the programmer**.
9. Both addition and subtraction are always treated as **elements operations**.

**Lesson XII**

**Software**

We have described the term 'hardware' already: it refers to all the machinery, electronic or mechanical, which together makes up a computer. Programs can be produced by individual computer users to perform specific jobs in which the user alone is interested. But many types of problem are more general, and programs written to solve one instance of a general problem may be used to perform the same job for others faced with similar problems; these programs are known as 'software'. In particular, manufacturers of computers spend large sums of money on developing this sort of software. A computer is obviously much more useful to a user if it comes with a number of ready-made programs which can be put to use at once: also the user is saved the time and expense involved in working out the programs for himself. This sort of software covers a great range of programs, from quite short and simple routines (for example programs to work out taxation for payroll users) to large and sophisticated programs (such as those for critical path method analysis used in scheduling complex production programs).

Ex. 1 Read and memorise the international words from the text:

machinery *n* – машины; оборудование; техника; механизмы; машинное оборудование; алгоритмы; механизмы обработки данных; individual *a, n* – отдельный; человек; specific *a,* – особый; general *a, n* – общий; генерал; sort *n* – вид; тип, сорт; routine *n* – программа; подпрограмма; taxation *n* – обложение налогом; взимание налога; размер, сумма налога; critical *a* – критический; важный; ответственный; дефицитный; metthod *n* – правило; способ; приём; complex *n, a* – комплекс; сложный; production *n* – произ- водство; продукция.

Ex. 2 Read the words and learn their meanings:

**vocabulary**

hardware *n* – мат.-техн. обеспечение machinery *n* – машиностроение job *n* – работа, задание alone *a* – один (-очный) instance *n* – пример to face *v* – встречать, сталкиваться similar *a* –подобный manufacturer *n* – производитель to develop *v* – разрабатывать obviously *adv* – очевидно to save *v* – спасать, экономить expense *n* – расходы

to involve v – включать, вовлекать to cover *v* – покрывать, охватывать quite *adv* – вполне routine *n* – подпрограмма taxation *n* – налогообложение payroll *n* – платёжная ведомость sophisticated *a* – усложнённый to chedule *v* – составлять расписание

Ex. 3 Learn the word combinations:

to make up – составлять in particular – в частности, особенно a ready-made programme – готовая программа at once – сразу (же) a great range of – большой диапазон to put to use – запустить

Ex. 4 a) Study the degrees of comparison of adjectives:

**large** – larger – (the) largest

**interesting** – more interesting – (the) *most* interesting

**big** – big*g*er – (the) big*g*est

**good** – better – (the) best

b) Now, pick out all the adjectives from the text and translate them.

Ex. 5 Answer the questions on the content of the text “**Software**” (page 23):

1. What term have we already described?

2. What makes up a computer?

3. Whom can programmes be produced by? 4. Are many types of problem more general?

5. When is a computer much more useful to a user?

6. How many programmes does this sort of software cover?

Ex. 6 Complete the sentences with the suitable words:

(**arithmetic, compound, interpreting, operations, automatic, “elementary operations”, the design, instruction**)

An **…** computer must be able to perform **…** operations; that is, it must be designed to change signals in accordance with certain arithmetic rules for … the signals.

There is, however, no need for each of the basic arithmetic processes to be carried out in a single operation: some **…** can be built up from others.

We shall use the term **…** to denote an operation which is performed when the machine obeys a single **…**.

**…** operations, on the other hand, need several instructions. The larger the number of different elementary operations, the more complicated **…** of the machine and the greater the amount of equipment needed.

**Lesson XIII**

**Languages**

Some of the most important pieces of software provided by manufacturers are computer languages. Instructions in the form operated on by the internal circuitry of a processor consist of a numeric code. Instructions operate on items of data stored with the program in the processor, and each item has its own 'pocket' or storage location, again identified by a number. Programs stored in the processor are thus a combination of numbers making up instructions and storage locations.

To work out the numerical equivalents for all the instructions in a program is a tedious and difficult task, requiring great attention to detail and the obeying of exacting rules. But computers themselves are very good at tedious and repetitive tasks, obeying exacting rules. So programming can be simplified by inventing a way for instructions to be written out in a form more easily comprehensible to the user, and making the computer itself translate this into the numbers and codes which are required to use the program on the computer. For example, the programmer can write 'SUBTRACT TAX FROM PAY'; this is then translated for him into the computer code, say 100 326 475, which is then used by the machine to perform the specified calculation. The nearer the language used by the programmer to the code used by the machine, the lower the 'level' of the language. High level languages are what are known as problem orientated, that is, they are designed to simplify the writing of instructions for certain types of problem, either commercial or scientific.

Ex. 1 Read and memorise the international words:

manufacture *n* – , combination *n* – , equivalent *n* – , detail *n* – , code *n* – , machine *n* – , calculation *n* – , translate *v* – , orient *n* – , form *v,n* – ,

Ex. 2 Read the words and learn their meanings:

**vocabulary**

piece *n* – кусок to provide *v* – обеспечивать internal *a* – внутренний circuitry *n* – схемы, цепи each *pron* – каждый own *a* – собственный location *n* – место расположения to identify *v* – определять tedious *a* – утомительный to obey *v* – подчиняться exacting *a* –изнуряющий repetitive *a* – повторяющийся to simplify *v* – упрощать to invent *v* – изобретать comprehensible *a* – понятный attention *n* – внимание

Ex. 3 Learn the word combinations:

**оperate on** – обрабатывать **consist of** – состоять из **storage location** – ячейка памяти **make up** – составлять **work out** – выработать **write out** – выписывать **make computer ~~to~~ translate** – заставить компьютер переводить **that is** – то есть **the bigger…, the heavier** – чем больше …, тем тяжелее …

**to be designed to** – быть предназначенным; предназначаться **either … or …--** или …, или …**;** либо …, либо ….

Ex. 4 Learn to translate the noun chains:

computer languages – , storage location – , computer code – , a high level language – ,

Ex. 5 Answer the questions on the content of the text “**Languages**” (page 27):

1. Are some of the important pieces of software programming languages?
2. Do instructions operate on items of data stored in the processor?
3. What make up instructions?

4. Are computers good at tedious tasks?

5. How can programming be simplified?

6. Why is programming simplified?

1. How is it done?
2. When is the “level” of the language lower?
3. What are high level languages designed for?

Ex. 6 Read and translate the text:

**The Advantages of the Computer**

Some of the first computers cost millions of dollars, but people were quick to learn that it was cheaper to let a million-dollar computer keep track of inventory or print payroll checks than to have a hundred clerks trying to do the same thing by hands. Scientists found that computers made fewer mistakes and could perform the tasks much faster than almost any number of people using manual methods. The demand for computers grew. As the demand grew, the number of factories able to produce computers also grew.

**Lesson XIV**

**People**

We have attempted to explain the concept of a computer as a device (hardware) using a process (software) to turn a raw material into information. But none of this can be achieved without human control over this powerful tool. Perhaps the most exacting task is that which falls on the systems analyst. He has to consult his client, the potential user of the machine, to establish the problem to be solved. He must then examine in detail existing procedures: when he has understood exactly what is required, he must design a complete solution to the problem, covering not only computer procedures, but all associated operations, clerical and other. He has to decide what product he is trying to produce: what sort of information his system is designed to extract. He decides the raw material required to obtain this, and devises the procedures by which this 'raw data' is collected, fed to the computer, processed and finally dispersed to those who are to use it. He must be skilled not only in analysing a problem and devising a solution, but in communicating his solution to the client or management specifying the requirements, to the users of the information, to those involved in collecting raw data and distributing information and to the other computer professionals involved in the development of the computer side of the system.

There are two associated groups of people: programmers and operators. Programmers obtain from the analyst detailed descriptions of specifications of individual jobs which the computer has to perform to make up the overall system. Each such job will require a program to be stored in the computer to perform the tasks. The programmer is responsible for preparing the detailed instructions, and making sure that his program is correct and will in fact perform the required

job accurately. Computer operators are responsible for the day-to-day operation of programs once these have been specified by the analyst and written by the programmer.

They load programs into the processor, check the operation of the various peripheral units and make sure that the correct raw data is read into the computer when required and that the finished product, the information output, is returned to its correct destination.

Ex. 1 Read and memorise international words:

specification *n* – ; concept *n* – ; material *n* – ; analyst *n* – ; to consult *v* – ; client *n* – ; potential *n* – ; to examine *v* – ; to associate *v* –; extract *v,n* – ; to collect *v* –; finish *v,n* – ; professional *a* – ; correct *v,a* – ; accurrate *a* – ; to analise *v* – ; to disperse - *v* ; peripheral *a* – ;

Ex. 2 Read the words and learn their meanings:

**vocabulary**

attempt *v,n* – попытка none *pron* – ни один to achieve *v* – достигать without *prep* – без to fall *v* – падать analyst *n* – аналитик to obtain *v* – получать to devise *v* – изобретать to feed *v* – кормить, подавать to disperse *v* – распространять clerical *a* – предельный to try *v* – стараться to be skilled – быть опытным to involve *v* – включать в to distribute *v* – распределять to prepare v – готовить check *v,n* – проверять , проверка load *v,n* – загружать, нагрузка, загрузка raw *a* – необработанный destination *n* – место назначения

Ex. 3 Learn the word combinations:

**to consult a client** – (про-)консультировать клиента **in detail** – подробно; (в) подробно(- стях) **to be skilled** – быть квалифицированным, умелым, искусным **to make up** – состав- лять; собирать **to be responsible for** – быть ответственным (отвечать) за **to make sure** – убедиться; удостовериться **in fact** – в (на) самом деле; действительно; фактически **day-to-day operation** – повседневная, ежедневная, каждодневная операция

Ex. 4 Learn to translate noun chains:

Read the text again attentively and pick out all the noun chains you can detect:

Ex. 5 a) Study the irregular verbs:

sell – ; bring – ; steal – ; stabd – ; write – .

b) Review the modal verbs and their equivalents:

can – ; must – ; may – ; should – ; would – ; could – ; might –; need – ;

Ex. 6 Answer the questions on the content of the text “**People**” (page 28):

1. How can one explain the concept of a computer?
2. Is it possible to turn a raw material into information without human control over a computer?
3. What kind of work is it, which falls on the system analyst?
4. What should the analyst do to establish the problem to be solved?
5. Must the analyst design a complete solution to the problem?
6. Does he have to decide what product he is trying to produce?
7. What kinds of product must he produce?
8. What procedures must he devise?
9. Whom is processed “raw data” dispersed by?
10. What must the analyst be skilled in?
11. How many associated groups of people are there?
12. What is the programmer responsible for?
13. Who is responsible for the day-to-day operation of prog-rammes?

Ex. 7 Match the beginnings of the sentences to their endings:

**Hardware**

1. Hardware is a term given to all the electronic and mechanical machines …
2. A computer system consists …
3. There is that part of the system which can perform operations as a result of instructions. This part is known …
4. Second, the unit used to store the raw material on which the processor is to set to work: this is known simply …
5. Finally, there are devices a) to place data into storage and …
6. These input devices and output devices …
7. A computer is thus a collective noun for a group of devices: …

**a**. … as a central processor.

**b**. … which together form a computer system.

**c**. … as a storage device.

**d**. … of three basic types of machine.

**e**. … to extract the finished product from the system.

**f**. … are known collectively.

**g**. … central processor, storage, input, and output devices.

**Lesson XV**

**The Future**

As the information explosion multiplies day by day, so the enormous problem of keeping this explosion under control becomes more and more acute. Computers, with their ability to process great quantities of facts at vast speeds, are the only means we have of creating some sort of order out of apparent chaos. Computers can give management the information they need to make correct decisions; politicians the facts on which to base policy decisions; research workers the details of results obtained by others in specific fields; architects up-to-date information about available materials and designs; doctors case histories of patients with specified symptoms; lawyers precedents; airline pilots flight plans. The list is endless, and it is endless just because we are continually devising new tasks for the computer and developing computers to perform them.

Ex. 1 Read out and memorise all the international words from the text “**The Future**” (page 30:

Ex. 2 Read the words and learn their meanings:

**vocabulary**

explosion *n* – взрыв to multiply *v* – умножать enormous *a* – огромный ability *n* – способность, quantity *n* – количество vast *a* – обширный chaos *n* – хаос, беспорядок history *n* – история, архив lawyer *n* – юрист precedent *n* – прецедент airline *n* – авиалиния flight *n* – полёт

Ex. 3 Learn the word combinations:

**day by day** – день за днём **so** – поэтому; следовательно; Так!; так **at a great speed** – на большой скорости **under control** – под управлением (контролем) **out of five** – из пяти **up-to-date** – (вполне) современный; новейший

Ex. 4 Learn to translate **noun chains**:

**policy decision** – решение алгоритма; **метод решения** **research worker** – научный работник **case history** – протокол случая **airline pilot** – лётчик **flight plan** – план полёта

Ex. 5 Answer the questions on the content of the text:

1. How does the information explosion multiply? 2. Does keeping this explosion under control become more and more difficult? 3. What is the only means we have of creating some sort of order out of chaos? 4. Can computers give management the information? 5. What kinds of management can computers give? 6. Why is the list of kinds of management endless?

**Part II**

**Supplementary Reading**

# Computers: History and Development

#### Nothing epitomizes modern life better than the computer. For better or worse, computers have infiltrated every aspect of our society. Today computers do much more than simply compute: [supermarket scanners](http://www.symbol.com/ST000121.HTM) calculate our grocery bill while keeping store inventory; computerized [telephone switching centers](http://www.dca.gov.au/comms_project/ch2.htm) play traffic cop to millions of calls and keep lines of communication untangled; and [automatic teller machines](http://www.visa.com/visa/locator/atm_region.html) let us conduct banking transactions from virtually anywhere in the world. But where did all this technology come from and where is it heading? To fully understand and appreciate the impact computers have on our lives and promises they hold for the future, it is important to understand their evolution.

## Early Computing Machines and Inventors

The [abacus](http://www.ai.mit.edu:/projects/abacus/abacus_obj.gif), which emerged about 5,000 years ago in Asia Minor and is still in use today, may be considered the first computer. This device allows users to make computations using a system of sliding beads arranged on a rack. Early merchants used the abacus to keep trading transactions. But as the use of [paper and pencil](http://hyperg.tu-graz.ac.at:80/CE0E553E/T0x811be681_0x00005f1d) spread, particularly in Europe, the abacus lost its importance. It took nearly 12 centuries, however, for the next significant advance in computing devices to emerge. In 1642, [Blaise Pascal](http://www-groups.dcs.st-and.ac.uk/%7Ehistory/Mathematicians/Pascal.html) (1623-1662), the 18-year-old son of a French tax collector, invented what he called a numerical wheel calculator to help his father with his duties. This brass rectangular box, also called a Pascaline, used eight movable dials to add sums up to eight figures long. Pascal's device used a base of ten to accomplish this. For example, as one dial moved ten notches, or one complete revolution, it moved the next dial - which represented the ten's column - one place. When the ten's dial moved one revolution, the dial representing the hundred's place moved one notch and so on. The drawback to the Pascaline, of course, was its limitation to addition.

In 1694, a German mathematician and philosopher, [Gottfried Wilhem von Leibniz](http://www-groups.dcs.st-and.ac.uk/%7Ehistory/Mathematicians/Leibniz.html) (1646-1716), improved the Pascaline by creating a machine that could also multiply. Like its predecessor, Leibniz's mechanical multiplier worked by a system of gears and dials. Partly by studying Pascal's original notes and drawings, Leibniz was able to refine his machine. The centerpiece of the machine was its stepped-drum gear design, which offered an elongated version of the simple flat gear. It wasn't until 1820, however, that mechanical calculators gained widespread use. Charles Xavier Thomas de Colmar, a Frenchman, invented a machine that could perform the four basic arithmetic functions. Colmar's mechanical calculator, the arithometer, presented a more practical approach to computing because it could add, subtract, multiply and divide. With its enhanced versatility, the arithometer was widely used up until the First World War. Although later inventors refined Colmar's calculator, together with fellow inventors Pascal and Leibniz, he helped define the age of mechanical computation.

The real beginnings of computers as we know them today, however, lay with an English mathematics professor, [Charles Babbage](http://www-groups.dcs.st-and.ac.uk/%7Ehistory/Mathematicians/Babbage.html) (1791-1871). Frustrated at the many errors he found while examining calculations for the Royal Astronomical Society, Babbage declared, "I wish to God these calculations had been performed by steam!" With those words, the automation of computers had begun. By 1812, Babbage noticed a natural harmony between machines and mathematics: machines were best at performing tasks repeatedly without mistake; while mathematics, particularly the production of mathematic tables, often required the simple repetition of steps. The problem centered on applying the ability of machines to the needs of mathematics. Babbage's first attempt at solving this problem was in 1822 when he proposed a machine to perform differential equations, called a [Difference Engine](http://www.londonmall.co.uk/babbage/intro.htm). Powered by steam and large as a locomotive, the machine would have a stored program and could perform calculations and print the results automatically. After working on the Difference Engine for 10 years, Babbage was suddenly inspired to begin work on the first general-purpose computer, which he called the Analytical Engine. Babbage's assistant, [Augusta Ada King, Countess of Lovelace](http://www.scottlan.edu/lriddle/women/love.htm) (1815-1842) and daughter of English poet [Lord Byron](http://copper.ucs.indiana.edu/%7Edcorcora/byron.html), was instrumental in the machine's design. One of the few people who understood the Engine's design as well as Babbage, she helped revise plans, secure funding from the British government, and communicate the specifics of the Analytical Engine to the public. Also, Lady Lovelace's fine understanding of the machine allowed her to create the instruction routines to be fed into the computer, making her the first female computer programmer. In the 1980's, the [U.S. Defense Department](http://www.doe.gov/html/servers/dodservr.html) named a programming language [ADA](http://lglwww.epfl.ch/Ada/) in her honor.

Babbage's steam-powered Engine, although ultimately never constructed, may seem primitive by today's standards. However, it outlined the basic elements of a modern general purpose computer and was a breakthrough concept. Consisting of over 50,000 components, the basic design of the Analytical Engine included input devices in the form of perforated cards containing operating instructions and a "store" for memory of 1,000 numbers of up to 50 decimal digits long. It also contained a "mill" with a control unit that allowed processing instructions in any sequence, and output devices to produce printed results. Babbage borrowed the idea of punch cards to encode the machine's instructions from the Jacquard loom. The loom, produced in 1820 and named after its inventor, Joseph-Marie Jacquard, used punched boards that controlled the patterns to be woven.

In 1889, an American inventor, [Herman Hollerith](http://www-mitpress.mit.edu/mitp/recent-books/comp/building-ibm.html) (1860-1929), also applied the Jacquard loom concept to computing. His first task was to find a faster way to compute the [U.S. census](http://www.census.gov/). The previous census in 1880 had taken nearly seven years to count and with an expanding population, the bureau feared it would take 10 years to count the latest census. Unlike Babbage's idea of using perforated cards to instruct the machine, Hollerith's method used cards to store data information which he fed into a machine that compiled the results mechanically. Each punch on a card represented one number, and combinations of two punches represented one letter. As many as 80 variables could be stored on a single card. Instead of ten years, census takers compiled their results in just six weeks with Hollerith's machine. In addition to their speed, the punch cards served as a storage method for data and they helped reduce computational errors. Hollerith brought his punch card reader into the business world, founding Tabulating Machine Company in 1896, later to become [International Business Machines (IBM)](http://www.ibm.com/) in 1924 after a series of mergers. Other companies such as [Remington Rand and Burroughs](http://press-gopher.uchicago.edu:70/CGI/cgi-bin/hfs.cgi/99/princeton/92025399.ctl) also manufactured punch readers for business use. Both business and government used punch cards for data processing until the 1960's.

In the ensuing years, several engineers made other significant advances. [Vannevar Bush](http://succeed.edtech.vt.edu/Htmls/P174013.html)(1890-1974) developed a calculator for solving differential equations in 1931. The machine could solve complex differential equations that had long left scientists and mathematicians baffled. The machine was cumbersome because hundreds of gears and shafts were required to represent numbers and their various relationships to each other. To eliminate this bulkiness, [John V. Atanasoff](http://www.cs.iastate.edu/jva/jva-archive.html) (b. 1903), a professor at Iowa State College (now called [Iowa State University](http://www.iastate.edu/)) and his graduate student, Clifford Berry**,** envisioned an all-electronic computer that applied Boolean algebra to computer circuitry. This approach was based on the mid-19th century work of [George Boole](http://www.dia.eui.upm.es/asignatu/sis_op1/comp_hd/boole.html) (1815-1864) who clarified the binary system of algebra, which stated that any mathematical equations could be stated simply as either true or false. By extending this concept to electronic circuits in the form of on or off, Atanasoff and Berry had developed the first all-electronic computer by 1940. Their project, however, lost its funding and their work was overshadowed by similar developments by other scientists.

## Five Generations of Modern Computers

### vac_tubeFirst Generation (1945-1956)

With the onset of the [Second World War](http://www.lib.muohio.edu/%7Eskimmel/wwii/), governments sought to develop computers to exploit their potential strategic importance. This increased funding for computer development projects hastened technical progress. By 1941 German engineer [Konrad Zuse](http://ei.cs.vt.edu/%7Ehistory/Zuse.html) had developed a computer, the Z3, to design [airplanes](http://aero.stanford.edu/) and missiles. The Allied forces, however, made greater strides in developing powerful computers. In 1943, the British completed a secret code-breaking computer called [Colossus](http://ei.cs.vt.edu/%7Ehistory/colossus.rebuilt.html) to decode [German messages](http://ei.cs.vt.edu/%7Ehistory/Enigma.small.GIF). The Colossus's impact on the development of the computer industry was rather limited for two important reasons. First, Colossus was not a general-purpose computer; it was only designed to decode secret messages. Second, the existence of the machine was kept secret until decades after the war.

American efforts produced a broader achievement. Howard H. Aiken (1900-1973), a Harvard engineer working with IBM, succeeded in producing an all-electronic calculator by 1944. The purpose of the computer was to create ballistic charts for the [U.S. Navy](http://www.navy.mil/). It was about half as long as a football field and contained about 500 miles of wiring. The Harvard-IBM Automatic Sequence Controlled Calculator, or Mark I for short, was a electronic relay computer. It used electromagnetic signals to move mechanical parts. The machine was slow (taking 3-5 seconds per calculation) and inflexible (in that sequences of calculations could not change); but it could perform basic arithmetic as well as more complex equations.

Another computer development spurred by the war was the Electronic Numerical Integrator and Computer ([ENIAC](http://ei.cs.vt.edu/%7Ehistory/ENIAC.1.GIF)), produced by a partnership between the U.S. government and the [University of Pennsylvania](http://www.upenn.edu:80/). Consisting of 18,000 vacuum tubes, 70,000 resistors and 5 million soldered joints, the computer was such a massive piece of machinery that it consumed 160 kilowatts of electrical power, enough energy to dim the lights in an entire section of [Philadelphia](http://www.usc.edu/dept/cs/personal/jdevlin/Philly.html). Developed by [John Presper Eckert](http://www-groups.dcs.st-and.ac.uk/%7Ehistory/ShortBiogs/E.html#Eckert) (1919-1995)andJohn W. Mauchly (1907-1980), ENIAC, unlike the Colossus and Mark I, was a general-purpose computer that computed at speeds 1,000 times faster than Mark I.

In the mid-1940's [John von Neumann](http://ei.cs.vt.edu/%7Ehistory/VonNeumann.html) (1903-1957) joined the University of Pennsylvania team, initiating concepts in computer design that remained central to computer engineering for the next 40 years. Von Neumann designed the Electronic Discrete Variable Automatic Computer ([EDVAC](http://ftp.arl.mil/ftp/historic-computers/jpeg/edvac1.jpg)) in 1945 with a memory to hold both a stored program as well as data. This "stored memory" technique as well as the "conditional control transfer," that allowed the computer to be stopped at any point and then resumed, allowed for greater versatility in computer programming. The key element to the von Neumann architecture was the central processing unit, which allowed all computer functions to be coordinated through a single source. In 1951, the [UNIVAC I](http://www.cc.gatech.edu/services/unisys-folklore/) (Universal Automatic Computer), built by Remington Rand, became one of the first commercially available computers to take advantage of these advances. Both the [U.S. Census Bureau](http://www.census.gov/) and [General Electric](http://www.ge.com/) owned UNIVACs. One of UNIVAC's impressive early achievements was predicting the winner of the 1952 presidential election, [Dwight D. Eisenhower](http://history.cc.ukans.edu/heritage/abilene/ikectr.html).

[[vt-phosm](http://www.dia.eui.upm.es/asignatu/sis_op1/comp_hd/vt-photo.html)](http://www.dia.eui.upm.es/asignatu/sis_op1/comp_hd/vt-photo.html)First generation computers were characterized by the fact that operating instructions were made-to-order for the specific task for which the computer was to be used. Each computer had a different binary-coded program called a machine language that told it how to operate. This made the computer difficult to program and limited its versatility and speed. Other distinctive features of first generation computers were the use of [vacuum tubes](http://www.dia.eui.upm.es/asignatu/sis_op1/comp_hd/vt-photo.html) (responsible for their breathtaking size) and magnetic drums for data storage.

### transistSecond Generation Computers (1956-1963)

[[tr-phosm](http://www.dia.eui.upm.es/asignatu/sis_op1/comp_hd/tr-photo.html)](http://www.dia.eui.upm.es/asignatu/sis_op1/comp_hd/tr-photo.html)By 1948, the invention of the [transistor](http://www.dia.eui.upm.es/asignatu/sis_op1/comp_hd/tr-photo.html) greatly changed the computer's development. The transistor replaced the large, cumbersome vacuum tube in televisions, radios and computers. As a result, the size of electronic machinery has been shrinking ever since. The transistor was at work in the computer by 1956. Coupled with early advances in magnetic-core memory, transistors led to second generation computers that were smaller, faster, more reliable and more energy-efficient than their predecessors. The first large-scale machines to take advantage of this transistor technology were early supercomputers, Stretch by IBM and LARC by Sperry-Rand. These computers, both developed for atomic energy laboratories, could handle an enormous amount of data, a capability much in demand by atomic scientists. The machines were costly, however, and tended to be too powerful for the business sector's computing needs, thereby limiting their attractiveness. Only two LARCs were ever installed: one in the [Lawrence Radiation Labs](http://www.llnl.gov/) in Livermore, California, for which the computer was named (Livermore Atomic Research Computer) and the other at the [U.S. Navy Research and Development Center](http://www.nrl.navy.mil/) in [Washington, D.C](http://www.proxima.com/dc/tourist/). Second generation computers replaced machine language with assembly language, allowing abbreviated programming codes to replace long, difficult binary codes.

Throughout the early 1960's, there were a number of commercially successful second generation computers used in business, universities, and government from companies such as Burroughs, [Control Data](http://www.cdc.com/), [Honeywell](http://www.honeywell.com/), IBM, Sperry-Rand, and others. These second generation computers were also of solid state design, and contained transistors in place of vacuum tubes. They also contained all the components we associate with the modern day computer: printers, tape storage, disk storage, memory, operating systems, and stored programs. One important example was the IBM 1401, which was universally accepted throughout industry, and is considered by many to be the Model T of the [computer industry](http://www.meu.edu/jcn/programs/programs.html). By 1965, most large business routinely processed financial information using second generation computers.

It was the stored program and programming language that gave computers the flexibility to finally be cost effective and productive for business use. The stored program concept meant that instructions to run a computer for a specific function (known as a program) were held inside the computer's memory, and could quickly be replaced by a different set of instructions for a different function. A computer could print customer invoices and minutes later design products or calculate paychecks. More sophisticated high-level languages such as [COBOL](http://www.cobol.org/) (Common Business-Oriented Language) and [FORTRAN](http://boris.qub.ac.uk/edward/computers/Fortran.html) (Formula Translator) came into common use during this time, and have expanded to the current day. These languages replaced cryptic binary machine code with words, sentences, and mathematical formulas, making it much easier to program a computer. New types of careers (programmer, analyst, and computer systems expert) and the entire [software industry](http://www.spa.org/default.htm) began with second generation computers.

### chiptopThird Generation Computers (1964-1971)

Though transistors were clearly an improvement over the vacuum tube, they still generated a great deal of heat, which damaged the computer's sensitive internal parts. The quartz rock eliminated this problem.Jack Kilby, an engineer with [Texas Instruments](http://www.ti.com/), developed the integrated circuit (IC) in 1958. The IC combined three electronic components onto a small silicon disc, which was made from quartz. Scientists later managed to fit even more components on a single chip, called a semiconductor. As a result, computers became ever smaller as more components were squeezed onto the chip. Another third-generation development included the use of an [operating system](http://www.yahoo.com/Computers_and_Internet/Operating_Systems/) that allowed machines to run many different programs at once with a central program that monitored and coordinated the computer's memory.

### floppyFourth Generation (1971-Present)

After the integrated circuits, the only place to go was down - in size, that is. Large scale integration (LSI) could fit hundreds of components onto one chip. By the 1980's, very large scale integration (VLSI) squeezed hundreds of thousands of components onto a chip. Ultra-large scale integration (ULSI) increased that number into the millions. The ability to fit so much onto an area about half the size of a U.S. dime helped diminish the size and price of computers. It also increased their power, efficiency and reliability. The [Intel](http://www.intel.com/) 4004 chip, developed in 1971, took the integrated circuit one step further by locating all the components of a computer (central processing unit, memory, and input and output controls) on a minuscule chip. Whereas previously the integrated circuit had had to be manufactured to fit a special purpose, now one microprocessor could be manufactured and then programmed to meet any number of demands. Soon everyday household items such as [microwave ovens](http://www.homecom.com/appliances/dept06.html), television sets and [automobiles](http://www.laguna-seca.com/auto.htm) with electronic [fuel injection](http://www.wp.com/HarveyRacing/page2.html) incorporated microprocessors.

Such condensed power allowed everyday people to harness a computer's power. They were no longer developed exclusively for large business or government contracts. By the mid-1970's, computer manufacturers sought to bring computers to general consumers. These minicomputers came complete with user-friendly software packages that offered even non-technical users an array of applications, most popularly word processing and spreadsheet programs. Pioneers in this field were [Commodore](http://www.dia.eui.upm.es/asignatu/sis_op1/comp_hd/commodr.html), [Radio Shack](http://www.eznet.com/%7Evt/rs/rs.htm) and [Apple Computers](http://www.apple.com/default.html). In the early 1980's, [arcade video games](http://sharkie.psych.indiana.edu/rynersw/vids/vids.html) such as [Pac Man](http://sharkie.psych.indiana.edu/rynersw/vids/coinop/coinop.html) and [home video game systems](http://www2.ecst.csuchico.edu/%7Egchance/) such as the Atari 2600 ignited consumer interest for more sophisticated, programmable home computers.

In 1981, IBM introduced its personal computer (PC) for use in the home, office and schools. The 1980's saw an expansion in computer use in all three arenas as clones of the IBM PC made the personal computer even more affordable. The number of personal computers in use more than doubled from 2 million in 1981 to 5.5 million in 1982. Ten years later, 65 million PCs were being used. Computers continued their trend toward a smaller size, working their way down from desktop to laptop computers (which could fit inside a briefcase) to palmtop (able to fit inside a breast pocket). In direct competition with IBM's PC was Apple's Macintosh line, introduced in 1984. Notable for its user-friendly design, the Macintosh offered an operating system that allowed users to move screen icons instead of typing instructions. Users controlled the screen cursor using a mouse, a device that mimicked the movement of one's hand on the computer screen.

As computers became more widespread in the workplace, new ways to harness their potential developed. As smaller computers became more powerful, they could be linked together, or networked, to share memory space, software, information and communicate with each other. As opposed to a mainframe computer which was one powerful computer that shared time with many terminals for many applications, networked computers allowed individual computers to form electronic co-ops. Using either direct wiring, called a [Local Area Network (LAN)](http://umbc7.umbc.edu/%7Epcucin1/lantutorial/page1.html), or telephone lines, these networks could reach enormous proportions. A global web of computer circuitry, the [Internet](http://info.isoc.org/home.html), for example, links computers worldwide into a single network of information. During the 1992 U.S. presidential election, vice-presidential candidate [Al Gore](http://www.whitehouse.gov/White_House/EOP/OVP/html/GORE_Home.html) promised to make the development of this so-called "information superhighway" an administrative priority. Though the possibilities envisioned by Gore and others for such a large network are often years (if not decades) away from realization, the most popular use today for computer networks such as the Internet is electronic mail, or E-mail, which allows users to type in a computer address and send messages through networked terminals across the office or across the world.

### Fifth Generation (Present and Beyond)

Defining the fifth generation of computers is somewhat difficult because the field is in its infancy. The most famous example of a fifth generation computer is the fictional [HAL9000](http://www.tardis.ed.ac.uk/%7Emartin/hal9000) from [Arthur C. Clarke](http://www.lsi.usp.br/%7Erbianchi/clarke/ACC.Homepage.html)'s novel, [2001: A Space Odyssey](http://pubweb.acns.nwu.edu/%7Egdd816/2001.html)*.* HAL performed all of the functions currently envisioned for real-life fifth generation computers. With [artificial intelligence](http://www.cs.washington.edu/research/jair/home.html), HAL could reason well enough to hold conversations with its human operators, use visual input, and learn from its own experiences. (Unfortunately, HAL was a little too human and had a psychotic breakdown, commandeering a spaceship and killing most humans on board.)

Though the wayward HAL9000 may be far from the reach of real-life computer designers, many of its functions are not. Using recent engineering advances, computers may be able to accept [spoken word instructions](http://159.226.21.1/nlpr/Index.html) and imitate human reasoning. The ability to translate a foreign language is also a major goal of fifth generation computers. This feat seemed a simple objective at first, but appeared much more difficult when programmers realized that human understanding relies as much on context and meaning as it does on the simple translation of words.

Many advances in the science of computer design and technology are coming together to enable the creation of fifth-generation computers. Two such engineering advances are parallel processing, which replaces von Neumann's single central processing unit design with a system harnessing the power of many CPUs to work as one. Another advance is [superconductor](http://www.nist.gov/item/NIST_Superconductor_Integrated-Circuit_Fabircation_Laboratory.html) technology, which allows the flow of electricity with little or no resistance, greatly improving the speed of information flow. Computers today have some attributes of fifth generation computers. For example, expert systems assist doctors in making diagnoses by applying the problem-solving steps a doctor might use in assessing a patient's needs. It will take several more years of development before expert systems are in widespread use.

**Masters** **of Software Engineering**

The Master of Science Degree in Software Engineering provides specialized knowledge and experience in developing and modifying large, complex software systems. It emphasizes technical and management aspects of the software engineering process. Software engineering is an established discipline based on requirement analysis, design, construction, testing, maintenance, economics, and management issues of software engineering. A pragmatic approach to problem solving is the hallmark of a software engineer. Software engineers are concerned with the theoretical and practical aspects of technology, cost, and social impact of software systems that are both effective and efficient.

Software engineers are in demand in every segment of society affected by computing technology. Potential employers include all software vendors and Internet-based companies, electronic business organizations, businesses that build and sell computers, research and development laboratories, aerospace companies, government contractors, banks, insurance companies, and manufacturing organizations. The master’s program is concerned with both technical and managerial issues, but primary emphasis is placed on the technical aspects of building and modifying high quality software systems.

Successful applicants have a broad variety of undergraduate backgrounds, including computer science, science and mathematics, engineering, liberal arts, and business.  Many of our students are working or have worked in the software industry.

The contents of the MS in Software Engineering program are revised on a regular basis to stay abreast of the latest developments in information technology. Recent additions to the program include software construction with the object-oriented Java programming language, requirements analysis with use cases and the Unified Modeling Language (UML), object-oriented software design with the UML, graphical user interface design, Web applications and services, software project management using the spiral life cycle model and the Capability Maturity Model (CMM), software architecture, reusable software architectures and product lines, design patterns, system testing and testing of object-oriented components, secure software design and programming, and formal methods using the Object Constraint Language (OCL).

|  |
| --- |
| **Undergraduate** Courses   [Go to grad courses](http://www.cs.gmu.edu/programs/masters/swe/swe-course.html#swe619)SWE 205 - Software Usability Analysis and Design (3:3:0). Prerequisite: None. Principles of user interface design. Concepts for objectively and quantitatively assessing the usability of software user interfaces. Outcomes include knowledge of quantitative engineering principles for designing usable software interfaces and an understanding that usability is more important than efficiency for almost all modern software projects, and often the primary factor that leads to product success. Major topics include cognitive models for human perceptions and needs, which are used as a basis for analytical and critical thinking about user interfaces; specific engineering principles for designing usable menus, forms, command languages, web sites, graphical user interfaces and web-based user interfaces. Assessments will include written analytical evaluations of existing user interfaces, exams, and HTML-based design projects. |
| SWE 301 - **Internship Preparation** (0:0:0). Prerequisite: Limited to ACS or CS majors with junior standing or permission of instructor. Preparation for Internship Educational Experience. Intended for, but not limited to, students planning internships in the Applied Computer Science Software Engineering Program. Internship employment opportunities. Basic interview skills. Techniques for applying academic knowledge to practical software development. Techniques for extracting knowledge from practical experience. Peer presentation from students who have completed internships. |
| SWE 332 **Object-Oriented Software Design and Implementation** (3:3:0). Prerequisite: CS 211. In-depth study of software design and implementation using a modern, object-oriented language with support for graphical user interfaces and complex data structures. Topics cover specifications, design patterns, and abstraction techniques, including typing, access control, inheritance, and polymorphism. Students will learn the proper engineering use of techniques such as information hiding, classes, objects, inheritance, exception handling, event-based systems, and concurrency. |
| SWE 401 - **Internship Reflection** (1:1:0). Prerequisite: SWE 301 and completion of internship. Reflection on Internship Educational Experience. Intended for, but not limited to, students completing internships in the Applied Computer Science Software Engineering Program. Analysis of techniques for applying academic knowledge to practical software development. Analysis of techniques for extracting knowledge from practical experience. Student presentations summarizing internships relating them to academic program goals. |
| SWE 421 **Software Requirements and Design Modeling** (3:3:0). Prerequisite: CS 211. An introduction to concepts, methods, and tools for the creation of large-scale software systems. Methods, tools, notations, and validation techniques to analyze, specify, prototype, and maintain software requirements. Introduction to object-oriented requirements modeling, including use case modeling, static modeling, and dynamic modeling using the Unified Modeling Language (UML) notation. Concepts and methods for the design of large-scale software systems. Fundamental design concepts and design modeling using UML notation. Students participage in a group project on software requirements, specification, and object-oriented software design. |
| SWE 432 **Design and Implementation of Software for the Web** (3:3:0). Prerequisite: Math 125 and CS 421. This course teaches students how to develop software for web applications. The concepts of client-server computing, theories of usable graphical user interfaces, and models for web-based information retrieval and processing are covered. Goals are to understand how to design usable software interfaces and implement them on the web, learn how to build software that accepts information from users across the web and returns data to the user, and understand how to interact with database engines to store and retrieve information. Specific topics that are included are HTML, CGI programming, Java, Java applets, Javascripts, and Java servlets. |
| SWE 437 **Software Testing and Maintenance** (3:3:0). Prerequisite: CS 211 and Math 125. Concepts and techniques for testing and modifying software in evolving environments. Topics include software testing at the unit, module, subsystem, and system levels; developer testing; automatic and manual techniques for generating test data; testing concurrent and distributed software; designing and implementing software to increase maintainability and reuse; evaluating software for change; and validating software changes. |
| SWE 443 **Software Architectures** (3:3:0). Prerequisite: SWE 421. This course teaches how to design, understand, and evaluate software systems at an architectural level of abstraction. By the end of the course, students will be able to recognize major architectural styles in existing software systems, describe a system's architecture accurately, generate architectural alternatives to address a problem and choose from among them, design a medium-sized software system that satisfies a specification of requirements, use existing tools to expedite software design, and evaluate the suitability of a given architecture in meeting a set of system requirments. |

**Software Engineering**

Software engineering (SE) is concerned with developing and maintaining software systems that behave reliably and efficiently, are affordable to develop and maintain, and satisfy all the requirements that customers have defined for them. It is important because of the impact of large, expensive software systems and the role of software in safety-critical applications. It integrates significant mathematics, computer science and practices whose origins are in engineering.

Students can find software engineering in two contexts: computer science programs offering one or more software engineering courses as elements of the CS curriculum, and in separate software engineering programs. Degree programs in computer science and in software engineering tend to have many courses in common; however, as of Spring 2006 there are few SE programs at the bachelor’s level. Software engineering focuses on software development and goes beyond programming to include such things as eliciting customers’ requirements, and designing and testing software. SE students learn how to assess customer needs and develop usable software that meets those needs.

Both computer science and software engineering curricula typically require a foundation in programming fundamentals and basic computer science theory. They diverge in their focus beyond these core elements. Computer science programs tend to keep the core small and then expect students to choose among more advanced courses (such as systems, networking, database, artificial intelligence, theory, etc.). In contrast, SE programs generally expect students to focus on a range of topics that are essential to the SE agenda (problem modeling and analysis, software design, software verification and validation, software quality, software process, software management, etc.). While both CS and SE programs typically require students to experience team project activity, SE programs tend to involve the students in significantly more of it, as effective team processes are essential to effective SE practices. In addition, a key requirement specified by the SE curriculum guidelines is that SE students should learn how to build software that is genuinely useful and usable by the customer and satisfies all the requirements defined for it.

Most people who now function in the U.S. as serious software engineers have degrees in computer science, not in software engineering. In large part this is because computer degrees have been widely available for more than 30 years and software engineering degrees have not. Positions that require development of large software systems often list “Software Engineer” as the position title. Graduates of computer science, computer engineering, and software engineering programs are good candidates for those positions, with the amount of software engineering study in the programs determining the suitability of that graduate for such a position.

Most IT professionals who have computing degrees come from CS or IS programs. It is far too soon for someone who wants to work as a software engineer or as an information technology practitioner to be afraid that they won’t have a chance if they don’t graduate from a degree program in one of the new disciplines. In general, a CS degree from a respected program is the most flexible of degrees and can open doors into the professional worlds of CS, SE, IT, and sometimes CE. A degree from a respected IS program allows entry to both IS and IT careers.

Media attention to outsourcing, offshoring, and job migration has caused many to be concerned about the future of computing-related careers. It is beyond the scope of this web site to address these issues. The [report of the British Computer Society](http://www.bcs.org/BCS/News/PositionsAndResponses/Positions/offshore/offshorereport.htm) addresses these issues as they impact the U.K. The [Globalization Report](http://www.acm.org/globalizationreport/) of the ACM Job Migration Task Force reflects an international perspective, not just a U.S-centric one.

**Computer Science**

Computer science (CS) spans the range from theory through programming to cutting-edge development of computing solutions. Computer science offers a foundation that permits graduates to adapt to new technologies and new ideas. The work of computer scientists falls into three categories: a) designing and building software; b) developing effective ways to solve computing problems, such as storing information in databases, sending data over networks or providing new approaches to security problems; and c) devising new and better ways of using computers and addressing particular challenges in areas such as robotics, computer vision, or digital forensics (although these specializations are not available in all computer science prog- rams). Most computer science programs require some mathematical background.

Let us consider what is involved in a career path in each area.

**Career Path 1**: Designing and implementing software. This refers to the work of software development which has grown to include aspects of web development, interface design, security issues, mobile computing, and so on. This is the career path that the majority of computer science graduates follow. While a bachelor’s degree is generally sufficient for entry into this kind of career, many software professionals return to school to obtain a terminal master’s degree. (Rarely is a doctorate involved.) Career opportunities occur in a wide variety of settings including large or small software companies, large or small computer services companies, and large organizations of all kinds (industry, government, banking, healthcare, etc.). Degree programs in software engineering also educate students for this career path.

**Career Path 2**: Devising new ways to use computers. This refers to innovation in the application of computer technology. A career path in this area can involve advanced graduate work, followed by a position in a research university or industrial research and development laboratory; it can involve entrepreneurial activity such as was evident during the dot-com boom of the 1990s; or it can involve a combination of the two.

**Career Path 3**: Developing effective ways to solve computing problems. This refers to the application or development of computer science theory and knowledge of algorithms to ensure the best possible solutions for computationally intensive problems. As a practical matter, a career path in the development of new computer science theory typically requires graduate work to the Ph.D. level, followed by a position in a research university or an industrial research and development laboratory.

**Career Path 4**: Planning and managing organizational technology infrastructure. This is the type of work for which the new information technology (IT) programs explicitly aim to educate students. Career paths 2 and 3 are undenably in the domain of computer science graduates. Career paths 1 and 4 have spawned the new majors in software engineering and information technology, respectively, and information systems graduates often follow Career path 1, too. Computer scientists continue to fill these positions, but programs in software engineering, information technology, and information systems offer alternative paths to these careers.

**Computer Engineering**

Computer engineering (CE) students study the design of digital hardware and software systems including communications systems, computers and devices that contain computers. For them, programming is focused on digital devices and their interfaces with users and other devices. An important area within computing engineering is the development of embedded systems. Devices such as cell phones, digital audio players, digital video recorders, alarm systems, x-ray machines, and laser surgical tools all require integration of hardware and embedded software, and are all the result of computer engineering.

Computer engineering majors are offered by a fairly large number of universities, almost always within engineering. This major requires significant study of mathematics.

Read more about Aaron and other computing majors on our [Faces of Computing](http://computingcareers.acm.org/?page_id=61) page.

[Excerpt from Bureau of Labor Statistics site  
**Occupational Outlook Handbook**](http://www.bls.gov/oco/ocos027.htm):

Computer hardware engineers research, design, develop, test, and oversee the installation of computer hardware and supervise its manufacture and installation. Hardware refers to computer chips, circuit boards, computer systems, and related equipment such as keyboards, modems, and printers…. The work of computer hardware engineers is very similar to that of electronics engineers, but, unlike electronics engineers, computer hardware engineers work exclusively with computers and computer-related equipment. The rapid advances in computer technology are largely a result of the research, development, and design efforts of computer hardware engineers.

**Information Systems**

Information systems (IS) is concerned with the information that computer systems can provide to aid a company, non-profit or governmental organization in defining and achieving its goals. It is also concerned with the processes that an enterprise can implement and improve using information technology. IS professionals must understand both technical and organizational factors, and must be able to help an organization determine how information and technology-enabled business processes can provide a foundation for superior organizational performance. They serve as a bridge between the technical and management communities within an organization.

What information does the enterprise need? How is that information generated? Is it delivered to the people who need it? Is it presented to them in ways that permit them to use it readily? Is the organization structured to be able to use technology effectively? Are the business processes of the organization well designed? Do they use the opportunities created by information technology fully? Does the organization use the communication and collaboration capabilities of information technologies appropriately? Is the organization capable of adapting quickly enough to changing external circumstances? These are the important issues that businesses rely on IS people to address.

A majority of IS programs are located in business schools; however, they may have different names such as management information systems, computer information systems, or business information systems. All IS degrees combine business and computing topics, but the emphasis between technical and organizational issues varies among programs. For example, programs differ substantially in the amount of programming required.

Traditionally, many graduates of IS programs have functioned in roles that are similar to the roles for which IT programs explicitly prepare their students. Information systems graduates continue to fill these roles, but the new programs in information technology offer an alternative path to these positions.

**Information Technology**

Information technology (IT) is a label that has two meanings. In common usage, the term “information technology” is often used to refer to all of computing. As a name of an undergraduate degree program, it refers to the preparation of students to meet the computer technology needs of business, government, healthcare, schools, and other kinds of organizations.

IT professionals possess the right combination of knowledge and practical, hands-on expertise to take care of both an organization’s information technology infrastructure and the people who use it. They assume responsibility for selecting hardware and software products appropriate for an organization. They integrate those products with organizational needs and infrastructure, and install, customize and maintain those applications, thereby providing a secure and effective environment that supports the activities of the organization’s computer users. In IT, programming often involves writing short programs that typically connect existing components (scripting).

Planning and managing an organization’s IT infrastructure is a difficult and complex job that requires a solid foundation in applied computing as well as management and people skills. Those in the IT discipline require special skills – in understanding, for example, how networked systems are composed and structured, and what their strengths and weaknesses are. There are important software systems concerns such as reliability, security, usability, and effectiveness and efficiency for their intended purpose; all of these concerns are vital. These topics are difficult and intellectually demanding.

**Mixed Disciplinary Majors**

Because computing is such an important and dynamic field, many interdisciplinary majors, some very recent developments, exist at some schools. Here are are just a few examples of these opportunities. Some of these programs are offered at a number of U.S. schools as of Spring 2006; some only at a handful of U.S. schools.

Bioinformatics combines elements from at least biology, biochemistry, and computer science, and prepares students for careers in the biotechnology and pharmaceutical industries, or for graduate school in informatics. Some programs may also include elements from information systems, chemistry, mathematics, and statistics.

Computational science means science done computationally, and serves as a bridge between computing technology and basic sciences. It blends several fields including computer science, applied mathematics, and one or more application sciences (such as physics, chemistry, biology, engineering, earth sciences, business and others). Some programs also include information systems.

Computer Science and Mathematics combines computer science with mathematics of course. Some of these programs are found at schools that do not have a full major in computer science; some are found at universities with very large computer science departments.

**Gaming and Animation.** Majors for students interested in creating computer games and computer animations are being developed at a number of schools. These majors have various flavors and may combine either or both of computer science and information technology work with either or both of art and (digital) media studies.

Medical (or health) informatics programs are for students interested in students who want to work in a medical environment. Some students will work as technology experts for hospitals; some in public health; some students may be premed or pre-dental. Coursework may be drawn from any or all of computer science, information systems, or information technology in combination with biology, chemistry, and courses unique to this interdisciplinary field.

Be aware that especially in the newer interdisciplinary areas, different schools use different names for the same subject. For example, one school’s “bioinformatics” may be another school’s “computational biology.”

**What Computing Professionals Do**

There are numerous creative, fun, and interesting jobs and careers in computing fields. In this section you will find examples that show a variety of interesting careers available to graduates of computing programs, with testimonials from young computing professionals, excerpts from articles and job listings. Follow the links back to the original sites for further information and more choices.

According to [CNN Money Magazine’s report on the “best jobs in America,”](http://money.cnn.com/magazines/moneymag/bestjobs/2006/) the No. 1 “Best Job in America” in 2006 was [software engineering](http://money.cnn.com/magazines/moneymag/bestjobs/2006/snapshots/1.html), ahead even of finance, medicine, real estate, and law. The number of software engineering jobs is expected to grow by 46% in the next ten years.

There are also numerous job listings seeking computing professionals on [careers.com](http://www.careers.com/), the [IEEE’s careers site](http://careers.ieee.org/), and elsewhere. The [ACM Career Resource Centre](http://www.acm.org/careernews/), the [U.S. Bureau of Labor Statistics](http://www.bls.gov/oco/oco1002.htm), [career-space.com](http://career-space.com/lifestyles/index2.htm), and many other sites offer testimonials and interviews with professionals currently working in computing careers.

**Brochure and Poster**

You may have already seen our brochure. It is designed to introduce you to the various disciplines that are part of computing, and to give you a sense of the many opportunities available in this ever-expanding field. Download the different versions of the brochure below!

[Black & White, 8-1/2 x 11, for High School Students](http://computingcareers.acm.org/wp-content/uploads/2007/02/careerbw.pdf)

[Full-color, 8-1/2 x 11, for High School Students, Spanish version!](http://computingcareers.acm.org/wp-content/uploads/2007/04/careersbrospanish-04-16-07.pdf)

[Full-color, original size (~ 20 x 9) for High School Students](http://www.acm.org/membership/CareersBrochure.pdf)

Cover1Back  
   

**Computing Careers Poster**

This engaging poster, developed in cooperation with CSTA, ACM-W, and ASCA, invites students (especially young women) to consider a career in the computing disciplines and makes connections between popular professions and computer science.

|  |
| --- |
| Classes Taught |

 Spring 2011 Classes

SWE 205 - [Software Usability Analysis and Design](http://cs.gmu.edu/%7Eoffutt/classes/205/) (Tuesday/Thursday, 3:00-4:15, [AB 2026](http://eagle.gmu.edu/map/fairfax.php?building=avt))

SWE 632 - [User Interface Design and Development](http://cs.gmu.edu/%7Eoffutt/classes/632/) (Monday, 4:30-7:20, [IN 206](http://eagle.gmu.edu/map/fairfax.php?building=innovation_hall))

Spring office hours: Wed 2:30-4:00

 Fall 2011 Classes (tentative)

SWE 637 - [Software Testing and Quality Evaluation](http://cs.gmu.edu/%7Eoffutt/classes/637) (Monday, 4:30-7:20)

SWE 642 - [Software Engineering for the World Wide Web](http://cs.gmu.edu/%7Eoffutt/classes/642/) (Tuesday, 4:30-7:20)

 Other Classes

SWE 432 - [Design and Implementation of Software for the Web](http://cs.gmu.edu/%7Eoffutt/classes/432/) (last taught, Fall 2007)

SWE 437 - [Software Testing and Maintenance](http://cs.gmu.edu/%7Eoffutt/classes/437/) (Last taught, Spring 2008)

INFS 590 - [Program Design and Data Structures](http://cs.gmu.edu/%7Eoffutt/classes/590) (last taught, Fall 1999)

SWE 619 - [Software Construction](http://cs.gmu.edu/%7Eoffutt/classes/619) (last taught, Fall 1997)

SWE 626 - [Software Project Laboratory](http://cs.gmu.edu/%7Eoffutt/classes/626/) (last taught, Fall 2000)

SWE 763 - [Software Engineering Experimentation](http://cs.gmu.edu/%7Eoffutt/classes/763/) (Last taught, Fall 2008)

SWE 825 / IT 825 - [Special Topics in Web-Based Software](http://cs.gmu.edu/%7Eoffutt/classes/821/) (last taught, Spring 2007)

IT 821 / SWE 699 - [OO and Architecture-based Testing](http://cs.gmu.edu/%7Eoffutt/classes/821-ootest/) (last taught, Spring 2000)

IT 824 - [Analysis of Software for Testing](http://cs.gmu.edu/%7Eoffutt/classes/824/) (last taught, Fall 1999)

 [ACS Software Engineering undergraduate program](http://www.cs.gmu.edu/programs/undergraduate/acs/BSACSSWEBrochure.pdf) (PDF)

 [Minor in Software Engineering](http://www.cs.gmu.edu/programs/undergraduate/minor/swe/)

 [Some talks for PhD students](http://www.cs.gmu.edu/~offutt/classes/phd/)

 [A few of my web applications](http://cs.gmu.edu/%7Eoffutt/jeffwebapps)

|  |
| --- |
| Graduate Programs |

[MS program in Software Engineering](http://www.cs.gmu.edu/programs/masters/swe/)

[Certificate in Software Engineering](http://catalog.gmu.edu/preview_program.php?catoid=15&poid=6190&bc=1)

Certificate in Web-based Software Engineering

[PhD program in Information Technology](http://ite.gmu.edu/PhDprogr/)

Concentration in Software Engineering ([Local description](http://www.cs.gmu.edu/programs/phd/it/swe/)) ([Catalog](http://www.gmu.edu/catalog/ite/iprograms.html#Anchor24))

[IT Engineer Degree](http://ite.gmu.edu/engineer/)

[Dr. Nash's FAQ list](http://mason.gmu.edu/~snash/PhDFAQ.html)

[Dr. Nash's orientation notes](http://mason.gmu.edu/~snash/orientation.html)

[PhD program in Computer Science](http://www.cs.gmu.edu/programs/phd/)

|  |
| --- |
| Research and Publications |

It would be easier to write papers on testing if you didn't have to contemplate actually using the stuff you are writing about.   
- Richard Carver

[Current Research Project Descriptions](http://cs.gmu.edu/%7Eoffutt/rsrch/Projects.html)     |     [Invited Talks, Tutorials, and Short Courses](http://cs.gmu.edu/%7Eoffutt/talks.html)     |     [A Java mutation tool](http://cs.gmu.edu/%7Eoffutt/mujava/)

[Web Software Engineering](http://cs.gmu.edu/%7Eoffutt/rsrch/web.html)

[Object-oriented Analysis and Testing](http://cs.gmu.edu/%7Eoffutt/rsrch/integ.html)

[Software Maintenance and Open-Source Software](http://cs.gmu.edu/%7Eoffutt/rsrch/maint.html)

[Specification-based (Model-based) Software Testing](http://cs.gmu.edu/%7Eoffutt/rsrch/spec.html)

[Architecture-based and System Testing](http://cs.gmu.edu/%7Eoffutt/rsrch/archtest.html)

[Automatic Test Data Generation](http://cs.gmu.edu/%7Eoffutt/rsrch/atdg.html)

[Mutation Testing](http://cs.gmu.edu/%7Eoffutt/rsrch/mut.html)

[Other Software Testing](http://cs.gmu.edu/%7Eoffutt/rsrch/test.html)

[Software Engineering Education](http://cs.gmu.edu/%7Eoffutt/rsrch/see.html)

[General Software Engineering](http://cs.gmu.edu/%7Eoffutt/rsrch/softwareeng.html)

[My most cited "publication"](http://cs.gmu.edu/%7Eoffutt/hints-net-write.html)

[Software Engineering journals](http://cs.gmu.edu/%7Eoffutt/rsrch/sejournals.html)

[The MCDC web page](http://www.dsl.uow.edu.au/~sergiy/MCDC.html)

|  |
| --- |
| Biosketch |

Jeff Offutt is a full professor of Software Engineering in the Volgenau school of Information Technology at George Mason University. He also holds a part-time visiting faculty position at [University of Skövde](http://www.his.se/english/), Skövde Sweden, where he participates

in the [Distributed Real-Time Systems Research Group (DRTS)](http://www.his.se/research/drts/), contributing expertise on software engineering and software testing. His current research interests include software testing, analysis and testing of web applications, object-oriented program analysis, module and integration testing, formal methods, and software maintenance. He has published over 130 refereed research papers and has received funding from various government agencies and companies. His current projects include testing of web applications, analysis and testing of object-oriented software, measuring software maintenance of open-source software, and deriving tests from formal specifications of safety critical software. He was on the technical board of advisors for Certess, Inc.

He received the Best Teacher Award from the School of Information Technology & Engineering in 2003. His [textbook](http://www.cs.gmu.edu/~offutt/softwaretest/), Introduction to Software Testing (co-authored with [Paul Ammann](http://www.cs.gmu.edu/~pammann/)), was published by Cambridge University Press in January 2008. He leads the MS in Software Engineering program at GMU, teaches MS and PhD courses in Software Engineering and has developed new courses in a variety of Software Engineering subjects, including software testing, construction, design, user interface design, experimentation, and analysis.

Dr. Offutt received a BS degree with a double major in mathematics and data processing from [Morehead State University](http://www.morehead-st.edu), [Morehead](http://en.wikipedia.org/wiki/Morehead,_Kentucky), [Kentucky](http://www.uky.edu/KentuckyAtlas), in 1982, an MS degree in computer science from the [Georgia Institute of Technology](http://www.cc.gatech.edu) in 1985, and a PhD in computer science from the [Georgia Institute of Technology](http://www.cc.gatech.edu) in 1988. From 1988 to 1992, Offutt was an Assistant Professor in the department of [Computer Science at Clemson University.](http://www.cs.clemson.edu)

Offutt is editor-in-chief of Wiley's journal of [Software Testing, Verification and Reliability](http://www3.interscience.wiley.com/cgi-bin/jhome/13635?CRETRY=1&SRETRY=0), is chair of the [steering committee](http://www.cs.gmu.edu/~offutt/icst/) for the International Conference on Software Testing, Verification and Validation [(ICST)](http://www.cs.colostate.edu/icst2008/) and program co-chair for ICST 2009, has served on numerous conference program committees, was program chair for ICECCS 2001, has been on the editorial boards for the [Springer's Empirical Software Engineering Journal](http://www.sce.carleton.ca/squall/emse) (2006-), the Journal of [Software and Systems Modeling](http://www.sosym.org/) (2002-), the [Software Quality Journal](http://www.springerlink.com/(epfsuk5505bk2n55wrztp0qm)/app/home/journal.asp?referrer=parent&backto=linkingpublicationresults,1:100222,1) (2002-), and [IEEE Transactions on Software Engineering](http://www.computer.org/tse/) (2001-2005), is a regular reviewer for NSF and several major research journals, and has been invited to speak throughout the US, Japan, China, and Sweden. He has been involved in a number of software proof-of-concept research systems, including [muJava](http://cs.gmu.edu/~offutt/mujava/), Mothra, Godzilla, CBat, Mistix, Albert, CoupTest, and SpecTest, several of which have been used by many other software engineering researchers. Offutt previously worked on the Software Test and Evaluation Project, for Georgia Tech's Software Engineering Research Center, and helped design and implement the Mothra mutation testing system with [Rich DeMillo](http://www.scs.gatech.edu/people/richard-demillo).

His doctoral research was a method for automatically generating test data to satisfy mutation analysis and included algorithms and an implementation of an automatic test data generator that was integrated with the Mothra system. Largely by using the Mothra system, he invented, developed, and experimentally validated algorithms and engineering techniques that proved that mutation testing can be practical and effective. He has made fundamental contributions to several software testing problems, including mutation, automatic test data generation, object-oriented testing, input space partitioning, specification-based testing, model-based testing, and testing of web applications. He has also published papers on software metrics, maintenance, and software engineering education.

**Masters of Software Engineering**

The Master of Science Degree in Software Engineering provides specialized knowledge and experience in developing and modifying large, complex software systems. It emphasizes technical and management aspects of the software engineering process. Software engineering is an established discipline based on requirement analysis, design, construction, testing, maintenance, economics, and management issues of software engineering. A pragmatic approach to problem solving is the hallmark of a software engineer. Software engineers are concerned with the theoretical and practical aspects of technology, cost, and social impact of software systems that are both effective and efficient.

Software engineers are in demand in every segment of society affected by computing technology. Potential employers include all software vendors and Internet-based companies, electronic business organizations, businesses that build and sell computers, research and development laboratories, aerospace companies, government contractors, banks, insurance companies, and manufacturing organizations. The master’s program is concerned with both technical and managerial issues, but primary emphasis is placed on the technical aspects of building and modifying high quality software systems.

Successful applicants have a broad variety of undergraduate backgrounds, including computer science, science and mathematics, engineering, liberal arts, and business.  Many of our students are working or have worked in the software industry.

The contents of the MS in Software Engineering program are revised on a regular basis to stay abreast of the latest developments in information technology. Recent additions to the program include software construction with the object-oriented Java programming language, requirements analysis with use cases and the Unified Modeling Language

(UML), object-oriented software design with the UML, graphical user interface design, Web applications and services, software project management using the spiral life cycle model and the Capability Maturity Model (CMM), software architecture, reusable software architectures and product lines, design patterns, system testing and testing of object-oriented components, secure software design and programming, and formal methods using the Object Constraint Language (OCL).

All classes are scheduled in the late afternoon and early evening to accommodate employed students.

Changes to the core, effective Fall 2009

Effective Fall 2009, the core [Degree Requirements](http://www.cs.gmu.edu/programs/masters/swe/swe-degree.html) have changed. SWE 637, Software Testing, has been inserted into the core in place of SWE 620, Software Requirements. This change affects students who start in Fall 2009 or after. Previously matriculated students still fall under the old rules, but may change to the new rules by filing an updated Plan of Study.   
[New Degree Requirements](http://www.cs.gmu.edu/programs/masters/swe/swe-degree.html)   
[Old Degree Requirements](http://www.cs.gmu.edu/programs/masters/swe/swe-degree-2008.html)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | Masters of Software Engineering Degree Requirements  (These requirements are effective Fall 2009: [Pre Fall 2009 Degree Requirements](http://www.cs.gmu.edu/programs/masters/swe/swe-degree-2008.html) are effective for students who matriculated before Fall 2009)  [Software Architecture](http://catalog.gmu.edu/preview_program.php?catoid=15&poid=6189&bc=1)  In addition to the general requirements of the university, the MS in Software Engineering requires a minimum of 30 graduate credits. The coursework is divided into three categories: a breadth requirement of 12 hours of core courses, a depth requirement of 9 hours of emphasis courses, and 9 hours of elective courses.  Four core courses (12 credits) are required of all SWE graduates:  [SWE 619](http://www.cs.gmu.edu/programs/masters/swe/swe-course.html#swe619)   Object-Oriented Software Specification and Construction  [SWE 621](http://www.cs.gmu.edu/programs/masters/swe/swe-course.html#swe621)   Software Modeling and Architectural Design  [SWE 622](http://www.cs.gmu.edu/programs/masters/swe/swe-course.html#swe622)   Distributed Software Engineering  [SWE 637](http://www.cs.gmu.edu/programs/masters/swe/swe-course.html#swe637)   Software Testing  Emphasis courses (9 credits). Students may choose an emphasis by taking three courses from one of the emphases areas of software design, software assurance, software management, and web applications. Or, students may mix and match from different areas or take three courses from the [elective list](http://www.cs.gmu.edu/programs/masters/swe/swe-electives.html).  Software Design  [SWE 626](http://www.cs.gmu.edu/programs/masters/swe/swe-course.html#swe626)   Software Project Laboratory  [SWE 632](http://www.cs.gmu.edu/programs/masters/swe/swe-course.html#swe632)   User Interface Design and Development  [SWE 721](http://www.cs.gmu.edu/programs/masters/swe/swe-course.html#swe721)   Reusable Software Architectures  [SWE 727](http://www.cs.gmu.edu/programs/masters/swe/swe-course.html#swe727)   Quality of Service for Software Architectures  [SWE 781](http://www.cs.gmu.edu/programs/masters/swe/swe-course.html#swe781)   Secure Software Design and Programming  Software Assurance  [ISA 562](http://www.cs.gmu.edu/programs/masters/isa/isa-course.html#isa562)   Information Security Theory and Practice  [SWE 623](http://www.cs.gmu.edu/programs/masters/swe/swe-course.html#swe623)   Formal Methods and Models in Software Engineering  [SWE 723](http://www.cs.gmu.edu/programs/masters/swe/swe-course.html#swe723)   Precise Modeling  [SWE 781](http://www.cs.gmu.edu/programs/masters/swe/swe-course.html#swe781)   Secure Software Design and Programming  Software Management  [SWE 620](http://www.cs.gmu.edu/programs/masters/swe/swe-course.html#swe620)   Software Requirements Analysis and Specification  [SWE 625](http://www.cs.gmu.edu/programs/masters/swe/swe-course.html#swe625)   Software Project Management  [SWE 626](http://www.cs.gmu.edu/programs/masters/swe/swe-course.html#swe626)   Software Project Laboratory  [SWE 630](http://www.cs.gmu.edu/programs/masters/swe/swe-course.html#swe630)   Software Engineering Economics  Web Applications  [INFS 614](http://www.cs.gmu.edu/programs/masters/infs/infs-course.html#infs614)   Database Management  [SWE 632](http://www.cs.gmu.edu/programs/masters/swe/swe-course.html#swe632)   User Interface Design and Development  [SWE 642](http://www.cs.gmu.edu/programs/masters/swe/swe-course.html#swe642)   Software Engineering for the World Wide Web  [SWE 645](http://www.cs.gmu.edu/programs/masters/swe/swe-course.html#swe645)   Component-Based Software Development  Elective courses (9 credits).  Students may select the remaining courses from the list of [approved elective courses](http://www.cs.gmu.edu/programs/masters/swe/swe-electives.html), including other emphasis areas and courses from other MS programs in the department and the Volgenau school. Students may choose other graduate electives with the consent of their faculty advisor and the graduate coordinator.  Students, with the consent of a faculty sponsor and faculty advisor, may also elect courses in individualized study, special topics, or a 6-credit thesis, which is primarily intended for students planning to pursue a PhD in Information Technology with a concentration in software engineering. The course designations are:  [SWE 699](http://www.cs.gmu.edu/programs/masters/swe/swe-course.html#swe699)   Special Topics in Software Engineering Credits  [SWE 795](http://www.cs.gmu.edu/programs/masters/swe/swe-course.html#sw785)   Advanced Topics in Software Engineering Credits  [SWE 796](http://www.cs.gmu.edu/programs/masters/swe/swe-course.html#swe796)   Directed Readings in Software Engineering Credits  [SWE 798](http://www.cs.gmu.edu/programs/masters/swe/swe-course.html#swe798)   Research Project Credits  [SWE 799](http://www.cs.gmu.edu/programs/masters/swe/swe-course.html#swe799)   Thesis Credits | | | |
|  | Masters of Software Engineering Course Descriptions  SWE course descriptions are also listed in the [University Catalog](http://catalog.gmu.edu/preview_course_incoming.php?cattype=combined&prefix=swe)  About the Program  [MSSWE Home](http://www.cs.gmu.edu/programs/masters/swe/index.html)46  [Admission Requirements](http://www.cs.gmu.edu/programs/masters/swe/swe-admit.html)  [Foundation Requirements](http://www.cs.gmu.edu/programs/masters/swe/swe-found.html)  [Testing Out of Foundations](http://www.cs.gmu.edu/exams/testout)  [Degree Requirements](http://www.cs.gmu.edu/programs/masters/swe/swe-degree.html)  [SWE Course Descriptions](http://www.cs.gmu.edu/programs/masters/swe/swe-course.html)  [Pre-approved Electives](http://www.cs.gmu.edu/programs/masters/swe/swe-electives.html)  [Advising](http://www.cs.gmu.edu/programs/masters/swe/swe-adv.html)  [Plan of Study Form](http://www.cs.gmu.edu/programs/masters/swe/msswe-plan.pdf)  [Plan of Study (pre Fall 2009)](http://www.cs.gmu.edu/programs/masters/swe/msswe-plan-preF2009.pdf)  [Self Evaluation Form](http://ite.gmu.edu/forms/MS-NDISYSISAandSWESelfEvalForm.doc) (MS-word)  Accelerated BS-MSSWE  [For GMU CS students](http://www.cs.gmu.edu/programs/masters/swe/bscs-swe.html)  [For GMU IT students](http://www.cs.gmu.edu/programs/masters/swe/bsit-swe.html)  Graduate Certificates  [SWE](http://catalog.gmu.edu/preview_program.php?catoid=15&poid=6190&bc=1)  [Web-Based SWE](http://catalog.gmu.edu/preview_program.php?catoid=15&poid=6235&bc=1)  [Software Architecture](http://catalog.gmu.edu/preview_program.php?catoid=15&poid=6189&bc=1)  **Undergraduate Courses**   [Go to grad courses](http://www.cs.gmu.edu/programs/masters/swe/swe-course.html#swe619)   |  | | --- | | SWE 205 - Software Usability Analysis and Design (3:3:0). Prerequisite: None. Principles of user interface design. Concepts for objectively and quantitatively assessing the usability of software user interfaces. Outcomes include knowledge of quantitative engineering principles for designing usable software interfaces and an understanding that usability is more important than efficiency for almost all modern software projects, and often the primary factor that leads to product success. Major topics include cognitive models for human perceptions and needs, which are used as a basis for analytical and critical thinking about user interfaces; specific engineering principles for designing usable menus, forms, command languages, web sites, graphical user interfaces and web-based user interfaces. Assessments will include written analytical evaluations of existing user interfaces, exams, and HTML-based design projects. | | SWE 301 - **Internship Preparation** (0:0:0). Prerequisite: Limited to ACS or CS majors with junior standing or permission of instructor. Preparation for Internship Educational Experience. Intended for, but not limited to, students planning internships in the Applied Computer Science Software Engineering Program. Internship employment opportunities. Basic interview skills. Techniques for applying academic knowledge to practical software development. Techniques for extracting knowledge from practical experience. Peer presentation from students who have completed internships. | | SWE 332 **Object-Oriented Software Design and Implementation** (3:3:0). Prerequisite: CS 211. In-depth study of software design and implementation using a modern, object-oriented language with support for graphical user interfaces and complex data structures. Topics cover specifications, design patterns, and abstraction techniques, including typing, access control, inheritance, and polymorphism. Students will learn the proper engineering use of techniques such as information hiding, classes, objects, inheritance, exception handling, event-based systems, and concurrency. | | SWE 401 - **Internship Reflection** (1:1:0). Prerequisite: SWE 301 and completion of internship. Reflection on Internship Educational Experience. Intended for, but not limited to, students completing internships in the Applied Computer Science Software Engineering Program. Analysis of techniques for applying academic knowledge to practical software development. Analysis of techniques for extracting knowledge from practical experience. Student presentations summarizing internships relating them to academic program goals. | | SWE 421 **Software Requirements and Design Modeling** (3:3:0). Prerequisite: CS 211. An introduction to concepts, methods, and tools for the creation of large-scale software systems. Methods, tools, notations, and validation techniques to analyze, specify, prototype, and maintain software requirements. Introduction to object-oriented requirements modeling, including use case modeling, static modeling, and dynamic modeling using the Unified Modeling Language (UML) notation. Concepts and methods for the design of large-scale software systems. Fundamental design concepts and design modeling using UML notation. Students participage in a group project on software requirements, specification, and object-oriented software design. | | SWE 432 **Design and Implementation of Software for the Web** (3:3:0). Prerequisite: Math 125 and CS 421. This course teaches students how to develop software for web applications. The concepts of client-server computing, theories of usable graphical user interfaces, and models for web-based information retrieval and processing are covered. Goals are to understand how to design usable software interfaces and implement them on the web, learn how to build software that accepts information from users across the web and returns data to the user, and understand how to interact with database engines to store and retrieve information. Specific topics that are included are HTML, CGI programming, Java, Java applets, Javascripts, and Java **servlets**. | | SWE 437 **Software Testing and Maintenance** (3:3:0). Prerequisite: CS 211 and Math 125. Concepts and techniques for testing and modifying software in evolving environments. Topics include software testing at the unit, module, subsystem, and system levels; developer testing; automatic and manual techniques for generating test data; testing concurrent and distributed software; designing and implementing software to increase maintainability and reuse; evaluating software for change; and validating software changes. | | SWE 443 **Software Architectures** (3:3:0). Prerequisite: SWE 421. This course teaches how to design, understand, and evaluate software systems at an architectural level of abstraction. By the end of the course, students will be able to recognize major architectural styles in existing software systems, describe a system's architecture accurately, generate architectural alternatives to address a problem and choose from among them, design a medium-sized software system that satisfies a specification of requirements, use existing tools to expedite software design, and evaluate the suitability of a given architecture in meeting a set of system requirments. |   **Foundation Courses**   |  | | --- | | SWE 510 **Object-Oriented Programming in Java** (3:3:0). Prerequisite: Undergraduate courses or equivalent knowledge in programming in a high-level language. This course introduces students to programming in the Java language. Topics include problem-solving methods and algorithm development, program structures, abstract data types, simple data and file structures, and program development in a modular, object-oriented manner. Introductory use of OO language features, including data hiding, inheritance, polymorphism, and exception handling. Goals include design and development of Java classes and class type hierarchies. An introduction to Java servlets and applets is included. Emphasis on program development is reinforced through several programming projects. Credit cannot be applied to any graduate degree in Volgenau School or the BS degree in computer science. |   **Graduate Courses**   |  | | --- | | SWE 619 **Object-Oriented Software Specification and Construction** (3:3:0). Prerequisites: SWE foundation courses or equivalent. An in-depth study of software construction using a modern, object-oriented language with support for graphical user interfaces and complex data structures. Specifications, design patterns, and abstraction techniques, including procedural, data, iteration, type, and polymorphic. Information hiding, classes, objects, and inheritance. Exception handling, event-based systems, and concurrency. | | SWE 620 **Software Requirements Analysis and Specification** (3:3:0). Prerequisites: SWE foundation courses or equivalent. An in-depth study of methods, tools, notations, and validation techniques for the analysis, specification, prototyping, and maintenance of software requirements. In-depth study of object-oriented requirements modeling, including use case modeling, static modeling and dynamic modeling using the Unified Modeling Language (UML) notation. Students participate in a group project on software requirements and specification using a modern method. | | SWE 621 **Software Modeling and Architectural Design** (3:3:0). Prerequisites: SWE 619, with 620 recommended, or permission of instructor. (MSCS students may substitute CS 540 and CS 571 for SWE 619). Concepts and methods for the architectural design of large-scale software systems. Fundamental design concepts and design notations are introduced. Several design methods are presented and compared. In-depth study of object-oriented analysis and design modeling using the Unified Modeling Language (UML) notation. Students participate in a group project on object-oriented software design. | | SWE 622 **Distributed Software Engineering** (3:3:0). Prerequisites: SWE foundation courses or equivalent. Hands-on introduction to techniques and programming interfaces for distributed software engineering. Networking protocols at several layers. Construction of distributed and concurrent software using network protocol services. Applications of Internet and Web-based software. | | SWE 623 **Formal Methods and Models in Software Engineering** (3:3:0). Prerequisites: SWE 619 or permission of instructor. Formal mechanisms for specifying, validating, and verifying software systems. Program verification through Hoare's method and Dijkstra's weakest preconditions. Formal specification via algebraic specifications and abstract model specifications, including initial specification and refinement towards implementation. Integration of formal methods with existing programming languages, and the application of formal methods to requirements analysis, testing, safety analysis, and object-oriented approaches. Formal methods using the Object Constraint Language (OCL). | | SWE 625 **Software Project Management** (3:3:0). Prerequisites: SWE foundation courses or equivalent. Lifecycle and process models; process metrics; planning for a software project; mechanisms for monitoring and controlling schedule, budget, quality, and productivity; and leadership, motivation, and team building. | | SWE 626 **Software Project Laboratory** (3:3:6). Prerequisites: SWE 619, 620, and 621; or permission of instructor. Covers requirements analysis, design, implementation, and management of software development project. Students work in teams to develop or modify software product, applying sound principles of software engineering. Uses both industrial, academic standards to assess quality of work products. | | SWE 630 **Software Engineering Economics** (3:3:0). Prerequisite: SWE 625. Covers quantitative models of the software lifecycle, cost-effectiveness analysis in software engineering, multiple-goal decision analysis, uncertainty and risk analysis, software cost estimation, software engineering metrics; and quantitative lifecycle management techniques. | | SWE 631/CS 631 **Object-Oriented Design Patterns** (3:3:0). Prerequisite: SWE 619 or 620 or CS 540 or 571 or a graduate course in object-oriented programming or equivalent. Principles of object-oriented design through design patterns. A study of the selection of appropriate object-oriented structure after the system requirements or requirements specification of the software system have been developed. Design patterns are created in the logic view of the software system. A study of generalized design solutions for generalized software design problems. A study of the reuse of design patterns. Once developed, design patterns may be specified in any object-oriented language. | | SWE 632 **User Interface Design and Development** (3:3:0). Prerequisite: SWE 619, or CS 540 and 571, or permission of instructor. Principles of user interface design, development, and programming. Includes user psychology and cognitive science, menu system design, command language design, icon and window design, graphical user interfaces and web-based user interfaces. | | SWE 637 **Software Testing** (3:3:0). Prerequisite: SWE 619 or permission of instructor. Concepts and techniques for testing software and assuring its quality. Topics cover software testing at the unit, module, subsystem, and system levels; automatic and manual techniques for generating and validating test data; the testing process; static vs. dynamic analysis; functional testing; inspections; and reliability assessment. | | SWE 641/SYST 621 **Systems Architecture for Large-Scale Systems** (3:3:0). Prerequisite: SYST 510 or equivalent. Introduction to system architecture for the technical description of large-scale systems. An intensive study of the relationships between the different types of architecture representations and the methodologies used to obtain them. Systems engineering approaches for transitioning from functional descriptions to structure and architectural descriptions. Analysis of existing architectures and design of new architectures. The role of modeling, prototyping, and simulation in architecture development. Executable models of system architectures and performance evaluation. The role of the systems architect, the systems architecting process, and systems management of architecture and design activities. System interoperability, integration, and interfaces. A case study of a large-scale system conceptual architecture will be used to demonstrate application of systems architecture principles. | | SWE 642 **Software Engineering for the World Wide Web** (3:3:0). Prerequisites: SWE 619, or CS 540 and 571, or permission of instructor. Detailed study of engineering methods and technologies for building highly interactive web sites for e-commerce and other web-based applications. Presents engineering principles for building web sites that exhibit high reliability, usability, security, availability, scalability, and maintainability. Teaches methods such as client-server programming, component-based software development, middleware, and reusable components. | | SWE 645 **Component-Based Software Development** (3:3:0). Prerequisite: SWE 619, or CS 540 and CS 571 or permission of instructor. Introduction to the concepts and foundations of software component and component-based software. Detailed study of the engineering principles of modeling, designing, implementing, testing, and deploying component-based software. State-of-the-art component technologies will also be explored. | | SWE 699 **Special Topics in Software Engineering** (3:3:0). Prerequisite: Permission of instructor. Special topics not occurring in the regular SWE sequence. May be repeated for credit when semester topic is different. | | SWE 720 **Advanced Software Requirements** (3:3:0). Prerequisites: SWE 620 and 621. The course gives state-of-the-art and state-of-the-practice in software requirements engineering. In-depth coverage of selected methods, tools, notations, or validation techniques for the analysis and specification of software requirements. The course work includes a project investigating or applying approaches to requirements engineering. | | SWE 721 **Reusable Software Architectures** (3:3:0). Prerequisites: SWE 620 and 621. This course investigates the software concepts that promote reuse of software architectures. The influence of object technology on software design and reuse is studied. Domain Modeling methods, which model the application domain as a software product family from which target systems can be configured, are investigated. The course also covers reusable software patterns including architecture patterns and design patterns, software components, and object-oriented frameworks. | | SWE 723 **Precise Modeling** (3:3:0). Prerequisite: SWE 621. This course discusses ongoing advances in modeling techniques for software design, including but not limited to introducing precision, performance, security and safety aspects. UML, its meta-models and proposed enhancements such as Object Security Constraint Language, Object Temporal Constraint Language, QoS Profiles and the theory behind them and their implementations will be discussed. | | SWE 727 - **Quality of Service for Software Architectures** (3:3:0). Prerequisite: SWE 621 or permission of instructor. Builds on acquired skills for modeling architectures, and focuses on the relationship between architectural patterns and qualities of service (QoS). By the end of the course, students will be able to elicit the QoS preferences of stakeholder; recognize major architectural styles and the QoS tradeoffs that each presents; design for and reconcile competing QoS requirements; and evaluate a given architecture with respect to a set of QoS requirements. | | SWE 763 **Software Engineering Experimentation** (3:3:0). Prerequisite: SWE 621 or permission of instructor. A detailed study of the scientific process, particularly using the experimental method. The course examines how empirical studies are carried out in software engineering. The distinction between analytical techniques and empirical techniques is reviewed. Other topics include experimentation required in software engineering, kinds of problems that can be solved using experimentation, methods used to control variables and eliminate bias in experimentation, and analysis and presentation of empirical data for decision making. | | SWE 781 **Secure Software Design and Programming** (3:3:0). Prerequisites: SWE 619 or permission of instructor. Theory and practice of software security, focusing in particular on some common software security risks, including buffer overflows, race conditions and random number generation, and on the identification of potential threats and vulnerabilities early in the design cycle. The emphasis is on methodologies and tools for identifying and eliminating security vulnerabilities, techniques to prove the absence of vulnerabilities, and ways to avoid security holes in new software and on essential guidelines for building secure software: how to design software with security in mind from the ground up and to integrate analysis and risk management throughout the software life cycle. | | SWE 795 - **Advanced Topics in Software Engineering** (3:3:0). Prerequisite: 12 credits applicable toward MS program. Advanced topics not occurring in existing courses. Topics normally assume knowledge in one or more existing MS SWE courses.  Note: Repeatable within degree for credit when subject differs. | | SWE 796 **Directed Readings in Software Engineering** (3:3:0). Prerequisite: Permission of instructor. Analysis and investigation of a contemporary problem in software engineering. Prior approval by a faculty member who supervises the student's work is required. A written report is also required. A maximum of 6 hours may be earned. (In order to register, the student must complete an independent study form, which is available in the department office. The form must be initialed by the faculty sponsor and approved by the department chairman.) | | SWE 798 **Research Project** (3:3:0). Prerequisite: 18 credits applicable toward MS. Master's degree candidates undertake a project using knowledge gained in MS program. Topics chosen in consultation with a faculty sponsor. Prior approval required by faculty sponsor who supervises student's work. Research projectg is chosen under guidance of full-time graduate faculty member, resulting in written technical report. (In order to register, the student must complete an independent study form, which is available in the department office. The form must be initialed by the faculty sponsor and approved by the department chairman.) | | SWE 799 **Thesis** (6:0:0). Prerequisite: Permission of advisor. A research project completed under the supervision of a faculty member, which results in a technical report accepted by a three-member faculty committee. The report must be defended in an oral presentation. (In order to register, the student must complete an independent study form, which is available in the department office. The form must be initialed by the faculty sponsor and approved by the department chairman.) | | SWE 825/IT 825 **Special Topics in Web-Based Software** (3:3:0). Prerequisite: SWE 642. Advanced topics in specifying, designing, modeling, developing, deplyoing, testing, and maintaining software written as web applications and web services. May be repeated with change in topic. | | | | | |
|  |  | | | | |
|  | Masters of Software Engineering Course Descriptions  SWE course descriptions are also listed in the [University Catalog](http://catalog.gmu.edu/preview_course_incoming.php?cattype=combined&prefix=swe)  About the Program  [MSSWE Home](http://www.cs.gmu.edu/programs/masters/swe/index.html)  [Software Architecture](http://catalog.gmu.edu/preview_program.php?catoid=15&poid=6189&bc=1)  [Admission Requirements](http://www.cs.gmu.edu/programs/masters/swe/swe-admit.html)  [Foundation Requirements](http://www.cs.gmu.edu/programs/masters/swe/swe-found.html)  [Testing Out of Foundations](http://www.cs.gmu.edu/exams/testout)  [Degree Requirements](http://www.cs.gmu.edu/programs/masters/swe/swe-degree.html)  [SWE Course Descriptions](http://www.cs.gmu.edu/programs/masters/swe/swe-course.html)  [Pre-approved Electives](http://www.cs.gmu.edu/programs/masters/swe/swe-electives.html)  [Advising](http://www.cs.gmu.edu/programs/masters/swe/swe-adv.html)  [Plan of Study Form](http://www.cs.gmu.edu/programs/masters/swe/msswe-plan.pdf)  [Plan of Study (pre Fall 2009)](http://www.cs.gmu.edu/programs/masters/swe/msswe-plan-preF2009.pdf)  [Self Evaluation Form](http://ite.gmu.edu/forms/MS-NDISYSISAandSWESelfEvalForm.doc) (MS-word)  Accelerated BS-MSSWE  [For GMU CS students](http://www.cs.gmu.edu/programs/masters/swe/bscs-swe.html)  [For GMU IT students](http://www.cs.gmu.edu/programs/masters/swe/bsit-swe.html)  Graduate Certificates  [SWE](http://catalog.gmu.edu/preview_program.php?catoid=15&poid=6190&bc=1)  [Web-Based SWE](http://catalog.gmu.edu/preview_program.php?catoid=15&poid=6235&bc=1)  **Undergraduate Courses**   [Go to grad courses](http://www.cs.gmu.edu/programs/masters/swe/swe-course.html#swe619)   |  | | --- | | SWE 205 - Software Usability Analysis and Design (3:3:0). Prerequisite: None. Principles of user interface design. Concepts for objectively and quantitatively assessing the usability of software user interfaces. Outcomes include knowledge of quantitative engineering principles for designing usable software interfaces and an understanding that usability is more important than efficiency for almost all modern software projects, and often the primary factor that leads to product success. Major topics include cognitive models for human perceptions and needs, which are used as a basis for analytical and critical thinking about user interfaces; specific engineering principles for designing usable menus, forms, command languages, web sites, graphical user interfaces and web-based user interfaces. Assessments will include written analytical evaluations of existing user interfaces, exams, and HTML-based design projects. | | SWE 301 - **Internship Preparation** (0:0:0). Prerequisite: Limited to ACS or CS majors with junior standing or permission of instructor. Preparation for Internship Educational Experience. Intended for, but not limited to, students planning internships in the Applied Computer Science Software Engineering Program. Internship employment opportunities.  **Basic** interview skills. Techniques for applying academic knowledge to practical software development. Techniques for extracting knowledge from practical experience. Peer presentation from students who have completed internships. | | SWE 332 **Object-Oriented Software Design and Implementation** (3:3:0). Prerequisite: CS 211. In-depth study of software design and implementation using a modern, object-oriented language with support for graphical user interfaces and complex data structures. Topics cover specifications, design patterns, and abstraction techniques, including typing, access control, inheritance, and polymorphism. Students will learn the proper engineering use of techniques such as information hiding, classes, objects, inheritance, exception handling, event-based systems, and concurrency. | | | SWE 401 - **Internship Reflection** (1:1:0). Prerequisite: SWE 301 and completion of internship. Reflection on Internship Educational Experience. Intended for, but not limited to, students completing internships in the Applied Computer Science Software Engineering Program. Analysis of techniques for applying academic knowledge to practical software development. Analysis of techniques for extracting knowledge from practical experience. Student presentations summarizing internships relating them to academic program goals. | | | SWE 421 **Software Requirements and Design Modeling** (3:3:0). Prerequisite: CS 211. An introduction to concepts, methods, and tools for the creation of large-scale software systems. Methods, tools, notations, and validation techniques to analyze, specify, prototype, and maintain software requirements. Introduction to object-oriented requirements modeling, including use case modeling, static modeling, and dynamic modeling using the Unified Modeling Language (UML) notation. Concepts and methods for the design of large-scale software systems. Fundamental design concepts and design modeling using UML notation. Students participage in a group project on software requirements, specification, and object-oriented software design.  **Doesn’t move anywhere** | | SWE 432 **Design and Implementation of Software for the Web** (3:3:0). Prerequisite: Math 125 and CS 421. This course teaches students how to develop software for web applications. The concepts of client-server computing, theories of usable graphical user interfaces, and models for web-based information retrieval and processing are covered. Goals are to understand how to design usable software interfaces and implement them on the web, learn how to build software that accepts information from users across the web and returns data to the user, and understand how to interact with database engines to store and retrieve information. Specific topics that are included are HTML, CGI programming, Java, Java applets, Javascripts, and Java servlets. | | SWE 437 **Software Testing and Maintenance** (3:3:0). Prerequisite: CS 211 and Math 125. Concepts and techniques for testing and modifying software in evolving environments. Topics include software testing at the unit, module, subsystem, and system levels; developer testing; automatic and manual techniques for generating test data; testing concurrent and distributed software; designing and implementing software to increase maintainability and reuse; evaluating software for change; and validating software changes. | | SWE 443 **Software Architectures** (3:3:0). Prerequisite: SWE 421. This course teaches how to design, understand, and evaluate software systems at an architectural level of abstraction. By the end of the course, students will be able to recognize major architectural styles in existing software systems, describe a system's architecture accurately, generate architectural alternatives to address a problem and choose from among them, design a medium-sized software system that satisfies a specification of requirements, use existing tools to expedite software design, and evaluate the suitability of a given architecture in meeting a set of system requirments. |   **Foundation Courses**  SWE 510 **Object-Oriented Programming in Java** (3:3:0). Prerequisite: Undergraduate courses or equivalent knowledge in programming in a high-level language. This course introduces students to programming in the Java language. Topics include problem-solving methods and algorithm development, program structures, abstract data types, simple data and file structures, and program development in a modular, object-oriented manner. Introductory use of OO language features, including data hiding, inheritance, polymorphism, and exception handling. Goals include design and development of Java classes and class type hierarchies. An introduction to Java servlets and applets is included. Emphasis on program development is reinforced through several programming projects. Credit cannot be applied to any graduate degree in Volgenau School or the BS degree in computer science.  **Graduate Courses**   |  | | --- | | SWE 619 **Object-Oriented Software Specification and Construction** (3:3:0). Prerequisites: SWE foundation courses or equivalent. An in-depth study of software construction using a modern, object-oriented language with support for graphical user interfaces and complex data structures. Specifications, design patterns, and abstraction techniques, including procedural, data, iteration, type, and polymorphic. Information hiding, classes, objects, and inheritance. Exception handling, event-based systems, and concurrency. | | SWE 620 **Software Requirements Analysis and Specification** (3:3:0). Prerequisites: SWE foundation courses or equivalent. An in-depth study of methods, tools, notations, and validation techniques for the analysis, specification, prototyping, and maintenance of software requirements. In-depth study of object-oriented requirements modeling, including use case modeling, static modeling and dynamic modeling using the Unified Modeling Language (UML) notation. Students participate in a group project on software requirements and specification using a modern method. | | SWE 621 **Software Modeling and Architectural Design** (3:3:0). Prerequisites: SWE 619, with 620 recommended, or permission of instructor. (MSCS students may substitute CS 540 and CS 571 for SWE 619). Concepts and methods for the architectural design of large-scale software systems. Fundamental design concepts and design notations are introduced. Several design methods are presented and compared. In-depth study of object-oriented analysis and design modeling using the Unified Modeling Language (UML) notation. Students participate in a group project on object-oriented software design. | | SWE 622 **Distributed Software Engineering** (3:3:0). Prerequisites: SWE foundation courses or equivalent. Hands-on introduction to techniques and programming interfaces for distributed software engineering. Networking protocols at several layers. Construction of distributed and concurrent software using network protocol services. Applications of Internet and Web-based software. | | SWE 623 **Formal Methods and Models in Software Engineering** (3:3:0). Prerequisites: SWE 619 or permission of instructor. Formal mechanisms for specifying, validating, and verifying software systems. Program verification through Hoare's method and Dijkstra's weakest preconditions. Formal specification via algebraic specifications and abstract model specifications, including initial specification and refinement towards implementation. Integration of formal methods with existing programming languages, and the application of formal methods to requirements analysis, testing, safety analysis, and object-oriented approaches. Formal methods using the Object Constraint Language (OCL). | | SWE 625 **Software Project Management** (3:3:0). Prerequisites: SWE foundation courses or equivalent. Lifecycle and process models; process metrics; planning for a software project; mechanisms for monitoring and controlling schedule, budget, quality, and productivity; and leadership, motivation, and team building. | | SWE 626 **Software Project Laboratory** (3:3:6). Prerequisites: SWE 619, 620, and 621; or permission of instructor. Covers requirements analysis, design, implementation, and management of software development project. Students work in teams to develop or modify software product, applying sound principles of software engineering. Uses both industrial, academic standards to assess quality of work products. | | SWE 630 **Software Engineering Economics** (3:3:0). Prerequisite: SWE 625. Covers quantitative models of the software lifecycle, cost-effectiveness analysis in software engineering, multiple-goal decision analysis, uncertainty and risk analysis, software cost estimation, software engineering metrics; and quantitative lifecycle management techniques. | | SWE 631/CS 631 **Object-Oriented Design Patterns** (3:3:0). Prerequisite: SWE 619 or 620 or CS 540 or 571 or a graduate course in object-oriented programming or equivalent. Principles of object-oriented design through design patterns. A study of the selection of appropriate object-oriented structure after the system requirements or requirements specification of the software system have been developed. Design patterns are created in the logic view of the software system. A study of generalized design solutions for generalized software design problems. A study of the reuse of design patterns. Once developed, design patterns may be specified in any object-oriented language. | | SWE 632 **User Interface Design and Development** (3:3:0). Prerequisite: SWE 619, or CS 540 and 571, or permission of instructor. Principles of user interface design, development, and programming. Includes user psychology and cognitive science, menu system design, command language design, icon and window design, graphical user interfaces and web-based user interfaces. | | SWE 637 **Software Testing** (3:3:0). Prerequisite: SWE 619 or permission of instructor. Concepts and techniques for testing software and assuring its quality. Topics cover software testing at the unit, module, subsystem, and system levels; automatic and manual techniques for generating and validating test data; the testing process; static vs. dynamic analysis; functional testing; inspections; and reliability assessment. | | SWE 641/SYST 621 **Systems Architecture for Large-Scale Systems** (3:3:0). Prerequisite: SYST 510 or equivalent. Introduction to system architecture for the technical description of large-scale systems. An intensive study of the relationships between the different types of architecture representations and the methodologies used to obtain them. Systems engineering approaches for transitioning from functional descriptions to structure and architectural descriptions. Analysis of existing architectures and design of new architectures. The role of modeling, prototyping, and simulation in architecture development. Executable models of system architectures and performance evaluation. The role of the systems architect, the systems architecting process, and systems management of architecture and design activities. System interoperability, integration, and interfaces. A case study of a large-scale system conceptual architecture will be used to demonstrate application of systems architecture principles. | | SWE 642 **Software Engineering for the World Wide Web** (3:3:0). Prerequisites: SWE 619, or CS 540 and 571, or permission of instructor. Detailed study of engineering methods and technologies for building highly interactive web sites for e-commerce and other web-based applications. Presents engineering principles for building web sites that exhibit high reliability, usability, security, availability, scalability, and maintainability. Teaches methods such as client-server programming, component-based software development, middleware, and reusable components. | | SWE 645 **Component-Based Software Development** (3:3:0). Prerequisite: SWE 619, or CS 540 and CS 571 or permission of instructor. Introduction to the concepts and foundations of software component and component-based software. Detailed study of the engineering principles of modeling, designing, implementing, testing, and deploying component-based software. State-of-the-art component technologies will also be explored. | | SWE 699 **Special Topics in Software Engineering** (3:3:0). Prerequisite: Permission of instructor. Special topics not occurring in the regular SWE sequence. May be repeated for credit when semester topic is different. | | SWE 720 **Advanced Software Requirements** (3:3:0). Prerequisites: SWE 620 and 621. The course gives state-of-the-art and state-of-the-practice in software requirements engineering. In-depth coverage of selected methods, tools, notations, or validation techniques for the analysis and specification of software requirements. The course work includes a project investigating or applying approaches to requirements engineering. | | SWE 721 **Reusable Software Architectures** (3:3:0). Prerequisites: SWE 620 and 621. This course investigates the software concepts that promote reuse of software architectures. The influence of object technology on software design and reuse is studied. Domain Modeling methods, which model the application domain as a software product family from which target systems can be configured, are investigated. The course also covers reusable software patterns including architecture patterns and design patterns, software components, and object-oriented frameworks. | | SWE 723 **Precise Modeling** (3:3:0). Prerequisite: SWE 621. This course discusses ongoing advances in modeling techniques for software design, including but not limited to introducing precision, performance, security and safety aspects. UML, its meta-models and proposed enhancements such as Object Security Constraint Language, Object Temporal Constraint Language, QoS Profiles and the theory behind them and their implementations will be discussed. | | SWE 727 - **Quality of Service for Software Architectures** (3:3:0). Prerequisite: SWE 621 or permission of instructor. Builds on acquired skills for modeling architectures, and focuses on the relationship between architectural patterns and qualities of service (QoS). By the end of the course, students will be able to elicit the QoS preferences of stakeholder; recognize major architectural styles and the QoS tradeoffs that each presents; design for and reconcile competing QoS requirements; and evaluate a given architecture with respect to a set of QoS requirements. | | SWE 763 **Software Engineering Experimentation** (3:3:0). Prerequisite: SWE 621 or permission of instructor. A detailed study of the scientific process, particularly using the experimental method. The course examines how empirical studies are carried out in software engineering. The distinction between analytical techniques and empirical techniques is reviewed. Other topics include experimentation required in software engineering, kinds of problems that can be solved using experimentation, methods used to control variables and eliminate bias in experimentation, and analysis and presentation of empirical data for **decision making.** | | SWE 781 **Secure Software Design and Programming** (3:3:0). Prerequisites: SWE 619 or permission of instructor. Theory and practice of software security, focusing in particular on some common software security risks, including buffer overflows, race conditions and random number generation, and on the identification of potential threats and vulnerabilities early in the design cycle. The emphasis is on methodologies and tools for identifying and eliminating security vulnerabilities, techniques to prove the absence of vulnerabilities, and ways to avoid security holes in new software and on essential guidelines for building secure software: how to design software with security in mind from the ground up and to integrate analysis and risk management throughout the software life cycle. | | SWE 795 - **Advanced Topics in Software Engineering** (3:3:0). Prerequisite: 12 credits applicable toward MS program. Advanced topics not occurring in existing courses. Topics normally assume knowledge in one or more existing MS SWE courses.  Note: Repeatable within degree for credit when subject differs. | | SWE 796 **Directed Readings in Software Engineering** (3:3:0). Prerequisite: Permission of instructor. Analysis and investigation of a contemporary problem in software engineering. Prior approval by a faculty member who supervises the student's work is required. A written report is also required. A maximum of 6 hours may be earned. (In order to register, the student must complete an independent study form, which is available in the department office. The form must be initialed by the faculty sponsor and approved by the department chairman.) | | SWE 798 **Research Project** (3:3:0). Prerequisite: 18 credits applicable toward MS. Master's degree candidates undertake a project using knowledge gained in MS program. Topics chosen in consultation with a faculty sponsor. Prior approval required by faculty sponsor who supervises student's work. Research projectg is chosen under guidance of full-time graduate faculty member, resulting in written technical report. (In order to register, the student must complete an independent study form, which is available in the department office. The form must be initialed by the faculty sponsor and approved by the department chairman.) | | SWE 799 **Thesis** (6:0:0). Prerequisite: Permission of advisor. A research project completed under the supervision of a faculty member, which results in a technical report accepted by a three-member faculty committee. The report must be defended in an oral presentation. (In order to register, the student must complete an independent study form, which is available in the department office. The form must be initialed by the faculty sponsor and approved by the department chairman.) | | SWE 825/IT 825 **Special Topics in Web-Based Software** (3:3:0). Prerequisite: SWE 642. Advanced topics in specifying, designing, modeling, developing, deplyoing, testing, and maintaining software written as web applications and web services. May be repeated with change in topic. | | |
|  | |  |  | | --- | --- | | 2010-2011 University Catalog 2010-2011 University Catalog | [Close Window](javascript:window.close();) | | | |
|  | |  |  | | --- | --- | |  | •  [SWE 205 - Software Usability Analysis and Design](http://catalog.gmu.edu/preview_course_nopop.php?catoid=15&coid=100200) | |  | •  [SWE 301 - Internship Preparation](http://catalog.gmu.edu/preview_course_nopop.php?catoid=15&coid=100201) | |  | •  [SWE 332 - Object-Oriented Software Design and Implementation](http://catalog.gmu.edu/preview_course_nopop.php?catoid=15&coid=100202) | |  | •  [SWE 401 - Internship Reflection](http://catalog.gmu.edu/preview_course_nopop.php?catoid=15&coid=100203) | |  | •  [SWE 421 - Software Requirements and Design Modeling](http://catalog.gmu.edu/preview_course_nopop.php?catoid=15&coid=100204) | |  | •  [SWE 432 - Design and Implementation of Software for the Web](http://catalog.gmu.edu/preview_course_nopop.php?catoid=15&coid=100205) | |  | •  [SWE 437 - Software Testing and Maintenance](http://catalog.gmu.edu/preview_course_nopop.php?catoid=15&coid=100206) | |  | •  [SWE 443 - Software Architectures](http://catalog.gmu.edu/preview_course_nopop.php?catoid=15&coid=100207) | |  | •  [SWE 510 - Object-Oriented Programming in Java](http://catalog.gmu.edu/preview_course_nopop.php?catoid=15&coid=100208) | |  | •  [SWE 521 - Software Engineering Essentials](http://catalog.gmu.edu/preview_course_nopop.php?catoid=15&coid=100557) | |  | •  [SWE 619 - Object-Oriented Software Specification and Construction](http://catalog.gmu.edu/preview_course_nopop.php?catoid=15&coid=100209) | |  | •  [SWE 620 - Software Requirements Analysis and Specification](http://catalog.gmu.edu/preview_course_nopop.php?catoid=15&coid=100210) | |  | •  [SWE 621 - Software Modeling and Architectural Design](http://catalog.gmu.edu/preview_course_nopop.php?catoid=15&coid=100211) | |  | •  [SWE 622 - Distributed Software Engineering](http://catalog.gmu.edu/preview_course_nopop.php?catoid=15&coid=100212) | |  | •  [SWE 623 - Formal Methods and Models in Software Engineering](http://catalog.gmu.edu/preview_course_nopop.php?catoid=15&coid=100213) | |  | •  [SWE 625 - Software Project Management](http://catalog.gmu.edu/preview_course_nopop.php?catoid=15&coid=100214) | |  | •  [SWE 626 - Software Project Laboratory](http://catalog.gmu.edu/preview_course_nopop.php?catoid=15&coid=100215) | |  | •  [SWE 630 - Software Engineering Economics](http://catalog.gmu.edu/preview_course_nopop.php?catoid=15&coid=100216) | |  | •  [SWE 631 - Object-Oriented Design Patterns](http://catalog.gmu.edu/preview_course_nopop.php?catoid=15&coid=100217) | |  | •  [SWE 632 - User Interface Design and Development](http://catalog.gmu.edu/preview_course_nopop.php?catoid=15&coid=100218) | |  | •  [SWE 637 - Software Testing](http://catalog.gmu.edu/preview_course_nopop.php?catoid=15&coid=100219) | |  | •  [SWE 641 - Systems Architecture for Large-Scale Systems](http://catalog.gmu.edu/preview_course_nopop.php?catoid=15&coid=100220) | |  | •  [SWE 642 - Software Engineering for the World Wide Web](http://catalog.gmu.edu/preview_course_nopop.php?catoid=15&coid=100221) | |  | •  [SWE 645 - Component-Based Software Development](http://catalog.gmu.edu/preview_course_nopop.php?catoid=15&coid=100222) | |  | •  [SWE 699 - Special Topics in Software Engineering](http://catalog.gmu.edu/preview_course_nopop.php?catoid=15&coid=100223) | |  | •  [SWE 720 - Advanced Software Requirements](http://catalog.gmu.edu/preview_course_nopop.php?catoid=15&coid=100224) | |  | •  [SWE 721 - Reusable Software Architectures](http://catalog.gmu.edu/preview_course_nopop.php?catoid=15&coid=100225) | |  | •  [SWE 723 - Precise Modeling](http://catalog.gmu.edu/preview_course_nopop.php?catoid=15&coid=100226) | |  | •  [SWE 727 - Quality of Service for Software Architectures](http://catalog.gmu.edu/preview_course_nopop.php?catoid=15&coid=100228) | |  | •  [SWE 763 - Software Engineering Experimentation](http://catalog.gmu.edu/preview_course_nopop.php?catoid=15&coid=100229) | |  | •  [SWE 781 - Secure Software Design and Programming](http://catalog.gmu.edu/preview_course_nopop.php?catoid=15&coid=100230) | |  | •  [SWE 795 - Advanced Topics in Software Engineering](http://catalog.gmu.edu/preview_course_nopop.php?catoid=15&coid=100231) | |  | •  [SWE 796 - Directed Readings in Software Engineering](http://catalog.gmu.edu/preview_course_nopop.php?catoid=15&coid=100232) | |  | •  [SWE 798 - Research Project](http://catalog.gmu.edu/preview_course_nopop.php?catoid=15&coid=100233) | |  | •  [SWE 799 - Thesis](http://catalog.gmu.edu/preview_course_nopop.php?catoid=15&coid=100234) | |  | •  [SWE 821 - Software Engineering Seminar](http://catalog.gmu.edu/preview_course_nopop.php?catoid=15&coid=107045) | |  | •  [SWE 823 - Software for Critical Systems](http://catalog.gmu.edu/preview_course_nopop.php?catoid=15&coid=107046) | |  | •  [SWE 824 - Program Analysis for Software Testing](http://catalog.gmu.edu/preview_course_nopop.php?catoid=15&coid=107047) | |  | •  [SWE 825 - Special Topics in Web-Based Software](http://catalog.gmu.edu/preview_course_nopop.php?catoid=15&coid=107048) | |  | •  [SWE 860 - Software Analysis and Design of Real-Time Systems](http://catalog.gmu.edu/preview_course_nopop.php?catoid=15&coid=107050) |   This course information is from the 2010-2011 University Catalog. [View this catalog](http://catalog.gmu.edu/index.php?catoid=15). | | | |
|  |  | | |  |

**Computer science**

**From Wikipedia, the free encyclopedia**

Jump to: [navigation](http://en.wikipedia.org/wiki/Computer_science#mw-head), [search](http://en.wikipedia.org/wiki/Computer_science#p-search)

|  |  |
| --- | --- |
| [Utah teapot representing computer graphics](http://en.wikipedia.org/wiki/Computer_graphics_(computer_science)) | [Plot of a quicksort algorithm](http://en.wikipedia.org/wiki/Computational_complexity_theory) |
| [large capital lambda](http://en.wikipedia.org/wiki/Programming_language_theory) | [Microsoft Tastenmaus mouse representing human-computer interaction](http://en.wikipedia.org/wiki/Human%E2%80%93computer_interaction) |

Computer science deals with the theoretical foundations of information and computation, and with practical techniques for their implementation and application.

Computer science or computing science (sometimes abbreviated CS) is the study of the theoretical foundations of [information](http://en.wikipedia.org/wiki/Information) and [computation](http://en.wikipedia.org/wiki/Computation) and of practical techniques for their implementation and application in [computer](http://en.wikipedia.org/wiki/Computer) systems.[[1]](http://en.wikipedia.org/wiki/Computer_science#cite_note-0)[[2]](http://en.wikipedia.org/wiki/Computer_science#cite_note-1)[[3]](http://en.wikipedia.org/wiki/Computer_science#cite_note-2)[[4]](http://en.wikipedia.org/wiki/Computer_science#cite_note-3) It is frequently described as the systematic study of [algorithmic](http://en.wikipedia.org/wiki/Algorithm) processes that create, describe, and transform information. Computer science has many [sub-fields](http://en.wikipedia.org/wiki/List_of_computer_science_fields); some, such as [computer graphics](http://en.wikipedia.org/wiki/Computer_graphics), emphasize the computation of specific results, while others, such as [computational complexity theory](http://en.wikipedia.org/wiki/Computational_complexity_theory), study the properties of [computational problems](http://en.wikipedia.org/wiki/Computational_problem). Still others focus on the challenges in implementing computations. For example, [programming language theory](http://en.wikipedia.org/wiki/Programming_language_theory) studies approaches to describe computations, while [computer programming](http://en.wikipedia.org/wiki/Computer_programming) applies specific [programming languages](http://en.wikipedia.org/wiki/Programming_language) to solve specific computational problems, and [human-computer interaction](http://en.wikipedia.org/wiki/Human-computer_interaction) focuses on the challenges in making computers and computations useful, usable, and universally accessible to [people](http://en.wikipedia.org/wiki/Humans).

The general public sometimes confuses computer science with careers that deal with computers (such as [information technology](http://en.wikipedia.org/wiki/Information_technology)), or think that it relates to their own experience of computers, which typically involves activities such as gaming, web-browsing, and word-processing. However, the focus of computer science is more on understanding the properties of the programs used to implement software such as games and web-browsers, and using that understanding to create new programs or improve existing ones.[[5]](http://en.wikipedia.org/wiki/Computer_science#cite_note-4)

|  |
| --- |
| Contents  [[hide](http://en.wikipedia.org/wiki/Computer_science)]  [1 History](http://en.wikipedia.org/wiki/Computer_science#History)  [2 Major achievements](http://en.wikipedia.org/wiki/Computer_science#Major_achievements)  [3 Areas of computer science](http://en.wikipedia.org/wiki/Computer_science#Areas_of_computer_science)  [3.1 Theoretical computer science](http://en.wikipedia.org/wiki/Computer_science#Theoretical_computer_science)  [3.1.1 Theory of computation](http://en.wikipedia.org/wiki/Computer_science#Theory_of_computation)  [3.2 Algorithms and data structures](http://en.wikipedia.org/wiki/Computer_science#Algorithms_and_data_structures)  [3.3 Computer elements and architecture](http://en.wikipedia.org/wiki/Computer_science#Computer_elements_and_architecture)  [3.4 Multiprocessing](http://en.wikipedia.org/wiki/Computer_science#Multiprocessing)  [3.5 Computational science](http://en.wikipedia.org/wiki/Computer_science#Computational_science)  [3.6 Artificial Intelligence](http://en.wikipedia.org/wiki/Computer_science#Artificial_Intelligence)  [3.7 Software Systems](http://en.wikipedia.org/wiki/Computer_science#Software_Systems)  [4 Paradigms](http://en.wikipedia.org/wiki/Computer_science#Paradigms)  [5 Relationship with other fields](http://en.wikipedia.org/wiki/Computer_science#Relationship_with_other_fields)  [6 Computer science education](http://en.wikipedia.org/wiki/Computer_science#Computer_science_education)  [7 See also](http://en.wikipedia.org/wiki/Computer_science#See_also)  [8 References](http://en.wikipedia.org/wiki/Computer_science#References)  [9 Further reading](http://en.wikipedia.org/wiki/Computer_science#Further_reading)  [10 External links](http://en.wikipedia.org/wiki/Computer_science#External_links) |

[[edit](http://en.wikipedia.org/w/index.php?title=Computer_science&action=edit&section=1)] History

Main article: [**History of computer science**](http://en.wikipedia.org/wiki/History_of_computer_science)

|  |  |
| --- | --- |
| http://upload.wikimedia.org/wikipedia/commons/thumb/9/91/Wikiversity-logo.svg/40px-Wikiversity-logo.svg.png | Wikiversity has learning materials about [computing](http://en.wikiversity.org/wiki/computing) |

The early foundations of what would become computer science predate the invention of the modern [digital computer](http://en.wikipedia.org/wiki/Digital_computer). Machines for calculating fixed numerical tasks, such as the [abacus](http://en.wikipedia.org/wiki/Abacus), have existed since antiquity. [Wilhelm Schickard](http://en.wikipedia.org/wiki/Wilhelm_Schickard) designed the first mechanical calculator in 1623, but did not complete its construction.[[6]](http://en.wikipedia.org/wiki/Computer_science#cite_note-5) [Blaise Pascal](http://en.wikipedia.org/wiki/Blaise_Pascal) designed and constructed the first working mechanical calculator, the [Pascaline](http://en.wikipedia.org/wiki/Pascal%27s_calculator), in 1642. [Charles Babbage](http://en.wikipedia.org/wiki/Charles_Babbage) designed a [difference engine](http://en.wikipedia.org/wiki/Difference_engine) in [Victorian](http://en.wikipedia.org/wiki/Victorian_era) times[[7]](http://en.wikipedia.org/wiki/Computer_science#cite_note-6) helped by [Ada Lovelace](http://en.wikipedia.org/wiki/Ada_Lovelace).[[8]](http://en.wikipedia.org/wiki/Computer_science#cite_note-7) Around 1900, [punch-card machines](http://en.wikipedia.org/wiki/Key_punch)[[9]](http://en.wikipedia.org/wiki/Computer_science#cite_note-8) were introduced. However, all of these machines were constrained to perform a single task, or at best some subset of all possible tasks.

During the 1940s, as newer and more powerful computing machines were developed, the term computer came to refer to the machines rather than their human predecessors.[[10]](http://en.wikipedia.org/wiki/Computer_science#cite_note-9) As it became clear that computers could be used for more than just mathematical calculations, the field of computer science broadened to study [computation](http://en.wikipedia.org/wiki/Computation) in general. Computer science began to be established as a distinct academic discipline in the 1950s and early 1960s.[[11]](http://en.wikipedia.org/wiki/Computer_science#cite_note-Denning_cs_discipline-10)[[12]](http://en.wikipedia.org/wiki/Computer_science#cite_note-11) The first computer science degree program in the United States was formed at [Purdue University](http://en.wikipedia.org/wiki/Purdue_University) in 1962.[[13]](http://en.wikipedia.org/wiki/Computer_science#cite_note-12) Since practical computers became available, many applications of computing have become distinct areas of study in their own right.

Although many initially believed it was impossible that computers themselves could actually be a scientific field of study, in the late fifties it gradually became accepted among the greater academic population.[[14]](http://en.wikipedia.org/wiki/Computer_science#cite_note-Levy1984-13) It is the now well-known [IBM](http://en.wikipedia.org/wiki/IBM) brand that formed part of the computer science revolution during this time. IBM (short for International Business Machines) released the IBM 704 and later the IBM 709 computers, which were widely used during the exploration period of such devices. "Still, working with the IBM [computer] was frustrating...if you had misplaced as much as one letter in one instruction, the program would crash, and you would have to start the whole process over again".[[14]](http://en.wikipedia.org/wiki/Computer_science#cite_note-Levy1984-13) During the late 1950s, the computer science discipline was very much in its developmental stages, and such issues were commonplace.

Time has seen significant improvements in the usability and effectiveness of computer science technology. Modern society has seen a significant shift from computers being used solely by experts or professionals to a more widespread user base. Initially, computers were quite costly, and for their most-effective use, some degree of human aid was needed, in part by professional computer operators. However, as computers became widespread and far more affordable, less human assistance was needed, although residues of the original assistance still remained.

[[edit](http://en.wikipedia.org/w/index.php?title=Computer_science&action=edit&section=2)] Major achievements

|  |  |
| --- | --- |
| [Wiki letter w cropped.svg](http://en.wikipedia.org/wiki/File:Wiki_letter_w_cropped.svg) | This section requires [expansion](http://en.wikipedia.org/w/index.php?title=Computer_science&action=edit). |

[](http://en.wikipedia.org/wiki/File:Enigma.jpg)

[http://bits.wikimedia.org/skins-1.5/common/images/magnify-clip.png](http://en.wikipedia.org/wiki/File:Enigma.jpg)

The [German](http://en.wikipedia.org/wiki/Germany) military used the [Enigma machine](http://en.wikipedia.org/wiki/Enigma_machine) (shown here) during [World War II](http://en.wikipedia.org/wiki/World_War_II) for communication they thought to be secret. The large-scale decryption of Enigma traffic at [Bletchley Park](http://en.wikipedia.org/wiki/Bletchley_Park) was an important factor that contributed to Allied victory in WWII.[[15]](http://en.wikipedia.org/wiki/Computer_science#cite_note-kahnbook-14)

Despite its short history as a formal academic discipline, computer science has made a number of fundamental contributions to [science](http://en.wikipedia.org/wiki/Science) and [society](http://en.wikipedia.org/wiki/Society). These include:

The start of the "digital revolution," which includes the current [Information Age](http://en.wikipedia.org/wiki/Information_Age) and the [Internet](http://en.wikipedia.org/wiki/Internet).[[16]](http://en.wikipedia.org/wiki/Computer_science#cite_note-bgu-15)

A formal definition of [computation](http://en.wikipedia.org/wiki/Computation) and [computability](http://en.wikipedia.org/wiki/Computability), and proof that there are computationally [unsolvable](http://en.wikipedia.org/wiki/Undecidable_problem) and [intractable](http://en.wikipedia.org/wiki/Computational_complexity_theory#Intractability) problems.[[17]](http://en.wikipedia.org/wiki/Computer_science#cite_note-16)

The concept of a [programming language](http://en.wikipedia.org/wiki/Programming_language), a tool for the precise expression of methodological information at various levels of abstraction.[[18]](http://en.wikipedia.org/wiki/Computer_science#cite_note-17)

In [cryptography](http://en.wikipedia.org/wiki/Cryptography), [breaking the Enigma machine](http://en.wikipedia.org/wiki/Cryptanalysis_of_the_Enigma) was an important factor contributing to the Allied victory in World War II.[[15]](http://en.wikipedia.org/wiki/Computer_science#cite_note-kahnbook-14)

[Scientific computing](http://en.wikipedia.org/wiki/Scientific_computing) enabled practical evaluation of processes and situations of great complexity, as well as experimentation entirely by software. It also enabled advanced study of the mind, and mapping of the human genome became possible with the [Human Genome Project](http://en.wikipedia.org/wiki/Human_Genome_Project).[[16]](http://en.wikipedia.org/wiki/Computer_science#cite_note-bgu-15) [Distributed computing](http://en.wikipedia.org/wiki/Distributed_computing) projects such as [Folding@home](http://en.wikipedia.org/wiki/Folding@home) explore [protein folding](http://en.wikipedia.org/wiki/Protein_folding).

[Algorithmic trading](http://en.wikipedia.org/wiki/Algorithmic_trading) has increased the [efficiency](http://en.wikipedia.org/wiki/Economic_efficiency) and [liquidity](http://en.wikipedia.org/wiki/Market_liquidity) of financial markets by using [artificial intelligence](http://en.wikipedia.org/wiki/Artificial_intelligence), [machine learning](http://en.wikipedia.org/wiki/Machine_learning), and other [statistical](http://en.wikipedia.org/wiki/Statistics) and [numerical](http://en.wikipedia.org/wiki/Numerical_analysis) techniques on a large scale.[[19]](http://en.wikipedia.org/wiki/Computer_science#cite_note-18)

Image synthesis, including video by computing individual video frames.[[citation needed](http://en.wikipedia.org/wiki/Wikipedia:Citation_needed)]

Human language processing, including practical speech-to-text conversion and automated translation of languages[[citation needed](http://en.wikipedia.org/wiki/Wikipedia:Citation_needed)]

[Simulation](http://en.wikipedia.org/wiki/Simulation) of various processes, including computational fluid dynamics, physical, electrical, and electronic systems and circuits, as well as societies and social situations (notably war games) along with their habitats, among many others. Modern computers enable optimization of such designs as complete aircraft. Notable in electrical and electronic circuit design are [SPICE](http://en.wikipedia.org/wiki/SPICE) as well as software for physical realization of new (or modified) designs. The latter includes essential design software for integrated circuits.[[citation needed](http://en.wikipedia.org/wiki/Wikipedia:Citation_needed)]

[[edit](http://en.wikipedia.org/w/index.php?title=Computer_science&action=edit&section=3)] Areas of computer science

As a discipline, computer science spans a range of topics from theoretical studies of algorithms and the limits of computation to the practical issues of implementing computing systems in hardware and software.[[20]](http://en.wikipedia.org/wiki/Computer_science#cite_note-CSAB1997-19)[[21]](http://en.wikipedia.org/wiki/Computer_science#cite_note-20) [CSAB](http://en.wikipedia.org/wiki/CSAB_(professional_organization)), formerly called Computing Sciences Accreditation Board – which is made up of representatives of the [Association for Computing Machinery](http://en.wikipedia.org/wiki/Association_for_Computing_Machinery) (ACM), and the [IEEE Computer Society](http://en.wikipedia.org/wiki/IEEE_Computer_Society) (IEEE-CS) [[22]](http://en.wikipedia.org/wiki/Computer_science#cite_note-21) – identifies four areas that it considers crucial to the discipline of computer science: theory of computation, algorithms and data structures, programming methodology and languages, and computer elements and architecture. In addition to these four areas, CSAB also identifies fields such as software engineering, artificial intelligence, computer networking and communication, database systems, parallel computation, distributed computation, computer-human interaction, computer graphics, operating systems, and numerical and symbolic computation as being important areas of computer science.[[20]](http://en.wikipedia.org/wiki/Computer_science#cite_note-CSAB1997-19)

[[edit](http://en.wikipedia.org/w/index.php?title=Computer_science&action=edit&section=4)] Theoretical computer science

The broader field of [theoretical computer science](http://en.wikipedia.org/wiki/Theoretical_computer_science) encompasses both the classical theory of computation and a wide range of other topics that focus on the more abstract, logical, and mathematical aspects of computing.

|  |  |  |  |
| --- | --- | --- | --- |
| P \rightarrow Q \, | [DFAexample.svg](http://en.wikipedia.org/wiki/File:DFAexample.svg) | [Elliptic curve simple.png](http://en.wikipedia.org/wiki/File:Elliptic_curve_simple.png) | [6n-graf.svg](http://en.wikipedia.org/wiki/File:6n-graf.svg) |
| [Mathematical logic](http://en.wikipedia.org/wiki/Mathematical_logic) | [Automata theory](http://en.wikipedia.org/wiki/Automata_theory) | [Number theory](http://en.wikipedia.org/wiki/Computational_number_theory) | [Graph theory](http://en.wikipedia.org/wiki/Graph_theory) |
| \Gamma\vdash x: Int | [Commutative diagram for morphism.svg](http://en.wikipedia.org/wiki/File:Commutative_diagram_for_morphism.svg) | [SimplexRangeSearching.png](http://en.wikipedia.org/wiki/File:SimplexRangeSearching.png) | [Blochsphere.svg](http://en.wikipedia.org/wiki/File:Blochsphere.svg) |
| [Type theory](http://en.wikipedia.org/wiki/Type_theory) | [Category theory](http://en.wikipedia.org/wiki/Category_theory) | [Computational geometry](http://en.wikipedia.org/wiki/Computational_geometry) | [Quantum computing theory](http://en.wikipedia.org/wiki/Quantum_computer) |

[[edit](http://en.wikipedia.org/w/index.php?title=Computer_science&action=edit&section=5)] **Theory of computation**

According to [Peter J. Denning](http://en.wikipedia.org/wiki/Peter_J._Denning), the fundamental question underlying computer science is, "What can be (efficiently) automated?"[[11]](http://en.wikipedia.org/wiki/Computer_science#cite_note-Denning_cs_discipline-10) The study of the [theory of computation](http://en.wikipedia.org/wiki/Theory_of_computation) is focused on answering fundamental questions about what can be computed and what amount of resources are required to perform those computations. In an effort to answer the first question, [computability theory](http://en.wikipedia.org/wiki/Computability_theory_(computer_science)) examines which computational problems are solvable on various theoretical [models of computation](http://en.wikipedia.org/wiki/Models_of_computation). The second question is addressed by [computational complexity theory](http://en.wikipedia.org/wiki/Computational_complexity_theory), which studies the time and space costs associated with different approaches to solving a computational problem.

The famous "[P=NP?](http://en.wikipedia.org/wiki/P_%3D_NP_problem)" problem, one of the [Millennium Prize Problems](http://en.wikipedia.org/wiki/Millennium_Prize_Problems),[[23]](http://en.wikipedia.org/wiki/Computer_science#cite_note-22) is an open problem in the theory of computation.

|  |  |  |
| --- | --- | --- |
| [Wang tiles.png](http://en.wikipedia.org/wiki/File:Wang_tiles.png) | P = NP ? | GNITIRW-TERCES |
| Computability theory | [Computational complexity theory](http://en.wikipedia.org/wiki/Computational_complexity_theory) | [Cryptography](http://en.wikipedia.org/wiki/Cryptography) |

[[edit](http://en.wikipedia.org/w/index.php?title=Computer_science&action=edit&section=6)] Algorithms and data structures

|  |  |  |
| --- | --- | --- |
| O(n2) | [Sorting quicksort anim.gif](http://en.wikipedia.org/wiki/File:Sorting_quicksort_anim.gif) | [Singly linked list.png](http://en.wikipedia.org/wiki/File:Singly_linked_list.png) |
| [Analysis of algorithms](http://en.wikipedia.org/wiki/Analysis_of_algorithms) | [Algorithms](http://en.wikipedia.org/wiki/Algorithms) | [Data structures](http://en.wikipedia.org/wiki/Data_structures) |

[[edit](http://en.wikipedia.org/w/index.php?title=Computer_science&action=edit&section=7)] Computer elements and architecture

|  |  |  |
| --- | --- | --- |
| [NOR ANSI.svg](http://en.wikipedia.org/wiki/File:NOR_ANSI.svg) | [Fivestagespipeline.png](http://en.wikipedia.org/wiki/File:Fivestagespipeline.png) | [SIMD.svg](http://en.wikipedia.org/wiki/File:SIMD.svg) |
| [Digital logic](http://en.wikipedia.org/wiki/Digital_logic) | [Microarchitecture](http://en.wikipedia.org/wiki/Microarchitecture) | [Multiprocessing](http://en.wikipedia.org/wiki/Multiprocessing) |

[[edit](http://en.wikipedia.org/w/index.php?title=Computer_science&action=edit&section=8)] **Multiprocessing**

Multiprocessing is running a system with more than one processor. The theory is of course that you can double performance by using two processors instead of one. And the reality of course is that it doesn't work this well, although multiprocessing can result in improved performance under certain conditions. In order to employ multiprocessing effectively, the computer system must have all of the following in place:

**Motherboard Support**

A motherboard capable of handling multiple processors. This means additional sockets or slots for the extra chips, and a chipset capable of handling the multiprocessing arrangement.

**Processor Support**

Processors that are capable of being used in a multiprocessing system. Not all are, and in fact some versions of the same processor are while others are not.

**Operating System Support**

An operating system that supports multiprocessing, such as Windows NT or one of the various flavors of UNIX.[[24]](http://en.wikipedia.org/wiki/Computer_science#cite_note-23)

[[edit](http://en.wikipedia.org/w/index.php?title=Computer_science&action=edit&section=9)] **Computational science**

[Computational science](http://en.wikipedia.org/wiki/Computational_science) (or [scientific computing](http://en.wikipedia.org/wiki/Scientific_computing)) is the field of study concerned with constructing [mathematical models](http://en.wikipedia.org/wiki/Scientific_modelling) and [quantitative analysis](http://en.wikipedia.org/wiki/Numerical_analysis) techniques and using computers to analyse and solve [scientific](http://en.wikipedia.org/wiki/Scientific) problems. In practical use, it is typically the application of [computer simulation](http://en.wikipedia.org/wiki/Computer_simulation) and other forms of [computation](http://en.wikipedia.org/wiki/Computation) to problems in various scientific disciplines.

|  |  |  |  |
| --- | --- | --- | --- |
| [Lorenz attractor yb.svg](http://en.wikipedia.org/wiki/File:Lorenz_attractor_yb.svg) | [Quark wiki.jpg](http://en.wikipedia.org/wiki/File:Quark_wiki.jpg) | [Naphthalene-3D-balls.png](http://en.wikipedia.org/wiki/File:Naphthalene-3D-balls.png) | [1u04-argonaute.png](http://en.wikipedia.org/wiki/File:1u04-argonaute.png) |
| [**Numerical analysis**](http://en.wikipedia.org/wiki/Numerical_analysis) | [**Computational physics**](http://en.wikipedia.org/wiki/Computational_physics) | [**Computational chemistry**](http://en.wikipedia.org/wiki/Computational_chemistry) | [**Bioinformatics**](http://en.wikipedia.org/wiki/Bioinformatics) |

[[edit](http://en.wikipedia.org/w/index.php?title=Computer_science&action=edit&section=10)] **Artificial Intelligence**

This branch of computer science aims to create synthetic systems which solve computational problems, reason and/or communicate like animals and humans do. This theoretical and applied subfield requires a very rigorous and integrated expertise in multiple subject areas such as [applied mathematics](http://en.wikipedia.org/wiki/Applied_mathematics), [logic](http://en.wikipedia.org/wiki/Logic), [semiotics](http://en.wikipedia.org/wiki/Semiotics), [electrical engineering](http://en.wikipedia.org/wiki/Electrical_engineering), [philosophy of mind](http://en.wikipedia.org/wiki/Philosophy_of_mind), [neurophysiology](http://en.wikipedia.org/wiki/Neurophysiology), and [social intelligence](http://en.wikipedia.org/wiki/Social_intelligence) which can be used to advance the field of intelligence research or be applied to other subject areas which require computational understanding and modelling such as in [finance](http://en.wikipedia.org/wiki/Finance) or the [physical sciences](http://en.wikipedia.org/wiki/Physical_sciences). This field started in full earnest when [Alan Turing](http://en.wikipedia.org/wiki/Alan_Turing), the pioneer of computer science and [artificial intelligence](http://en.wikipedia.org/wiki/Artificial_intelligence), proposed the [Turing Test](http://en.wikipedia.org/wiki/Turing_Test) for the purpose of answering the ultimate question... "Can computers think ?".

|  |  |  |  |
| --- | --- | --- | --- |
| [Brain.png](http://en.wikipedia.org/wiki/File:Brain.png) | [Eye.png](http://en.wikipedia.org/wiki/File:Eye.png) | [Corner.png](http://en.wikipedia.org/wiki/File:Corner.png) | [KnnClassification.svg](http://en.wikipedia.org/wiki/File:KnnClassification.svg) |
| [Machine Learning](http://en.wikipedia.org/wiki/Machine_Learning) | [Computer vision](http://en.wikipedia.org/wiki/Computer_vision) | [Image Processing](http://en.wikipedia.org/wiki/Image_Processing) | [Pattern Recognition](http://en.wikipedia.org/wiki/Pattern_Recognition) |
| [User-FastFission-brain.gif](http://en.wikipedia.org/wiki/File:User-FastFission-brain.gif) | [Data.png](http://en.wikipedia.org/wiki/File:Data.png) | [Sky.png](http://en.wikipedia.org/wiki/File:Sky.png) | [Earth.png](http://en.wikipedia.org/wiki/File:Earth.png) |
| [Cognitive Science](http://en.wikipedia.org/wiki/Cognitive_Science) | [Data Mining](http://en.wikipedia.org/wiki/Data_Mining) | [Evolutionary Computation](http://en.wikipedia.org/wiki/Evolutionary_Computation) | [Information Retrieval](http://en.wikipedia.org/wiki/Information_Retrieval) |
| [Neuron.png](http://en.wikipedia.org/wiki/File:Neuron.png) | [English.png](http://en.wikipedia.org/wiki/File:English.png) | [HONDA ASIMO.jpg](http://en.wikipedia.org/wiki/File:HONDA_ASIMO.jpg) | [Wacom Pen-tablet.jpg](http://en.wikipedia.org/wiki/File:Wacom_Pen-tablet.jpg) |
| [Knowledge Representation](http://en.wikipedia.org/wiki/Knowledge_Representation) | [Natural Language Processing](http://en.wikipedia.org/wiki/Natural_Language_Processing) | [Robotics](http://en.wikipedia.org/wiki/Robotics) | [Human–computer interaction](http://en.wikipedia.org/wiki/Human%E2%80%93computer_interaction) |

[[edit](http://en.wikipedia.org/w/index.php?title=Computer_science&action=edit&section=11)] **Software Systems**

Main article: [Software systems](http://en.wikipedia.org/wiki/Software_systems)

The field of [software systems](http://en.wikipedia.org/wiki/Software_systems) encompasses a wide range of sub-fields, which involve principled design and analysis of various kinds of software based systems. These fields use concepts and techniques from [theoretical computer science](http://en.wikipedia.org/wiki/Theoretical_computer_science), [artificial intelligence](http://en.wikipedia.org/wiki/Artificial_intelligence), [mathematics](http://en.wikipedia.org/wiki/Mathematics), [electrical engineering](http://en.wikipedia.org/wiki/Electrical_engineering), and so on to build software systems for various real world tasks.

|  |  |  |  |
| --- | --- | --- | --- |
| [Operating system placement.svg](http://en.wikipedia.org/wiki/File:Operating_system_placement.svg) | [NETWORK-Library-LAN.png](http://en.wikipedia.org/wiki/File:NETWORK-Library-LAN.png) | [Emp Tables (Database).PNG](http://en.wikipedia.org/wiki/File:Emp_Tables_(Database).PNG) | [Padlock.svg](http://en.wikipedia.org/wiki/File:Padlock.svg) |
| [Operating systems](http://en.wikipedia.org/wiki/Operating_systems) | [Computer networks](http://en.wikipedia.org/wiki/Computer_networking) | [Databases](http://en.wikipedia.org/wiki/Databases) | [Computer security](http://en.wikipedia.org/wiki/Computer_security) |
| [Roomba original.jpg](http://en.wikipedia.org/wiki/File:Roomba_original.jpg) | [Flowchart.png](http://en.wikipedia.org/wiki/File:Flowchart.png) | [Ideal compiler.png](http://en.wikipedia.org/wiki/File:Ideal_compiler.png) | [Python add5 syntax.svg](http://en.wikipedia.org/wiki/File:Python_add5_syntax.svg) |
| [Ubiquitous computing](http://en.wikipedia.org/wiki/Ubiquitous_computing) | [Systems architecture](http://en.wikipedia.org/wiki/Systems_architecture) | [Compiler design](http://en.wikipedia.org/wiki/Compiler_design) | [Programming languages](http://en.wikipedia.org/wiki/Programming_languages) |

[[edit](http://en.wikipedia.org/w/index.php?title=Computer_science&action=edit&section=12)] **Paradigms**

Following [Peter Wegner](http://en.wikipedia.org/wiki/Peter_Wegner), Amnon H. Eden proposes that there are three [paradigms](http://en.wikipedia.org/wiki/Paradigm) at work in various areas of computer science:[[25]](http://en.wikipedia.org/wiki/Computer_science#cite_note-24) a "rationalist paradigm", which treats computer science as branch of mathematics, which is prevalent in theoretical computer science, and mainly employs [deductive reasoning](http://en.wikipedia.org/wiki/Deductive_reasoning), a "technocratic paradigm", readily identifiable with [engineering](http://en.wikipedia.org/wiki/Engineering) approaches, most prominent in software engineering, and a "scientific paradigm", which approaches computer-related artifacts from the empirical perspective of [natural sciences](http://en.wikipedia.org/wiki/Natural_science), and identifiable in some branches of artificial intelligence (the study of [artificial life](http://en.wikipedia.org/wiki/Artificial_life) for instance).

[[edit](http://en.wikipedia.org/w/index.php?title=Computer_science&action=edit&section=13)] **Relationship with other fields**

Despite its name, a significant amount of computer science does not involve the study of computers themselves. Because of this, several alternative names have been proposed. Certain departments of major universities prefer the term computing science, to emphasize precisely that difference. Danish scientist [Peter Naur](http://en.wikipedia.org/wiki/Peter_Naur) suggested the term datalogy, to reflect the fact that the scientific discipline revolves around data and data treatment, while not necessarily involving computers. The first scientific institution to use the term was the Department of Datalogy at the University of Copenhagen, founded in 1969, with Peter Naur being the first professor in datalogy. The term is used mainly in the Scandinavian countries. Also, in the early days of computing, a number of terms for the practitioners of the field of computing were suggested in the Communications of the ACM – turingineer, turologist, flow-charts-man, applied meta-mathematician, and applied epistemologist.[[26]](http://en.wikipedia.org/wiki/Computer_science#cite_note-25) Three months later in the same journal, comptologist was suggested, followed next year by hypologist.[[27]](http://en.wikipedia.org/wiki/Computer_science#cite_note-26) The term computics has also been suggested.[[28]](http://en.wikipedia.org/wiki/Computer_science#cite_note-27) In continental Europe, names such as informatique (French), Informatik (German) or informatica (Dutch), derived from information and possibly mathematics or automatic, are more common than names derived from computer/computation.

The renowned computer scientist [Edsger Dijkstra](http://en.wikipedia.org/wiki/Edsger_W._Dijkstra) stated, "Computer science is no more about computers than astronomy is about telescopes." The design and deployment of computers and computer systems is generally considered the province of disciplines other than computer science. For example, the study of [computer hardware](http://en.wikipedia.org/wiki/Computer_hardware) is usually considered part of [computer engineering](http://en.wikipedia.org/wiki/Computer_engineering), while the study of commercial [computer systems](http://en.wikipedia.org/wiki/Computer_system) and their deployment is often called [information technology](http://en.wikipedia.org/wiki/Information_technology) or [information systems](http://en.wikipedia.org/wiki/Information_systems). However, there has been much cross-fertilization of ideas between the various computer-related disciplines. Computer science research has also often crossed into other disciplines, such as [philosophy](http://en.wikipedia.org/wiki/Philosophy), [cognitive science](http://en.wikipedia.org/wiki/Cognitive_science), [linguistics](http://en.wikipedia.org/wiki/Linguistics), [mathematics](http://en.wikipedia.org/wiki/Mathematics), [physics](http://en.wikipedia.org/wiki/Physics), [statistics](http://en.wikipedia.org/wiki/Computational_statistics), and [economics](http://en.wikipedia.org/wiki/Computational_economics).

Computer science is considered by some to have a much closer relationship with mathematics than many scientific disciplines, with some observers saying that computing is a mathematical science.[[11]](http://en.wikipedia.org/wiki/Computer_science#cite_note-Denning_cs_discipline-10) Early computer science was strongly influenced by the work of mathematicians such as [Kurt Gödel](http://en.wikipedia.org/wiki/Kurt_G%C3%B6del) and [Alan Turing](http://en.wikipedia.org/wiki/Alan_Turing), and there continues to be a useful interchange of ideas between the two fields in areas such as [mathematical logic](http://en.wikipedia.org/wiki/Mathematical_logic), [category theory](http://en.wikipedia.org/wiki/Category_theory), [domain theory](http://en.wikipedia.org/wiki/Domain_theory), and [algebra](http://en.wikipedia.org/wiki/Algebra).

The relationship between computer science and [software engineering](http://en.wikipedia.org/wiki/Software_engineering) is a contentious issue, which is further muddied by [disputes](http://en.wikipedia.org/wiki/Debates_within_software_engineering) over what the term "software engineering" means, and how computer science is defined. [David Parnas](http://en.wikipedia.org/wiki/David_Parnas), taking a cue from the relationship between other engineering and science disciplines, has claimed that the principal focus of computer science is studying the properties of computation in general, while the principal focus of software engineering is the design of specific computations to achieve practical goals, making the two separate but complementary disciplines.[[29]](http://en.wikipedia.org/wiki/Computer_science#cite_note-28)

The academic, political, and funding aspects of computer science tend to depend on whether a department formed with a mathematical emphasis or with an engineering emphasis. Computer science departments with a mathematics emphasis and with a numerical orientation consider alignment with [computational science](http://en.wikipedia.org/wiki/Computational_science). Both types of departments tend to make efforts to bridge the field educationally if not across all research.

[[edit](http://en.wikipedia.org/w/index.php?title=Computer_science&action=edit&section=14)] **Computer science education**

Some universities teach computer science as a theoretical study of computation and algorithmic reasoning. These programs often feature the [theory of computation](http://en.wikipedia.org/wiki/Theory_of_computation), [analysis of algorithms](http://en.wikipedia.org/wiki/Analysis_of_algorithms), [formal methods](http://en.wikipedia.org/wiki/Formal_methods), [concurrency theory](http://en.wikipedia.org/wiki/Concurrency_(computer_science)), [databases](http://en.wikipedia.org/wiki/Databases), [computer graphics](http://en.wikipedia.org/wiki/Computer_graphics), and [systems analysis](http://en.wikipedia.org/wiki/Systems_analysis), among others. They typically also teach [computer programming](http://en.wikipedia.org/wiki/Computer_programming), but treat it as a vessel for the support of other fields of computer science rather than a central focus of high-level study.

Other colleges and universities, as well as [secondary schools](http://en.wikipedia.org/wiki/Secondary_school) and vocational programs that teach computer science, emphasize the practice of advanced programming rather than the theory of algorithms and computation in their computer science curricula. Such curricula tend to focus on those skills that are important to workers entering the software industry. The practical aspects of computer programming are often referred to as [software engineering](http://en.wikipedia.org/wiki/Software_engineering). However, there is a lot of [disagreement](http://en.wikipedia.org/wiki/Debates_within_software_engineering) over the meaning of the term, and whether or not it is the same thing as programming.

[[edit](http://en.wikipedia.org/w/index.php?title=Computer_science&action=edit&section=15)] See also

|  |  |
| --- | --- |
| [http://upload.wikimedia.org/wikipedia/commons/thumb/a/a8/Office-book.svg/30px-Office-book.svg.png](http://en.wikipedia.org/wiki/Wikipedia:Books) | [Book:Computer science](http://en.wikipedia.org/wiki/Book:Computer_science) |
| [Books](http://en.wikipedia.org/wiki/Wikipedia:Books) are collections of articles that can be downloaded or ordered in print. | |

Main article: [Outline of computer science](http://en.wikipedia.org/wiki/Outline_of_computer_science)

|  |  |
| --- | --- |
| [http://upload.wikimedia.org/wikipedia/commons/thumb/d/d2/Internet_map_1024.jpg/28px-Internet_map_1024.jpg](http://en.wikipedia.org/wiki/File:Internet_map_1024.jpg) | [Computer science portal](http://en.wikipedia.org/wiki/Portal:Computer_science) |
| [http://upload.wikimedia.org/wikipedia/commons/thumb/c/ca/Bus_icon.svg/32px-Bus_icon.svg.png](http://en.wikipedia.org/wiki/File:Bus_icon.svg) | [Computer networking portal](http://en.wikipedia.org/wiki/Portal:Computer_networking) |
| [http://upload.wikimedia.org/wikipedia/commons/thumb/c/c1/Computer-aj_aj_ashton_01.svg/28px-Computer-aj_aj_ashton_01.svg.png](http://en.wikipedia.org/wiki/File:Computer-aj_aj_ashton_01.svg) | [Information technology portal](http://en.wikipedia.org/wiki/Portal:Information_technology) |

[Computer scientist](http://en.wikipedia.org/wiki/Computer_scientist)

[Computing](http://en.wikipedia.org/wiki/Computing)

[Didactics of informatics](http://en.wikipedia.org/w/index.php?title=Didactics_of_informatics&action=edit&redlink=1)

[English in computer science](http://en.wikipedia.org/wiki/English_in_computing)

[History of computer science](http://en.wikipedia.org/wiki/History_of_computer_science)

[Informatics](http://en.wikipedia.org/wiki/Informatics_(academic_field))

[Information and communication technologies for development](http://en.wikipedia.org/wiki/Information_and_communication_technologies_for_development)

[List of academic computer science departments](http://en.wikipedia.org/wiki/List_of_academic_computer_science_departments)

[List of computer science conferences](http://en.wikipedia.org/wiki/List_of_computer_science_conferences)

[List of computer scientists](http://en.wikipedia.org/wiki/List_of_computer_scientists)

[List of open problems in computer science](http://en.wikipedia.org/wiki/Unsolved_problems_in_computer_science)

[List of publications in computer science](http://en.wikipedia.org/wiki/List_of_important_publications_in_computer_science)

[List of pioneers in computer science](http://en.wikipedia.org/wiki/List_of_pioneers_in_computer_science)

[List of software engineering topics](http://en.wikipedia.org/wiki/List_of_software_engineering_topics)

[Philosophy of computer science](http://en.wikipedia.org/wiki/Philosophy_of_computer_science)

[Software engineering](http://en.wikipedia.org/wiki/Software_engineering)

[Women in computing](http://en.wikipedia.org/wiki/Women_in_computing)

[[edit](http://en.wikipedia.org/w/index.php?title=Computer_science&action=edit&section=16)] References

[^](http://en.wikipedia.org/wiki/Computer_science#cite_ref-0) Denning, P. J.; Comer, D. E.; Gries, D.; Mulder, M. C.; Tucker, A.; Turner, A. J.; Young, P. R. (Jan 1989). "Computing as a discipline". Communications of the ACM 32: 9. [doi](http://en.wikipedia.org/wiki/Digital_object_identifier):[10.1145/63238.63239](http://dx.doi.org/10.1145%2F63238.63239).  [edit](http://en.wikipedia.org/w/index.php?title=Template:Cite_doi/10.1145.2F63238.63239&action=edit&editintro=Template:Cite_doi/editintro2) "Computer science and engineering is the systematic study of algorithmic processes-their theory, analysis, design, efficiency, implementation, and application-that describe and transform information."

[^](http://en.wikipedia.org/wiki/Computer_science#cite_ref-1) Wegner, P. (October 13–15, 1976). "Research paradigms in computer science". Proceedings of the 2nd international Conference on Software Engineering. San Francisco, California, United States: IEEE Computer Society Press, Los Alamitos, CA. "Computer science is the study of information structures"

[^](http://en.wikipedia.org/wiki/Computer_science#cite_ref-2) "Computer science is the study of computation." [Computer Science Department, College of Saint Benedict](http://www.csbsju.edu/computerscience/curriculum), Saint John's University

[^](http://en.wikipedia.org/wiki/Computer_science#cite_ref-3) "Computer Science is the study of all aspects of computer systems, from the theoretical foundations to the very practical aspects of managing large software projects." [Massey University](http://study.massey.ac.nz/major.asp?major_code=2010&prog_code=93068)

[^](http://en.wikipedia.org/wiki/Computer_science#cite_ref-4) "Common myths and preconceptions about Cambridge Computer Science" [Computer Science Department](http://www.cl.cam.ac.uk/admissions/undergraduate/myths/), [University of Cambridge](http://en.wikipedia.org/wiki/University_of_Cambridge)

[^](http://en.wikipedia.org/wiki/Computer_science#cite_ref-5) Nigel Tout (2006). ["Calculator Timeline"](http://www.vintagecalculators.com/html/calculator_time-line.html). Vintage Calculator Web Museum. <http://www.vintagecalculators.com/html/calculator_time-line.html>. Retrieved 2006-09-18.

[^](http://en.wikipedia.org/wiki/Computer_science#cite_ref-6) ["Science Museum - Introduction to Babbage"](http://web.archive.org/web/20060908054017/http:/www.sciencemuseum.org.uk/on-line/babbage/index.asp). Archived from [the original](http://www.sciencemuseum.org.uk/on-line/babbage/index.asp) on 2006-09-08. [http://web.archive.org/web/20060908054017/http://www.sciencemuseum.org.uk/on-line/babbage/index.asp](http://web.archive.org/web/20060908054017/http:/www.sciencemuseum.org.uk/on-line/babbage/index.asp). Retrieved 2006-09-24.

[^](http://en.wikipedia.org/wiki/Computer_science#cite_ref-7) ["A Selection and Adaptation From Ada's Notes found in "Ada, The Enchantress of Numbers," by Betty Alexandra Toole Ed.D. Strawberry Press, Mill Valley, CA"](http://www.scottlan.edu/Lriddle/women/ada-love.htm). <http://www.scottlan.edu/Lriddle/women/ada-love.htm>. Retrieved 2006-05-04.

[^](http://en.wikipedia.org/wiki/Computer_science#cite_ref-8) ["IBM Punch Cards in the U.S. Army"](http://www.pattonhq.com/ibm.html). <http://www.pattonhq.com/ibm.html>. Retrieved 2006-09-24.

[^](http://en.wikipedia.org/wiki/Computer_science#cite_ref-9) The [Association for Computing Machinery](http://en.wikipedia.org/wiki/Association_for_Computing_Machinery) (ACM) was founded in 1947.

^ [a](http://en.wikipedia.org/wiki/Computer_science#cite_ref-Denning_cs_discipline_10-0) [b](http://en.wikipedia.org/wiki/Computer_science#cite_ref-Denning_cs_discipline_10-1) [c](http://en.wikipedia.org/wiki/Computer_science#cite_ref-Denning_cs_discipline_10-2) [Denning, P.J.](http://en.wikipedia.org/wiki/Peter_J._Denning) (2000). ["Computer Science: The Discipline"](http://web.archive.org/web/20060525195404/http:/www.idi.ntnu.no/emner/dif8916/denning.pdf) (PDF). Encyclopedia of Computer Science. Archived from [the original](http://www.idi.ntnu.no/emner/dif8916/denning.pdf) on 2006-05-25. [http://web.archive.org/web/20060525195404/http://www.idi.ntnu.no/emner/dif8916/denning.pdf](http://web.archive.org/web/20060525195404/http:/www.idi.ntnu.no/emner/dif8916/denning.pdf).

[^](http://en.wikipedia.org/wiki/Computer_science#cite_ref-11) [CAM.ac.uk](http://www.cl.cam.ac.uk/conference/EDSAC99/statistics.html)

[^](http://en.wikipedia.org/wiki/Computer_science#cite_ref-12) [Computer science pioneer Samuel D. Conte dies at 85](http://www.cs.purdue.edu/feature/conte.html) July 1, 2002

^ [a](http://en.wikipedia.org/wiki/Computer_science#cite_ref-Levy1984_13-0) [b](http://en.wikipedia.org/wiki/Computer_science#cite_ref-Levy1984_13-1) [Levy, Steven](http://en.wikipedia.org/wiki/Steven_Levy) (1984). [Hackers: Heroes of the Computer Revolution](http://en.wikipedia.org/wiki/Hackers:_Heroes_of_the_Computer_Revolution). Doubleday. [ISBN](http://en.wikipedia.org/wiki/International_Standard_Book_Number) [0-385-19195-2](http://en.wikipedia.org/wiki/Special:BookSources/0-385-19195-2).

^ [a](http://en.wikipedia.org/wiki/Computer_science#cite_ref-kahnbook_14-0) [b](http://en.wikipedia.org/wiki/Computer_science#cite_ref-kahnbook_14-1) [David Kahn](http://en.wikipedia.org/wiki/David_Kahn_(writer)), [The Codebreakers](http://en.wikipedia.org/wiki/The_Codebreakers), 1967, [ISBN 0-684-83130-9](http://en.wikipedia.org/wiki/Special:BookSources/0684831309).

^ [a](http://en.wikipedia.org/wiki/Computer_science#cite_ref-bgu_15-0) [b](http://en.wikipedia.org/wiki/Computer_science#cite_ref-bgu_15-1) [Cornell.edu](http://www.cis.cornell.edu/Dean/Presentations/Slides/bgu.pdf)

[^](http://en.wikipedia.org/wiki/Computer_science#cite_ref-16) Constable, R.L. (March 2000) (PDF). [Computer Science: Achievements and Challenges circa 2000](http://www.cs.cornell.edu/cis-dean/bgu.pdf). <http://www.cs.cornell.edu/cis-dean/bgu.pdf>.

[^](http://en.wikipedia.org/wiki/Computer_science#cite_ref-17) [Abelson, H.](http://en.wikipedia.org/wiki/Hal_Abelson); G.J. Sussman with J. Sussman (1996). Structure and Interpretation of Computer Programs (2nd ed.). MIT Press. [ISBN](http://en.wikipedia.org/wiki/International_Standard_Book_Number) [0-262-01153-0](http://en.wikipedia.org/wiki/Special:BookSources/0-262-01153-0). "The computer revolution is a revolution in the way we think and in the way we express what we think. The essence of this change is the emergence of what might best be called procedural epistemology — the study of the structure of knowledge from an imperative point of view, as opposed to the more declarative point of view taken by classical mathematical subjects."

[^](http://en.wikipedia.org/wiki/Computer_science#cite_ref-18) [Black box traders are on the march](http://www.telegraph.co.uk/money/main.jhtml?xml=/money/2006/08/27/ccsoft27.xml) The Telegraph, August 26, 2006

^ [a](http://en.wikipedia.org/wiki/Computer_science#cite_ref-CSAB1997_19-0) [b](http://en.wikipedia.org/wiki/Computer_science#cite_ref-CSAB1997_19-1) Computing Sciences Accreditation Board (28 May 1997). ["Computer Science as a Profession"](http://web.archive.org/web/20080617030847/http:/www.csab.org/comp_sci_profession.html). Archived from [the original](http://www.csab.org/comp_sci_profession.html) on 2008-06-17. [http://web.archive.org/web/20080617030847/http://www.csab.org/comp\_sci\_profession.html](http://web.archive.org/web/20080617030847/http:/www.csab.org/comp_sci_profession.html). Retrieved 2010-05-23.

[^](http://en.wikipedia.org/wiki/Computer_science#cite_ref-20) Committee on the Fundamentals of Computer Science: Challenges and Opportunities, National Research Council (2004). [Computer Science: Reflections on the Field, Reflections from the Field](http://www.nap.edu/catalog.php?record_id=11106#toc). National Academies Press. [ISBN](http://en.wikipedia.org/wiki/International_Standard_Book_Number) [978-0-309-09301-9](http://en.wikipedia.org/wiki/Special:BookSources/978-0-309-09301-9). <http://www.nap.edu/catalog.php?record_id=11106#toc>.

[^](http://en.wikipedia.org/wiki/Computer_science#cite_ref-21) [CSAB, Inc.](http://www.csab.org/)

[^](http://en.wikipedia.org/wiki/Computer_science#cite_ref-22) [Clay Mathematics Institute](http://www.claymath.org/millennium/P_vs_NP/) P=NP

[^](http://en.wikipedia.org/wiki/Computer_science#cite_ref-23) <http://www.pcguide.com/ref/cpu/arch/extSMP-c.html>

[^](http://en.wikipedia.org/wiki/Computer_science#cite_ref-24) Eden, A. H. (2007). ["Three Paradigms of Computer Science"](http://www.eden-study.org/articles/2007/three_paradigms_of_computer_science.pdf). [Minds and Machines](http://en.wikipedia.org/wiki/Minds_and_Machines) 17: 135–167. [doi](http://en.wikipedia.org/wiki/Digital_object_identifier):[10.1007/s11023-007-9060-8](http://dx.doi.org/10.1007%2Fs11023-007-9060-8). <http://www.eden-study.org/articles/2007/three_paradigms_of_computer_science.pdf>.  [edit](http://en.wikipedia.org/w/index.php?title=Template:Cite_doi/10.1007.2Fs11023-007-9060-8&action=edit&editintro=Template:Cite_doi/editintro2)

[^](http://en.wikipedia.org/wiki/Computer_science#cite_ref-25) Communications of the ACM 1(4):p.6

[^](http://en.wikipedia.org/wiki/Computer_science#cite_ref-26) Communications of the ACM 2(1):p.4

[^](http://en.wikipedia.org/wiki/Computer_science#cite_ref-27) IEEE Computer 28(12):p.136

[^](http://en.wikipedia.org/wiki/Computer_science#cite_ref-28) Parnas, D. L. (1998). Annals of Software Engineering 6: 19–37. [doi](http://en.wikipedia.org/wiki/Digital_object_identifier):[10.1023/A:1018949113292](http://dx.doi.org/10.1023%2FA%3A1018949113292).  [edit](http://en.wikipedia.org/w/index.php?title=Template:Cite_doi/10.1023.2FA:1018949113292&action=edit&editintro=Template:Cite_doi/editintro2), p. 19: "Rather than treat software engineering as a subfield of computer science, I treat it as an element of the set, Civil Engineering, Mechanical Engineering, Chemical Engineering, Electrical Engineering, [...]"

[[edit](http://en.wikipedia.org/w/index.php?title=Computer_science&action=edit&section=17)] Further reading

[Association for Computing Machinery](http://en.wikipedia.org/wiki/Association_for_Computing_Machinery). [1998 ACM Computing Classification System](http://www.acm.org/class/1998/overview.html). 1998.

Peter J. Denning. [Is computer science science?](http://portal.acm.org/citation.cfm?id=1053309&coll=&dl=ACM&CFID=15151515&CFTOKEN=6184618), Communications of the ACM, April 2005.

Peter J. Denning, [Great principles in computing curricula](http://portal.acm.org/citation.cfm?id=971303&dl=ACM&coll=&CFID=15151515&CFTOKEN=6184618), Technical Symposium on Computer Science Education, 2004.

[Donald E. Knuth](http://en.wikipedia.org/wiki/Donald_E._Knuth). Selected Papers on Computer Science, CSLI Publications, [Cambridge University Press](http://en.wikipedia.org/wiki/Cambridge_University_Press), 1996.

Joint Task Force of Association for Computing Machinery (ACM), [Association for Information Systems](http://en.wikipedia.org/wiki/Association_for_Information_Systems) (AIS) and [IEEE Computer Society](http://en.wikipedia.org/wiki/IEEE_Computer_Society) (IEEE-CS). [Computing Curricula 2005: The Overview Report](http://www.acm.org/education/curric_vols/CC2005-March06Final.pdf). September 30, 2005.

[List of computer science conferences](http://en.wikipedia.org/wiki/List_of_computer_science_conferences). A ranked list of CS conferences.

[[edit](http://en.wikipedia.org/w/index.php?title=Computer_science&action=edit&section=18)] External links

|  |  |
| --- | --- |
| Find more about Computer science on Wikipedia's [sister projects](http://en.wikipedia.org/wiki/Wikipedia:Wikimedia_sister_projects): | |
| [http://upload.wikimedia.org/wikipedia/commons/thumb/f/f8/Wiktionary-logo-en.svg/23px-Wiktionary-logo-en.svg.png](http://en.wiktionary.org/wiki/Special:Search/Computer_science) | [Definitions](http://en.wiktionary.org/wiki/Special:Search/Computer_science) from Wiktionary |
| [http://upload.wikimedia.org/wikipedia/commons/thumb/4/4a/Commons-logo.svg/18px-Commons-logo.svg.png](http://commons.wikimedia.org/wiki/Special:Search/Computer_science) | [Images and media](http://commons.wikimedia.org/wiki/Special:Search/Computer_science) from Commons |
| [http://upload.wikimedia.org/wikipedia/commons/thumb/1/1b/Wikiversity-logo-en.svg/25px-Wikiversity-logo-en.svg.png](http://en.wikiversity.org/wiki/Special:Search/Computer_science) | [Learning resources](http://en.wikiversity.org/wiki/Special:Search/Computer_science) from Wikiversity |
| [http://upload.wikimedia.org/wikipedia/commons/thumb/2/24/Wikinews-logo.svg/25px-Wikinews-logo.svg.png](http://en.wikinews.org/wiki/Special:Search/Computer_science) | [News stories](http://en.wikinews.org/wiki/Special:Search/Computer_science) from Wikinews |
| [http://upload.wikimedia.org/wikipedia/commons/thumb/f/fa/Wikiquote-logo.svg/21px-Wikiquote-logo.svg.png](http://en.wikiquote.org/wiki/Special:Search/Computer_science) | [Quotations](http://en.wikiquote.org/wiki/Special:Search/Computer_science) from Wikiquote |
| [http://upload.wikimedia.org/wikipedia/commons/thumb/4/4c/Wikisource-logo.svg/24px-Wikisource-logo.svg.png](http://en.wikisource.org/wiki/Special:Search/Computer_science) | [Source texts](http://en.wikisource.org/wiki/Special:Search/Computer_science) from Wikisource |
| [http://upload.wikimedia.org/wikipedia/commons/thumb/f/fa/Wikibooks-logo.svg/25px-Wikibooks-logo.svg.png](http://en.wikibooks.org/wiki/Special:Search/Computer_science) | [Textbooks](http://en.wikibooks.org/wiki/Special:Search/Computer_science) from Wikibooks |

Lectures and bibliographies

[Computer science](http://www.dmoz.org/Computers/Computer_Science/) at the [Open Directory Project](http://en.wikipedia.org/wiki/Open_Directory_Project)

[Directory of free university lectures in Computer Science](http://www.lecturefox.com/computerscience/)

[Collection of computer science lectures](http://videolectures.net/Top/Computer_Science/)

[bibliography/ Collection of Computer Science Bibliographies](http://liinwww.ira.uka.de/)

[CS Directory and resources](http://csdir.org/)

[Photographs of computer scientists](http://se.ethz.ch/~meyer/gallery/) ([Bertrand Meyer](http://en.wikipedia.org/wiki/Bertrand_Meyer)'s gallery)

Webcasts

[UCLA Computer Science 1 Freshman Computer Science Seminar Section 1](http://www.oid.ucla.edu/webcasts/courses/2006-2007/2006fall/cs1)

[Berkeley Introduction to Computers](http://webcast.berkeley.edu/course_details.php?seriesid=1906978395)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | [[hide](http://en.wikipedia.org/wiki/Computer_science)][v](http://en.wikipedia.org/wiki/Template:Computer_Science) · [d](http://en.wikipedia.org/wiki/Template_talk:Computer_Science) · [e](http://en.wikipedia.org/w/index.php?title=Template:Computer_Science&action=edit)Major fields of computer science | | |  |  | | [Mathematical foundations](http://en.wikipedia.org/wiki/Mathematics) | [Mathematical logic](http://en.wikipedia.org/wiki/Mathematical_logic) · [Set theory](http://en.wikipedia.org/wiki/Set_theory) · [Number theory](http://en.wikipedia.org/wiki/Number_theory) · [Graph theory](http://en.wikipedia.org/wiki/Graph_theory) · [Type theory](http://en.wikipedia.org/wiki/Type_theory) · [Category theory](http://en.wikipedia.org/wiki/Category_theory) · [Numerical analysis](http://en.wikipedia.org/wiki/Numerical_analysis)  · [Information theory](http://en.wikipedia.org/wiki/Information_theory) | |  |  | | [Theory of computation](http://en.wikipedia.org/wiki/Theory_of_computation) | [Automata theory](http://en.wikipedia.org/wiki/Automata_theory) · [Computability theory](http://en.wikipedia.org/wiki/Computability_theory) · [Computational complexity theory](http://en.wikipedia.org/wiki/Computational_complexity_theory) · [Quantum computing theory](http://en.wikipedia.org/wiki/Quantum_computer) | |  |  | | [Algorithms](http://en.wikipedia.org/wiki/Algorithm) and [data structures](http://en.wikipedia.org/wiki/Data_structure) | [Analysis of algorithms](http://en.wikipedia.org/wiki/Analysis_of_algorithms) · [Algorithm design](http://en.wikipedia.org/wiki/Algorithm_design) · [Computational geometry](http://en.wikipedia.org/wiki/Computational_geometry) | |  |  | | [Programming languages](http://en.wikipedia.org/wiki/Programming_language) and [Compilers](http://en.wikipedia.org/wiki/Compiler) | [Parsers](http://en.wikipedia.org/wiki/Parsing) · [Interpreters](http://en.wikipedia.org/wiki/Interpreter_(computing)) · [Procedural programming](http://en.wikipedia.org/wiki/Procedural_programming) · [Object-oriented programming](http://en.wikipedia.org/wiki/Object-oriented_programming) · [Functional programming](http://en.wikipedia.org/wiki/Functional_programming) · [Logic programming](http://en.wikipedia.org/wiki/Logic_programming) · [Programming paradigms](http://en.wikipedia.org/wiki/Programming_paradigm) | |  |  | | [Concurrent](http://en.wikipedia.org/wiki/Concurrency_(computer_science)), [Parallel](http://en.wikipedia.org/wiki/Parallel_computing), and [Distributed](http://en.wikipedia.org/wiki/Distributed_computing) systems | [Multiprocessing](http://en.wikipedia.org/wiki/Multiprocessing) · [Grid computing](http://en.wikipedia.org/wiki/Grid_computing) · [Concurrency control](http://en.wikipedia.org/wiki/Concurrency_control) | |  |  | | [Software engineering](http://en.wikipedia.org/wiki/Software_engineering) | [Requirements analysis](http://en.wikipedia.org/wiki/Requirements_analysis) · [Software design](http://en.wikipedia.org/wiki/Software_design) · [Computer programming](http://en.wikipedia.org/wiki/Computer_programming) · [Formal methods](http://en.wikipedia.org/wiki/Formal_methods) · [Software testing](http://en.wikipedia.org/wiki/Software_testing) · [Software development process](http://en.wikipedia.org/wiki/Software_development_process) | |  |  | | [System architecture](http://en.wikipedia.org/wiki/Systems_architecture) | [Computer architecture](http://en.wikipedia.org/wiki/Computer_architecture) · [Computer organization](http://en.wikipedia.org/wiki/Microarchitecture) · [Operating systems](http://en.wikipedia.org/wiki/Operating_system) | |  |  | | [Telecommunication](http://en.wikipedia.org/wiki/Telecommunication) & [Networking](http://en.wikipedia.org/wiki/Computer_networking) | [Computer audio](http://en.wikipedia.org/wiki/Computer_music) · [Routing](http://en.wikipedia.org/wiki/Routing) · [Network topology](http://en.wikipedia.org/wiki/Network_topology) · [Cryptography](http://en.wikipedia.org/wiki/Cryptography) | |  |  | | [Databases](http://en.wikipedia.org/wiki/Database) | [Database management systems](http://en.wikipedia.org/wiki/Database_management_system) · [Relational databases](http://en.wikipedia.org/wiki/Relational_database) · [SQL](http://en.wikipedia.org/wiki/SQL) · [Transactions](http://en.wikipedia.org/wiki/Transaction_processing) · [Database indexes](http://en.wikipedia.org/wiki/Index_(database)) · [Data mining](http://en.wikipedia.org/wiki/Data_mining) | |  |  | | [Artificial intelligence](http://en.wikipedia.org/wiki/Artificial_intelligence) | [Automated reasoning](http://en.wikipedia.org/wiki/Automated_reasoning) · [Computational linguistics](http://en.wikipedia.org/wiki/Computational_linguistics) · [Computer vision](http://en.wikipedia.org/wiki/Computer_vision) · [Evolutionary computation](http://en.wikipedia.org/wiki/Evolutionary_computation) · [Expert systems](http://en.wikipedia.org/wiki/Expert_system) · [Machine learning](http://en.wikipedia.org/wiki/Machine_learning) · [Natural language processing](http://en.wikipedia.org/wiki/Natural_language_processing) · [Robotics](http://en.wikipedia.org/wiki/Robotics) | |  |  | | [Computer graphics](http://en.wikipedia.org/wiki/Computer_graphics) | [Visualization](http://en.wikipedia.org/wiki/Visualization_(computer_graphics)) · [Computer animation](http://en.wikipedia.org/wiki/Computer_animation) · [Image processing](http://en.wikipedia.org/wiki/Image_processing) | |  |  | | [Human–computer interaction](http://en.wikipedia.org/wiki/Human%E2%80%93computer_interaction) | [Computer accessibility](http://en.wikipedia.org/wiki/Computer_accessibility) · [User interfaces](http://en.wikipedia.org/wiki/User_interface) · [Wearable computing](http://en.wikipedia.org/wiki/Wearable_computer) · [Ubiquitous computing](http://en.wikipedia.org/wiki/Ubiquitous_computing) · [Virtual reality](http://en.wikipedia.org/wiki/Virtual_reality) | |  |  | | [Scientific computing](http://en.wikipedia.org/wiki/Computational_science) | [Artificial life](http://en.wikipedia.org/wiki/Artificial_life) · [Bioinformatics](http://en.wikipedia.org/wiki/Bioinformatics) · [Cognitive science](http://en.wikipedia.org/wiki/Cognitive_science) · [Computational chemistry](http://en.wikipedia.org/wiki/Computational_chemistry) · [Computational neuroscience](http://en.wikipedia.org/wiki/Computational_neuroscience) · [Computational physics](http://en.wikipedia.org/wiki/Computational_physics) · [Numerical algorithms](http://en.wikipedia.org/wiki/Numerical_analysis) · [Symbolic mathematics](http://en.wikipedia.org/wiki/Symbolic_computation) | |  |  | | Note: Computer science can also be split up into different topics or fields according to the [ACM Computing Classification System](http://en.wikipedia.org/wiki/ACM_Computing_Classification_System). | | |

Retrieved from "<http://en.wikipedia.org/wiki/Computer_science>"

[Categories](http://en.wikipedia.org/wiki/Special:Categories): [Computer science](http://en.wikipedia.org/wiki/Category:Computer_science)

Hidden categories: [Articles to be expanded from June 2008](http://en.wikipedia.org/wiki/Category:Articles_to_be_expanded_from_June_2008) | [All articles to be expanded](http://en.wikipedia.org/wiki/Category:All_articles_to_be_expanded) | [All articles with unsourced statements](http://en.wikipedia.org/wiki/Category:All_articles_with_unsourced_statements) | [Articles with unsourced statements from October 2010](http://en.wikipedia.org/wiki/Category:Articles_with_unsourced_statements_from_October_2010)

This page was last modified on 5 January 2011 at 01:09.

Text is available under the [Creative Commons Attribution-ShareAlike License](http://en.wikipedia.org/wiki/Wikipedia:Text_of_Creative_Commons_Attribution-ShareAlike_3.0_Unported_License); additional terms may apply. See [Terms of Use](http://wikimediafoundation.org/wiki/Terms_of_Use) for details.  
Wikipedia® is a registered trademark of the [Wikimedia Foundation, Inc.](http://www.wikimediafoundation.org/), a non-profit organization.

[Page notice](http://en.wikipedia.org/w/index.php?title=Template:Editnotices/Page/Computer_science&action=edit&redlink=1)

|  |  |
| --- | --- |
|  | You are not currently [logged in](http://en.wikipedia.org/wiki/Special:UserLogin). If you save any edits, your [IP address](http://en.wikipedia.org/wiki/IP_address) will be [recorded publicly](http://en.wikipedia.org/wiki/Help:Page_history) in this page's [edit history](http://en.wikipedia.org/w/index.php?title=Computer_science&action=history). If you [create an account](http://en.wikipedia.org/w/index.php?title=Special:UserLogin&type=signup), you can conceal your IP address and be provided with many other [benefits](http://en.wikipedia.org/wiki/Wikipedia:Why_create_an_account%3F). Messages sent to your IP can be viewed on [your talk page](http://en.wikipedia.org/wiki/Special:MyTalk).  Please do not save test edits. If you want to experiment, please use the [sandbox](http://en.wikipedia.org/wiki/Wikipedia:Sandbox). If you need any help getting started with editing, see the [Wikipedia:new contributors' help page](http://en.wikipedia.org/wiki/Wikipedia:New_contributors%27_help_page). |

**Relationship with other fields ==**

Despite its name, a significant amount of computer science does not involve the study of computers themselves. Because of this, several alternative names have been proposed. Certain departments of major universities prefer the term ''computing science'', to emphasize precisely that difference. Danish scientist [[Peter Naur]] suggested the term ''datalogy'', to reflect the fact that the scientific discipline revolves around data and data treatment, while not necessarily involving computers. The first scientific institution to use the term was the Department of Datalogy at the University of Copenhagen, founded in 1969, with Peter Naur being the first professor in datalogy. The term is used mainly in the Scandinavian countries. Also, in the early days of computing, a number of terms for the practitioners of the field of computing were suggested in the ''Communications of the ACM'' – ''turingineer'', ''turologist'', ''flow-charts-man'', ''applied meta-mathematician'', and ''applied epistemologist''.<ref>Communications of the ACM 1(4):p.6</ref> Three months later in the same journal, ''comptologist'' was suggested, followed next year by ''hypologist''.<ref>Communications of the ACM 2(1):p.4</ref> The term ''computics'' has also been suggested.<ref>IEEE Computer 28(12):p.136</ref> In continental Europe, names such as ''informatique'' (French), ''Informatik'' (German) or ''informatica'' (Dutch), derived from information and possibly mathematics or automatic, are more common than names derived from computer/computation.

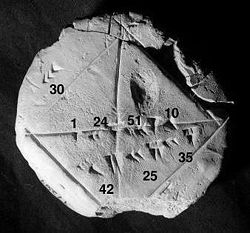
The renowned computer scientist [[Edsger W. Dijkstra|Edsger Dijkstra]] stated, "Computer science is no more about computers than astronomy is about telescopes." The design and deployment of computers and computer systems is generally considered the province of disciplines other than computer science. For example, the study of [[computer hardware]] is usually considered part of [[computer engineering]], while the study of commercial [[computer system]]s and their deployment is often called [[information technology]] or [[information systems]]. However, there has been much cross-fertilization of ideas between the various computer-related disciplines. Computer science research has also often crossed into other disciplines, such as [[philosophy]], [[cognitive science]], [[linguistics]], [[mathematics]], [[physics]], [[computational statistics|statistics]], and [[computational economics|economics]].

Computer science is considered by some to have a much closer relationship with mathematics than many scientific disciplines, with some observers saying that computing is a mathematical science.<ref name="Denning\_cs\_discipline" /> Early computer science was strongly influenced by the work of mathematicians such as [[Kurt Gödel]] and [[Alan Turing]], and there continues to be a useful interchange of ideas between the two fields in areas such as [[mathematical logic]], [[category theory]], [[domain theory]], and [[algebra]].

The relationship between computer science and [[software engineering]] is a contentious issue, which is further muddied by [[Debates within software engineering|disputes]] over what the term "software engineering" means, and how computer science is defined. [[David Parnas]], taking a cue from the relationship between other engineering and science disciplines, has claimed that the principal focus of computer science is studying the properties of computation in general, while the principal focus of software engineering is the design of specific computations to achieve practical goals, making the two separate but complementary disciplines.<ref>{{Cite doi|10.1023/A:1018949113292}}, p. 19: "Rather than treat software engineering as a subfield of computer science, I treat it as an element of the set, Civil Engineering, Mechanical Engineering, Chemical Engineering, Electrical Engineering, [...]"</ref>

The academic, political, and funding aspects of computer science tend to depend on whether a department formed with a mathematical emphasis or with an engineering emphasis. Computer science departments with a mathematics emphasis and with a numerical orientation consider alignment with [[computational science]]. Both types of departments tend to make efforts to bridge the field educationally if not across all research.

**Numerical analysis**

[](http://en.wikipedia.org/wiki/File:Ybc7289-bw.jpg)

Babylonian clay tablet BC 7289 (c. 1800–1600 BC) with annotations. The approximation of the [square root of 2](http://en.wikipedia.org/wiki/Square_root_of_2) is four [sexagesimal](http://en.wikipedia.org/wiki/Sexagesimal) figures, which is about six [decimal](http://en.wikipedia.org/wiki/Decimal) figures. 1 + 24/60 + 51/602 + 10/603 = 1.41421296...[[1]](http://en.wikipedia.org/wiki/Numerical_analysis#cite_note-0)

Numerical analysis is the study of [algorithms](http://en.wikipedia.org/wiki/Algorithm) that use numerical [approximation](http://en.wikipedia.org/wiki/Approximation) (as opposed to general [symbolic manipulations](http://en.wikipedia.org/wiki/Symbolic_computation)) for the problems of [mathematical analysis](http://en.wikipedia.org/wiki/Mathematical_analysis) (as distinguished from [discrete mathematics](http://en.wikipedia.org/wiki/Discrete_mathematics)).

One of the earliest mathematical writings is the Babylonian tablet BC 7289, which gives a sexagesimal numerical approximation of , the length of the diagonal in a unit square. Being able to compute the sides of a triangle (and hence, being able to compute square roots) is extremely important, for instance, in carpentry and construction.[[2]](http://en.wikipedia.org/wiki/Numerical_analysis#cite_note-1)

Numerical analysis continues this long tradition of practical mathematical calculations. Much like the Babylonian approximation of , modern numerical analysis does not seek exact answers, because exact answers are often impossible to obtain in practice. Instead, much of numerical analysis is concerned with obtaining approximate solutions while maintaining reasonable bounds on errors.

Numerical analysis naturally finds applications in all fields of engineering and the physical sciences, but in the 21st century, the life sciences and even the arts have adopted elements of scientific computations. [Ordinary differential equations](http://en.wikipedia.org/wiki/Ordinary_differential_equation) appear in the [movement of heavenly bodies (planets, stars and galaxies)](http://en.wikipedia.org/wiki/Celestial_mechanics); [optimization](http://en.wikipedia.org/wiki/Optimization_(mathematics)) occurs in portfolio management; [numerical linear algebra](http://en.wikipedia.org/wiki/Numerical_linear_algebra) is important for data analysis; [stochastic differential equations](http://en.wikipedia.org/wiki/Stochastic_differential_equation) and [Markov chains](http://en.wikipedia.org/wiki/Markov_chain) are essential in simulating living cells for medicine and biology.

Before the advent of modern computers numerical methods often depended on hand [interpolation](http://en.wikipedia.org/wiki/Interpolation) in large printed tables. Since the mid 20th century, computers calculate the required functions instead. The interpolation [algorithms](http://en.wikipedia.org/wiki/Algorithms) nevertheless may be used as part of the software for solving [differential equations](http://en.wikipedia.org/wiki/Differential_equations).

|  |
| --- |
| Contents  [[hide](http://en.wikipedia.org/wiki/Numerical_analysis)]  [1 General introduction](http://en.wikipedia.org/wiki/Numerical_analysis#General_introduction)  [1.1 History](http://en.wikipedia.org/wiki/Numerical_analysis#History)  [1.2 Direct and iterative methods](http://en.wikipedia.org/wiki/Numerical_analysis#Direct_and_iterative_methods)  [1.2.1 Discretization and numerical integration](http://en.wikipedia.org/wiki/Numerical_analysis#Discretization_and_numerical_integration)  [1.3 Discretization](http://en.wikipedia.org/wiki/Numerical_analysis#Discretization)  [2 The generation and propagation of errors](http://en.wikipedia.org/wiki/Numerical_analysis#The_generation_and_propagation_of_errors)  [2.1 Round-off](http://en.wikipedia.org/wiki/Numerical_analysis#Round-off)  [2.2 Truncation and discretization error](http://en.wikipedia.org/wiki/Numerical_analysis#Truncation_and_discretization_error)  [2.3 Numerical stability and well-posed problems](http://en.wikipedia.org/wiki/Numerical_analysis#Numerical_stability_and_well-posed_problems)  [3 Areas of study](http://en.wikipedia.org/wiki/Numerical_analysis#Areas_of_study)  [3.1 Computing values of functions](http://en.wikipedia.org/wiki/Numerical_analysis#Computing_values_of_functions)  [3.2 Interpolation, extrapolation, and regression](http://en.wikipedia.org/wiki/Numerical_analysis#Interpolation.2C_extrapolation.2C_and_regression)  [3.3 Solving equations and systems of equations](http://en.wikipedia.org/wiki/Numerical_analysis#Solving_equations_and_systems_of_equations)  [3.4 Solving eigenvalue or singular value problems](http://en.wikipedia.org/wiki/Numerical_analysis#Solving_eigenvalue_or_singular_value_problems)  [3.5 Optimization](http://en.wikipedia.org/wiki/Numerical_analysis#Optimization)  [3.6 Evaluating integrals](http://en.wikipedia.org/wiki/Numerical_analysis#Evaluating_integrals)  [3.7 Differential equations](http://en.wikipedia.org/wiki/Numerical_analysis#Differential_equations)  [4 Software](http://en.wikipedia.org/wiki/Numerical_analysis#Software)  [5 See also](http://en.wikipedia.org/wiki/Numerical_analysis#See_also)  [6 Notes](http://en.wikipedia.org/wiki/Numerical_analysis#Notes)  [7 References](http://en.wikipedia.org/wiki/Numerical_analysis#References)  [8 External links](http://en.wikipedia.org/wiki/Numerical_analysis#External_links) |

[[edit](http://en.wikipedia.org/w/index.php?title=Numerical_analysis&action=edit&section=1)] **General introduction**

The overall goal of the field of numerical analysis is the design and analysis of techniques to give approximate but accurate solutions to hard problems, the variety of which is suggested by the following.

Advanced numerical methods are essential in making [numerical weather prediction](http://en.wikipedia.org/wiki/Numerical_weather_prediction) feasible.

Computing the trajectory of a spacecraft requires the accurate numerical solution of a system of [ordinary differential equations](http://en.wikipedia.org/wiki/Ordinary_differential_equation).

Car companies can improve the crash safety of their vehicles by using computer simulations of car crashes. Such simulations essentially consist of solving [partial differential equations](http://en.wikipedia.org/wiki/Partial_differential_equation) numerically.

[Hedge funds](http://en.wikipedia.org/wiki/Hedge_fund) (private investment funds) use tools from all fields of numerical analysis to calculate the value of stocks and derivatives more precisely than other market participants.

Airlines use sophisticated optimization algorithms to decide ticket prices, airplane and crew assignments and fuel needs. This field is also called [operations research](http://en.wikipedia.org/wiki/Operations_research).

Insurance companies use numerical programs for [actuarial](http://en.wikipedia.org/wiki/Actuary) analysis.

The rest of this section outlines several important themes of numerical analysis.

[[edit](http://en.wikipedia.org/w/index.php?title=Numerical_analysis&action=edit&section=2)] **History**

The field of numerical analysis predates the invention of modern computers by many centuries. [Linear interpolation](http://en.wikipedia.org/wiki/Linear_interpolation) was already in use more than 2000 years ago. Many great mathematicians of the past were preoccupied by numerical analysis, as is obvious from the names of important algorithms like [Newton's method](http://en.wikipedia.org/wiki/Newton%27s_method), [Lagrange interpolation polynomial](http://en.wikipedia.org/wiki/Lagrange_polynomial), [Gaussian elimination](http://en.wikipedia.org/wiki/Gaussian_elimination), or [Euler's method](http://en.wikipedia.org/wiki/Euler%27s_method).

To facilitate computations by hand, large books were produced with formulas and tables of data such as interpolation points and function coefficients. Using these tables, often calculated out to 16 decimal places or more for some functions, one could look up values to plug into the formulas given and achieve very good numerical estimates of some functions. The canonical work in the field is the [NIST](http://en.wikipedia.org/wiki/NIST) publication edited by [Abramowitz and Stegun](http://en.wikipedia.org/wiki/Abramowitz_and_Stegun), a 1000-plus page book of a very large number of commonly used formulas and functions and their values at many points. The function values are no longer very useful when a computer is available, but the large listing of formulas can still be very handy.

The [mechanical calculator](http://en.wikipedia.org/wiki/Mechanical_calculator) was also developed as a tool for hand computation. These calculators evolved into electronic computers in the 1940s, and it was then found that these computers were also useful for administrative purposes. But the invention of the computer also influenced the field of numerical analysis, since now longer and more complicated calculations could be done.

[[edit](http://en.wikipedia.org/w/index.php?title=Numerical_analysis&action=edit&section=3)] **Direct and iterative methods**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Direct vs iterative methods  Consider the problem of solving 3x3+4=28 for the unknown quantity x.   |  |  | | --- | --- | | Direct method | | |  | 3x3 + 4 = 28. | | Subtract 4 | 3x3 = 24. | | Divide by 3 | x3 = 8. | | Take cube roots | x = 2. |   For the iterative method, apply the [bisection method](http://en.wikipedia.org/wiki/Bisection_method) to f(x) = 3x3 - 24. The initial values are a = 0, b = 3, f(a) = -24, f(b) = 57.   |  |  |  |  | | --- | --- | --- | --- | | Iterative method | | | | | a | b | mid | f(mid) | | 0 | 3 | 1.5 | -13.875 | | 1.5 | 3 | 2.25 | 10.17... | | 1.5 | 2.25 | 1.875 | -4.22... | | 1.875 | 2.25 | 2.0625 | 2.32... |   We conclude from this table that the solution is between 1.875 and 2.0625. The algorithm might return any number in that range with an error less than 0.2.  [[edit](http://en.wikipedia.org/w/index.php?title=Numerical_analysis&action=edit&section=4)] Discretization and numerical integration  [Schumacher (Ferrari) in practice at USGP 2005.jpg](http://en.wikipedia.org/wiki/File:Schumacher_(Ferrari)_in_practice_at_USGP_2005.jpg)  In a two hour race, we have measured the speed of the car at three instants and recorded them in the following table.   |  |  |  |  | | --- | --- | --- | --- | | Time | 0:20 | 1:00 | 1:40 | | km/h | 140 | 150 | 180 |   A **discretization** would be to say that the speed of the car was constant from 0:00 to 0:40, then from 0:40 to 1:20 and finally from 1:20 to 2:00. For instance, the total distance traveled in the first 40 minutes is approximately (2/3h x 140 km/h)=93.3 km. This would allow us to estimate the total distance traveled as 93.3 km + 100 km + 120 km = 313.3 km, which is an example of numerical integration (see below) using a [Riemann sum](http://en.wikipedia.org/wiki/Riemann_sum), because displacement is the [integral](http://en.wikipedia.org/wiki/Integral) of velocity.  Ill posed problem: Take the function f(x) = 1/(x − 1). Note that f(1.1) = 10 and f(1.001) = 1000: a change in x of less than 0.1 turns into a change in f(x) of nearly 1000. Evaluating f(x) near x = 1 is an ill-conditioned problem.  Well-posed problem: By contrast, the function is continuous and so evaluating it is well-posed, at least for x being not close to zero. |

**Direct methods** compute the solution to a problem in a finite number of steps. These methods would give the precise answer if they were performed in [infinite precision arithmetic](http://en.wikipedia.org/wiki/Computer_numbering_formats). Examples include [Gaussian elimination](http://en.wikipedia.org/wiki/Gaussian_elimination), the [QR](http://en.wikipedia.org/wiki/QR_algorithm) factorization method for solving [systems of linear equations](http://en.wikipedia.org/wiki/System_of_linear_equations), and the [simplex method](http://en.wikipedia.org/wiki/Simplex_method) of [linear programming](http://en.wikipedia.org/wiki/Linear_programming). In practice, [finite precision](http://en.wikipedia.org/wiki/Floating_point) is used and the result is an approximation of the true solution (assuming [stability](http://en.wikipedia.org/wiki/Numerically_stable)).

In contrast to direct methods, [iterative methods](http://en.wikipedia.org/wiki/Iterative_method) are not expected to terminate in a number of steps. Starting from an initial guess, iterative methods form successive approximations that [converge](http://en.wikipedia.org/wiki/Limit_of_a_sequence) to the exact solution only in the limit. A [convergence test](http://en.wikipedia.org/wiki/Convergence_test) is specified in order to decide when a sufficiently accurate solution has (hopefully) been found. Even using infinite precision arithmetic these methods would not reach the solution within a finite number of steps (in general). Examples include [Newton's method](http://en.wikipedia.org/wiki/Newton%27s_method), the [bisection method](http://en.wikipedia.org/wiki/Bisection_method), and [Jacobi iteration](http://en.wikipedia.org/wiki/Jacobi_iteration). In computational matrix algebra, iterative methods are generally needed for large problems.

Iterative methods are more common than direct methods in numerical analysis. Some methods are direct in principle but are usually used as though they were not, e.g. [GMRES](http://en.wikipedia.org/wiki/GMRES) and the [conjugate gradient method](http://en.wikipedia.org/wiki/Conjugate_gradient_method). For these methods the number of steps needed to obtain the exact solution is so large that an approximation is accepted in the same manner as for an iterative method.

[[edit](http://en.wikipedia.org/w/index.php?title=Numerical_analysis&action=edit&section=5)] **Discretization**

Furthermore, continuous problems must sometimes be replaced by a discrete problem whose solution is known to approximate that of the continuous problem; this process is called [discretization](http://en.wikipedia.org/wiki/Discretization). For example, the solution of a [differential equation](http://en.wikipedia.org/wiki/Differential_equation) is a function. This function must be represented by a finite amount of data, for instance by its value at a finite number of points at its domain, even though this domain is a continuum.

[[edit](http://en.wikipedia.org/w/index.php?title=Numerical_analysis&action=edit&section=6)] **The generation and propagation of errors**

|  |  |
| --- | --- |
|  | It has been suggested that this article or section be [merged](http://en.wikipedia.org/wiki/Wikipedia:Merging) into [Discretization error](http://en.wikipedia.org/wiki/Discretization_error). ([Discuss](http://en.wikipedia.org/wiki/Talk:Numerical_analysis#Merging_.22Errors.22)) |

The study of errors forms an important part of numerical analysis. There are several ways in which error can be introduced in the solution of the problem.

[[edit](http://en.wikipedia.org/w/index.php?title=Numerical_analysis&action=edit&section=7)] **Round-off**

[Round-off errors](http://en.wikipedia.org/wiki/Round-off_error) arise because it is impossible to represent all [real numbers](http://en.wikipedia.org/wiki/Real_number) exactly on a machine with finite memory (which is what all practical [digital computers](http://en.wikipedia.org/wiki/Digital_computer) are).

[[edit](http://en.wikipedia.org/w/index.php?title=Numerical_analysis&action=edit&section=8)] **Truncation and discretization error**

[Truncation](http://en.wikipedia.org/wiki/Truncation) errors are committed when an iterative method is terminated or a mathematical procedure is approximated, and the approximate solution differs from the exact solution. Similarly, discretization induces a [discretization error](http://en.wikipedia.org/wiki/Discretization_error) because the solution of the discrete problem does not coincide with the solution of the continuous problem. For instance, in the iteration in the sidebar to compute the solution of 3x3 + 4 = 28, after 10 or so iterations, we conclude that the root is roughly 1.99 (for example). We therefore have a truncation error of 0.01.

Once an error is generated, it will generally propagate through the calculation. For instance, we have already noted that the operation + on a calculator (or a computer) is inexact. It follows that a calculation of the type a+b+c+d+e is even more inexact.

What does it mean when we say that the truncation error is created when we approximate a mathematical procedure. We know that to integrate a function exactly requires one to find the sum of infinite trapezoids. But numerically one can find the sum of only finite trapezoids, and hence the approximation of the mathematical procedure. Similarly, to differentiate a function, the differential element approaches to zero but numerically we can only choose a finite value of the differential element.

[[edit](http://en.wikipedia.org/w/index.php?title=Numerical_analysis&action=edit&section=9)] **Numerical stability and well-posed problems**

[Numerical stability](http://en.wikipedia.org/wiki/Numerical_stability) is an important notion in numerical analysis. An algorithm is called numerically stable if an error, whatever its cause, does not grow to be much larger during the calculation. This happens if the problem is [well-conditioned](http://en.wikipedia.org/wiki/Condition_number), meaning that the solution changes by only a small amount if the problem data are changed by a small amount. To the contrary, if a problem is ill-conditioned, then any small error in the data will grow to be a large error.

Both the original problem and the algorithm used to solve that problem can be well-conditioned and/or ill-conditioned, and any combination is possible.

So an algorithm that solves a well-conditioned problem may be either numerically stable or numerically unstable. An art of numerical analysis is to find a stable algorithm for solving a well-posed mathematical problem. For instance, computing the square root of 2 (which is roughly 1.41421) is a well-posed problem. Many algorithms solve this problem by starting with an initial approximation x1 to , for instance x1=1.4, and then computing improved guesses x2, x3, etc.. One such method is the famous [Babylonian method](http://en.wikipedia.org/wiki/Babylonian_method), which is given by xk+1 = xk/2 + 1/xk. Another iteration, which we will call Method X, is given by xk + 1 = (xk2−2)2 + xk.[[3]](http://en.wikipedia.org/wiki/Numerical_analysis#cite_note-2) We have calculated a few iterations of each scheme in table form below, with initial guesses x1 = 1.4 and x1 = 1.42.

|  |  |  |  |
| --- | --- | --- | --- |
| Babylonian | Babylonian | Method X | Method X |
| x1 = 1.4 | x1 = 1.42 | x1 = 1.4 | x1 = 1.42 |
| x2 = 1.4142857... | x2 = 1.41422535... | x2 = 1.4016 | x2 = 1.42026896 |
| x3 = 1.414213564... | x3 = 1.41421356242... | x3 = 1.4028614... | x3 = 1.42056... |
|  |  | ... | ... |
|  |  | x1000000 = 1.41421... | x28 = 7280.2284... |

Observe that the Babylonian method converges fast regardless of the initial guess, whereas Method X converges extremely slowly with initial guess 1.4 and diverges for initial guess 1.42. Hence, the Babylonian method is numerically stable, while Method X is numerically unstable.

Numerical stability is affected by the number of the significant digits the machine keeps on, if we use a machine that keeps on the first four floating-point digits, a good example on loss of significance these two equivalent functions if we compare the results of and by looking to the two above results, we realize that loss of significance which is also called Subtractive Cancelation has a huge effect on the results, even though both functions are equivalent; to show that they are equivalent simply we need to start by f(x) and end with g(x), and so the true value for the result is 11.174755... which is exactly g(500)=11.1748 after rounding the result to 4 decimal digits now imagine that you use tens of terms like these functions in your program, your error will increase as you proceed in the program, unless you use the suitable formula of the two functions each time you evaluate either f(x), or g(x), the choice is dependent on the parity of x .

The example is taken from Mathew; Numerical methods using matlab , 3rd ed.

[[edit](http://en.wikipedia.org/w/index.php?title=Numerical_analysis&action=edit&section=10)] **Areas of study**

The field of numerical analysis is divided into different disciplines according to the problem that is to be solved.

[[edit](http://en.wikipedia.org/w/index.php?title=Numerical_analysis&action=edit&section=11)] **Computing values of functions**

|  |
| --- |
| Interpolation: We have observed the temperature to vary from 20 degrees Celsius at 1:00 to 14 degrees at 3:00. A linear interpolation of this data would conclude that it was 17 degrees at 2:00 and 18.5 degrees at 1:30pm.  **Extrapolation**: If the [gross domestic product](http://en.wikipedia.org/wiki/Gross_domestic_product) of a country has been growing an average of 5% per year and was 100 billion dollars last year, we might extrapolate that it will be 105 billion dollars this year.  Regression: In linear regression, given n points, we compute a line that passes as close as possible to those n points.  [How much for a glass of lemonade?](http://en.wikipedia.org/wiki/File:LemonadeJuly2006.JPG)  **Optimization**: Say you sell lemonade at a [lemonade stand](http://en.wikipedia.org/wiki/Lemonade_stand), and notice that at $1, you can sell 197 glasses of lemonade per day, and that for each increase of $0.01, you will sell one less lemonade per day. If you could charge $1.485, you would maximize your profit, but due to the constraint of having to charge a whole cent amount, charging $1.49 per glass will yield the maximum income of $220.52 per day.  Differential equation: If you set up 100 fans to blow air from one end of the room to the other and then you drop a feather into the wind, what happens? The feather will follow the air currents, which may be very complex. One approximation is to measure the speed at which the air is blowing near the feather every second, and advance the simulated feather as if it were moving in a straight line at that same speed for one second, before measuring the wind speed again. This is called the [Euler method](http://en.wikipedia.org/wiki/Euler_method) for solving an ordinary differential equation. |

One of the simplest problems is the evaluation of a function at a given point. The most straightforward approach, of just plugging in the number in the formula is sometimes not very efficient. For polynomials, a better approach is using the [Horner scheme](http://en.wikipedia.org/wiki/Horner_scheme), since it reduces the necessary number of multiplications and additions. Generally, it is important to estimate and control [round-off errors](http://en.wikipedia.org/wiki/Round-off_error) arising from the use of [floating point](http://en.wikipedia.org/wiki/Floating_point) arithmetic.

[[edit](http://en.wikipedia.org/w/index.php?title=Numerical_analysis&action=edit&section=12)] **Interpolation, extrapolation, and regression**

[Interpolation](http://en.wikipedia.org/wiki/Interpolation) solves the following problem: given the value of some unknown function at a number of points, what value does that function have at some other point between the given points?

[Extrapolation](http://en.wikipedia.org/wiki/Extrapolation) is very similar to interpolation, except that now we want to find the value of the unknown function at a point which is outside the given points.

[Regression](http://en.wikipedia.org/wiki/Regression_analysis) is also similar, but it takes into account that the data is imprecise. Given some points, and a measurement of the value of some function at these points (with an error), we want to determine the unknown function. The [least squares](http://en.wikipedia.org/wiki/Least_squares)-method is one popular way to achieve this.

[[edit](http://en.wikipedia.org/w/index.php?title=Numerical_analysis&action=edit&section=13)] Solving equations and systems of equations

Another fundamental problem is computing the solution of some given equation. Two cases are commonly distinguished, depending on whether the equation is linear or not. For instance, the equation 2x + 5 = 3 is linear while 2x2 + 5 = 3 is not.

Much effort has been put in the development of methods for solving [systems of linear equations](http://en.wikipedia.org/wiki/Systems_of_linear_equations). Standard direct methods, i.e., methods that use some [matrix decomposition](http://en.wikipedia.org/wiki/Matrix_decomposition) are [Gaussian elimination](http://en.wikipedia.org/wiki/Gaussian_elimination), [LU decomposition](http://en.wikipedia.org/wiki/LU_decomposition), [Cholesky decomposition](http://en.wikipedia.org/wiki/Cholesky_decomposition) for [symmetric](http://en.wikipedia.org/wiki/Symmetric_matrix) (or [hermitian](http://en.wikipedia.org/wiki/Hermitian_matrix)) and [positive-definite matrix](http://en.wikipedia.org/wiki/Positive-definite_matrix), and [QR decomposition](http://en.wikipedia.org/wiki/QR_decomposition) for non-square matrices. [Iterative methods](http://en.wikipedia.org/wiki/Iterative_method) such as the [Jacobi method](http://en.wikipedia.org/wiki/Jacobi_method), [Gauss–Seidel method](http://en.wikipedia.org/wiki/Gauss%E2%80%93Seidel_method), [successive over-relaxation](http://en.wikipedia.org/wiki/Successive_over-relaxation) and [conjugate gradient method](http://en.wikipedia.org/wiki/Conjugate_gradient_method) are usually preferred for large systems.

[Root-finding algorithms](http://en.wikipedia.org/wiki/Root-finding_algorithm) are used to solve nonlinear equations (they are so named since a root of a function is an argument for which the function yields zero). If the function is [differentiable](http://en.wikipedia.org/wiki/Derivative) and the derivative is known, then [Newton's method](http://en.wikipedia.org/wiki/Newton%27s_method) is a popular choice. [Linearization](http://en.wikipedia.org/wiki/Linearization) is another technique for solving nonlinear equations.

[[edit](http://en.wikipedia.org/w/index.php?title=Numerical_analysis&action=edit&section=14)] **Solving eigenvalue or singular value problems**

Several important problems can be phrased in terms of [eigenvalue decompositions](http://en.wikipedia.org/wiki/Eigenvalue_decomposition) or [singular value decompositions](http://en.wikipedia.org/wiki/Singular_value_decomposition). For instance, the [spectral image compression](http://en.wikipedia.org/wiki/Image_compression) algorithm[[4]](http://en.wikipedia.org/wiki/Numerical_analysis#cite_note-3) is based on the singular value decomposition. The corresponding tool in statistics is called [principal component analysis](http://en.wikipedia.org/wiki/Principal_component_analysis).

[[edit](http://en.wikipedia.org/w/index.php?title=Numerical_analysis&action=edit&section=15)] **Optimization**

Main article: [Optimization (mathematics)](http://en.wikipedia.org/wiki/Optimization_(mathematics))

Optimization problems ask for the point at which a given function is maximized (or minimized). Often, the point also has to satisfy some [constraints](http://en.wikipedia.org/wiki/Constraint_(mathematics)).

The field of optimization is further split in several subfields, depending on the form of the objective function and the constraint. For instance, [linear programming](http://en.wikipedia.org/wiki/Linear_programming) deals with the case that both the objective function and the constraints are linear. A famous method in linear programming is the [simplex method](http://en.wikipedia.org/wiki/Simplex_method).

The method of [Lagrange multipliers](http://en.wikipedia.org/wiki/Lagrange_multipliers) can be used to reduce optimization problems with constraints to unconstrained optimization problems.

[[edit](http://en.wikipedia.org/w/index.php?title=Numerical_analysis&action=edit&section=16)] **Evaluating integrals**

Main article: [Numerical integration](http://en.wikipedia.org/wiki/Numerical_integration)

Numerical integration, in some instances also known as numerical [quadrature](http://en.wikipedia.org/wiki/Quadrature_(mathematics)), asks for the value of a definite [integral](http://en.wikipedia.org/wiki/Integral). Popular methods use one of the [Newton–Cotes formulas](http://en.wikipedia.org/wiki/Newton%E2%80%93Cotes_formulas) (like the midpoint rule or [Simpson's rule](http://en.wikipedia.org/wiki/Simpson%27s_rule)) or [Gaussian quadrature](http://en.wikipedia.org/wiki/Gaussian_quadrature). These methods rely on a "divide and conquer" strategy, whereby an integral on a relatively large set is broken down into integrals on smaller sets. In higher dimensions, where these methods become prohibitively expensive in terms of computational effort, one may use [Monte Carlo](http://en.wikipedia.org/wiki/Monte_Carlo_method) or [quasi-Monte Carlo methods](http://en.wikipedia.org/wiki/Quasi-Monte_Carlo_method) (see [Monte Carlo integration](http://en.wikipedia.org/wiki/Monte_Carlo_integration)), or, in modestly large dimensions, the method of [sparse grids](http://en.wikipedia.org/wiki/Sparse_grid).

[[edit](http://en.wikipedia.org/w/index.php?title=Numerical_analysis&action=edit&section=17)] **Differential equations**

Main articles: [Numerical ordinary differential equations](http://en.wikipedia.org/wiki/Numerical_ordinary_differential_equations) and [Numerical partial differential equations](http://en.wikipedia.org/wiki/Numerical_partial_differential_equations)

Numerical analysis is also concerned with computing (in an approximate way) the solution of [differential equations](http://en.wikipedia.org/wiki/Differential_equation), both ordinary differential equations and [partial differential equations](http://en.wikipedia.org/wiki/Partial_differential_equation).

Partial differential equations are solved by first discretizing the equation, bringing it into a finite-dimensional subspace. This can be done by a [finite element method](http://en.wikipedia.org/wiki/Finite_element_method), a [finite difference](http://en.wikipedia.org/wiki/Finite_difference) method, or (particularly in engineering) a [finite volume method](http://en.wikipedia.org/wiki/Finite_volume_method). The theoretical justification of these methods often involves theorems from [functional analysis](http://en.wikipedia.org/wiki/Functional_analysis). This reduces the problem to the solution of an algebraic equation.

[[edit](http://en.wikipedia.org/w/index.php?title=Numerical_analysis&action=edit&section=18)] **Software**

Main articles: [List of numerical analysis software](http://en.wikipedia.org/wiki/List_of_numerical_analysis_software) and [Comparison of numerical analysis software](http://en.wikipedia.org/wiki/Comparison_of_numerical_analysis_software)

Since the late twentieth century, most algorithms are implemented in a variety of programming languages. The [Netlib](http://en.wikipedia.org/wiki/Netlib) repository contains various collections of software routines for numerical problems, mostly in [Fortran](http://en.wikipedia.org/wiki/Fortran) and [C](http://en.wikipedia.org/wiki/C_(programming_language)). Commercial products implementing many different numerical algorithms include the [IMSL](http://en.wikipedia.org/wiki/IMSL_Numerical_Libraries) and [NAG](http://en.wikipedia.org/wiki/Numerical_Algorithms_Group) libraries; a free alternative is the [GNU Scientific Library](http://en.wikipedia.org/wiki/GNU_Scientific_Library).

There are several popular numerical computing applications such as [MATLAB](http://en.wikipedia.org/wiki/MATLAB), [S-PLUS](http://en.wikipedia.org/wiki/S-PLUS), [LabVIEW](http://en.wikipedia.org/wiki/LabVIEW), and [IDL](http://en.wikipedia.org/wiki/IDL_(programming_language)) as well as free and open source alternatives such as [FreeMat](http://en.wikipedia.org/wiki/FreeMat), [Scilab](http://en.wikipedia.org/wiki/Scilab), [GNU Octave](http://en.wikipedia.org/wiki/GNU_Octave) (similar to Matlab), [IT++](http://en.wikipedia.org/wiki/IT%2B%2B) (a C++ library), [R](http://en.wikipedia.org/wiki/R_(programming_language)) (similar to S-PLUS) and certain variants of [Python](http://en.wikipedia.org/wiki/Python_(programming_language)). Performance varies widely: while vector and matrix operations are usually fast, scalar loops may vary in speed by more than an order of agnitude.[[5]](http://en.wikipedia.org/wiki/Numerical_analysis#cite_note-4)[[6]](http://en.wikipedia.org/wiki/Numerical_analysis#cite_note-5)

Many [computer algebra systems](http://en.wikipedia.org/wiki/Computer_algebra_system) such as [Mathematica](http://en.wikipedia.org/wiki/Mathematica) also benefit from the availability of [arbitrary precision arithmetic](http://en.wikipedia.org/wiki/Arbitrary_precision_arithmetic) which can provide more accurate results.

Also, any [spreadsheet](http://en.wikipedia.org/wiki/Spreadsheet) software can be used to solve simple problems relating to numerical analysis.

[[edit](http://en.wikipedia.org/w/index.php?title=Numerical_analysis&action=edit&section=19)] See also

[Scientific computing](http://en.wikipedia.org/wiki/Scientific_computing), [List of numerical analysis topics](http://en.wikipedia.org/wiki/List_of_numerical_analysis_topics), [Gram-Schmidt process](http://en.wikipedia.org/wiki/Gram-Schmidt_process)

[Numerical differentiation](http://en.wikipedia.org/wiki/Numerical_differentiation), [Symbolic-numeric computation](http://en.wikipedia.org/wiki/Symbolic-numeric_computation), [Analysis of algorithms](http://en.wikipedia.org/wiki/Analysis_of_algorithms)

[Numerical Recipes](http://en.wikipedia.org/wiki/Numerical_Recipes)

|  |  |
| --- | --- |
|  | Wikibooks has more on the topic of [Numerical analysis](http://en.wikibooks.org/wiki/Special:Search/Numerical_analysis) |

Journals

[Numerische Mathematik](http://www-gdz.sub.uni-goettingen.de/cgi-bin/digbib.cgi?PPN362160546), volumes 1-66, Springer, 1959-1994 (searchable; pages are images). (English) (German)

[Numerische Mathematik at SpringerLink](http://www.springerlink.com/content/0029-599X), volumes 1-112, Springer, 1959–2009

[SIAM Journal on Numerical Analysis](http://siamdl.aip.org/dbt/dbt.jsp?KEY=SJNAAM), volumes 1-47, SIAM, 1964–2009

Software and Code

[Lists of free software for scientific computing and numerical analysis](http://norma.mas.ecp.fr/wikimas/ScientificComputingSoftware) (English) (French)

[Numerical methods for Fortran programmers](http://people.sc.fsu.edu/~tomek/Fortran/num_meth.html)

[Java Number Cruncher](http://www.apropos-logic.com/nc/) features free, downloadable code samples that graphically illustrate common numerical algorithms

[Excel Implementations](http://www.ifh.uni-karlsruhe.de/people/fenton/Lectures.html)

[Several Numerical Mathematical Utilities (in Javascript)](http://www.akiti.ca/Mathfxns.html)

Online Texts

[Numerical Recipes](http://www.nr.com/oldverswitcher.html), William H. Press (free, downloadable previous editions)

[First Steps in Numerical Analysis](http://kr.cs.ait.ac.th/~radok/math/mat7/stepsa.htm#Numerical%20Analysis), R.J.Hosking, S.Joe, D.C.Joyce, and J.C.Turner

[Numerical Analysis for Engineering](http://ece.uwaterloo.ca/~dwharder/NumericalAnalysis/), D. W. Harder

[CSEP (Computational Science Education Project)](http://www.phy.ornl.gov/csep/CSEP/TEXTOC.html), [U.S. Department of Energy](http://en.wikipedia.org/wiki/U.S._Department_of_Energy)

Online Course Material

[Numerical Methods](http://www.damtp.cam.ac.uk/user/fdl/people/sd103/lectures/nummeth98/index.htm#L_1_Title_Page), Stuart Dalziel [University of Cambridge](http://en.wikipedia.org/wiki/University_of_Cambridge)

[Lectures on Numerical Analysis](http://www.math.upenn.edu/~wilf/DeturckWilf.pdf), Dennis Deturck and Herbert S. Wilf [University of Pennsylvania](http://en.wikipedia.org/wiki/University_of_Pennsylvania)

[Numerical methods](http://www.ifh.uni-karlsruhe.de/people/fenton/LectureNotes/Numerical-Methods.pdf), John D. Fenton [University of Karlsruhe](http://en.wikipedia.org/wiki/University_of_Karlsruhe)

[Numerical Methods for Science, Technology, Engineering and Mathematics](http://numericalmethods.eng.usf.edu/), Autar Kaw [University of South Florida](http://en.wikipedia.org/wiki/University_of_South_Florida)

[Numerical Analysis Project](http://math.fullerton.edu/mathews/numerical.html), John H. Mathews [California State University, Fullerton](http://en.wikipedia.org/wiki/California_State_University,_Fullerton)

[Numerical Methods - Online Course](http://www.math.jct.ac.il/~naiman/nm/), Aaron Naiman [Jerusalem College of Technology](http://en.wikipedia.org/wiki/Jerusalem_College_of_Technology)

[Numerical Methods for Physicists](http://www-teaching.physics.ox.ac.uk/computing/NumericalMethods/NMfP.pdf), Anthony O’Hare [Oxford University](http://en.wikipedia.org/wiki/Oxford_University)

[Lectures in Numerical Analysis](http://kr.cs.ait.ac.th/~radok/math/mat7/stepsa.htm#Numerical%20Analysis), R. Radok [Mahidol University](http://en.wikipedia.org/wiki/Mahidol_University)

[Introduction to Numerical Analysis for Engineering](http://ocw.mit.edu/OcwWeb/Mechanical-Engineering/2-993JSpring-2005/CourseHome/), Henrik Schmidt [Massachusetts Institute of Technology](http://en.wikipedia.org/wiki/Massachusetts_Institute_of_Technology)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | [[hide](http://en.wikipedia.org/wiki/Numerical_analysis)][v](http://en.wikipedia.org/wiki/Template:Mathematics-footer) · [d](http://en.wikipedia.org/wiki/Template_talk:Mathematics-footer) · [e](http://en.wikipedia.org/w/index.php?title=Template:Mathematics-footer&action=edit)[Areas](http://en.wikipedia.org/wiki/Areas_of_mathematics) of [mathematics](http://en.wikipedia.org/wiki/Mathematics) | | |  |  | | [Arithmetic](http://en.wikipedia.org/wiki/Arithmetic) · [Logic](http://en.wikipedia.org/wiki/Mathematical_logic) · [Set theory](http://en.wikipedia.org/wiki/Set_theory) · [Combinatorics](http://en.wikipedia.org/wiki/Combinatorics) · [Algebra](http://en.wikipedia.org/wiki/Algebra) ([elementary](http://en.wikipedia.org/wiki/Elementary_algebra) – [linear](http://en.wikipedia.org/wiki/Linear_algebra) – [abstract](http://en.wikipedia.org/wiki/Abstract_algebra)) · [Number theory](http://en.wikipedia.org/wiki/Number_theory) · [Analysis](http://en.wikipedia.org/wiki/Mathematical_analysis)/[Calculus](http://en.wikipedia.org/wiki/Calculus) · [Geometry](http://en.wikipedia.org/wiki/Geometry) · [Topology](http://en.wikipedia.org/wiki/Topology) · [Statistics](http://en.wikipedia.org/wiki/Mathematical_statistics) · [Differential equations](http://en.wikipedia.org/wiki/Differential_equation)/[Dynamical systems](http://en.wikipedia.org/wiki/Dynamical_systems_theory) · [Mathematical physics](http://en.wikipedia.org/wiki/Mathematical_physics) · Numerical analysis · [Computation](http://en.wikipedia.org/wiki/Theory_of_computation) · [Information theory](http://en.wikipedia.org/wiki/Information_theory) · [Probability](http://en.wikipedia.org/wiki/Probability_theory) · [Optimization](http://en.wikipedia.org/wiki/Optimization_(mathematics)) · [Game theory](http://en.wikipedia.org/wiki/Game_theory) | | |  |  | | [Category](http://en.wikipedia.org/wiki/Category:Mathematics)  · [Mathematics portal](http://en.wikipedia.org/wiki/Portal:Mathematics)  · [Outline](http://en.wikipedia.org/wiki/Topic_outline_of_mathematics) · [Lists](http://en.wikipedia.org/wiki/Lists_of_mathematics_topics) | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | [[show](http://en.wikipedia.org/wiki/Numerical_analysis)][v](http://en.wikipedia.org/wiki/Template:Physics-footer) · [d](http://en.wikipedia.org/wiki/Template_talk:Physics-footer) · [e](http://en.wikipedia.org/w/index.php?title=Template:Physics-footer&action=edit)General subfields within [physics](http://en.wikipedia.org/wiki/Physics) | | |  |  | | [Acoustics](http://en.wikipedia.org/wiki/Acoustics) · [Agrophysics](http://en.wikipedia.org/wiki/Agrophysics) ([Soil physics](http://en.wikipedia.org/wiki/Soil_physics)) · [Astrophysics](http://en.wikipedia.org/wiki/Astrophysics) · [Atmospheric physics](http://en.wikipedia.org/wiki/Atmospheric_physics) · [Atomic, molecular, and optical physics](http://en.wikipedia.org/wiki/Atomic,_molecular,_and_optical_physics) · [Biophysics](http://en.wikipedia.org/wiki/Biophysics) ([Medical physics](http://en.wikipedia.org/wiki/Medical_physics) • [Neurophysics](http://en.wikipedia.org/wiki/Neurophysics)) · [Chemical physics](http://en.wikipedia.org/wiki/Chemical_physics) ·[Condensedmatter physics](http://en.wikipedia.org/wiki/Condensed_matter_physics) · [Econophysics](http://en.wikipedia.org/wiki/Econophysics) · [Electromagnetism](http://en.wikipedia.org/wiki/Electromagnetism) · [Geophysics](http://en.wikipedia.org/wiki/Geophysics)  · [Mechanics](http://en.wikipedia.org/wiki/Mechanics) ([Classical mechanics](http://en.wikipedia.org/wiki/Classical_mechanics)  • [Continuum mechanics](http://en.wikipedia.org/wiki/Continuum_mechanics) • [Quantum mechanics](http://en.wikipedia.org/wiki/Quantum_mechanics) • [Fluid mechanics](http://en.wikipedia.org/wiki/Fluid_mechanics) • [Optomechanics](http://en.wikipedia.org/wiki/Optomechanics) • [Thermodynamics](http://en.wikipedia.org/wiki/Thermodynamics)) · [Nuclear physics](http://en.wikipedia.org/wiki/Nuclear_physics) · [Optics](http://en.wikipedia.org/wiki/Optics) · [Particle physics](http://en.wikipedia.org/wiki/Particle_physics) · [Psychophysics](http://en.wikipedia.org/wiki/Psychophysics) · [Quantum field theory](http://en.wikipedia.org/wiki/Quantum_field_theory) · [Relativity](http://en.wikipedia.org/wiki/Theory_of_relativity) ([Special relativity](http://en.wikipedia.org/wiki/Special_relativity) • [General relativity](http://en.wikipedia.org/wiki/General_relativity)) | | |

Retrieved from "<http://en.wikipedia.org/wiki/Numerical_analysis>"

[Categories](http://en.wikipedia.org/wiki/Special:Categories): [Numerical analysis](http://en.wikipedia.org/wiki/Category:Numerical_analysis) | [Mathematical analysis](http://en.wikipedia.org/wiki/Category:Mathematical_analysis) | [Applied mathematics](http://en.wikipedia.org/wiki/Category:Applied_mathematics) | [Physics](http://en.wikipedia.org/wiki/Category:Physics) | [Computational science](http://en.wikipedia.org/wiki/Category:Computational_science)

**Part III**

**Часто встречающиеся англо – русско – казахские грамматические термины**

|  |  |  |
| --- | --- | --- |
| attribute | определение | анықтауыш |
| article | артикль | артикль |
| affirmative sentence | утвердительное предложение | болымды сөйлем |
| abstract nouns | абстрактное существительные | дерексіз зат есім |
| animate nouns | одушевленные существительные | жанды зат есім |
| adverb | наречие | үстеу |
| adverbs of frequency | наречия частоты | жиілік үстеу |
| adverbs of degree | наречие меры | мөлшер үстеу |
| adverbs of time | наречия времени | мезгіл үстеу |
| adverbs of place | наречия место | мекен үстеу |
| adverbs of manner | наречие действия | сын бейне үстеу |
| adverbial modifier | обстоятельство | пысықтауыш |
| abbreviate | сокращать | қысқарту |
| auxiliary verb | вспомагательный глагол | көмекші етістік |
| active voice | действительный залог | негізгі етіс |
| adjective | прилагательное | сын есім |
| alternative question | альтернативный вопрос | балама сұрақ |
| bracket | скобка | жақша |
| capital letter | заглавная буква | бас әріп |
| consonant | согласный | дауыссыз |
| cardinal numeral | количественное числительное | есептік сан есім |
| conjunction | союз | шылау |
| common nouns | нарицательное существительные | жалпы зат есім |
| common case | общий падеж | жалпы септік |
| colon | двоеточие | қос нүкте |
| collective nouns | собирательные существительные | жинақтау зат есім |
| compound predicate | составное сказуемое | күрделі баяндауыш |
| countable nouns | исчисляемые существительные | саналатын зат есім |
| comparative degree | сравнительная степень | салыстырмалы шырай |
| complex object | сложное дополнение | күрделі толықтауыш |
| compound | соединение, состав | күрделі |
| composite adverb | составное наречие | құрама үстеу |
| composite sentence | составное предложение | құрмалас сөйлем |
| construction | конструкция | құрылым |
| complex sentence | сложно-подчиненное предложение | сабақтас құрмалас сөйлем |
| compound sentence | сложно-сочиненное предложение | салалас құрмалас сөйлем |
|  |  |  |
| сontinuous tense | продолженное время | шақтың созылыңқы түрі |
| comma | запятая | үтір |
| common nouns | нарицательные существительные | жалпы есім |
| definite article | определенный артикль | белгілілік артиклі |
| dash | тире | сызықша |
| degrees of comparison  of adjectives | степени сравнения прилагательных | сын есім шырайлары |
| demonstrative pronouns | указательные местоимения | сілтеу есімдік |
| derivative adverb | производное наречие | туынды үстеу |
| degrees of adverbs | степени сравнения наречий | үстеудің шырай түрлері |
| declarative sentence | повествовательное предложение | хабарлы сөйлем |
| derivative | производный | туынды |
| direct speech | прямая речь | төл сөз |
| direct object | прямое дополнение | тура толықтауыш |
| indirect speech | косвенная речь | төлеу сөз |
| ending | окончание | жалғау |
| extended sentence | распространенное предложение | жайылма сойлем |
| emphatic pronouns | выразительные местоимения | күшейтпелі есімдік |
| exclamatory sentence | восклицательное предложение | лепті сөйлем |
| Future Perfect Tense | будущее совершенное время | аяқталған келер шақ |
| Future Perfect Continuous Tense | будущее совершенное продолженное время | келер шақтың аяқталған созылыңқы түрі |
| fractional numerals | дробные числительные | бөлшектік есім |
| Future Indefinite  (Simple) Tense | будущее неопределенное время | жалпы келер шақ |
| full stop | точка | нүкте |
| Future Continuous (Progressive) Tense | будущее продолженное время | созылыңқы келер шақ |
| gerund | герундий | герундий |
| general question | общий вопрос | жалпы сұрақ |
| hyphen | дефис | дефис |
| inanimate nouns | неодушевленные существительные | жансыз зат есім |
| indicative mood | изъявительное наклонение | ашық рай |
| indefinite article | неопределенный артикль | белгісіздік артиклі |
| indefinite рronouns | неопределенные местоимения | белгісіздік есімдік |
| Indefinite (Simple) Tenses | неопределенные времена | шақтың жалпы түрі |
| imperative мood | повелительное наклонение | бұйрық рай |
| imperative sentence | повелительное предложение | бұйрықты сөйлем |
| rule | правило | ереже |
| indirect object | косвенное дополнение | жанама толықтауыш |
| interrogative pronouns | вопросительные местоимения | сұрау есімдік |
| interrogative sentence | вопросительное предложение | сұраулы сөйлем |
| infinitive | инфинитив | тұйық етістік |
| letter | буква | әріп |
| link verb | глагол-связка | байланыстырушы етістік |
| modal verb | модальный глагол | модаль етістік |
| mood | наклонение | рай |
| negative sentence | отрицательное предложение | болымсыз сөйлем |
| negative pronouns | отрицательные местоимения | болымсыздық есімдік |
| noun | существительное | зат есім |
| nominative case | именительный падеж | атау септік |
| numeral | числительное | сан есім |
| object | ополнение | толықтауыш |
| ordinal numerals | порядковые числительные | реттік сан есім |
| Perfect Tenses | совершённые времена | шақтың аяқталған түрі |
| Present Perfect Tense | настоящее совершенное время | аяқталған осы шақ |
| Past Perfect Tense | прошедшее совершенное время | аяқталған өткен шақ |
| Perfect Continuous Tense | совершенное продолженное время | шақтың аяқталған созылыңқы түрі |
| Present Perfect Continuous Tense | настоящее совершенное продолженное время | осы шақтың аяқталған созылыңқы түрі |
| Past Perfect Continuous Tense | прошедшее совершенное продолженное время | өткен шақтың аяқталған созылыңқы түрі |
| predicate | сказуемое | баяндауыш |
| preposition | предлог | демеулік |
| pronoun | местоимение | есімдік |
| participle | причастие | есімше |
| Present Indefinite (Simple)Tense | настоящее неопределенное время | жалпы осы шақ |
| Past Indefinite (Simple) Tense | прошедшее неопределенное время | жалпы өткен шақ |
| positive degree | положительная степень | жай шырай |
| personal sentence | личное предложение | жақты сөйлем |
| proper nouns | собственное существительные | жалқы есім |
| personal pronouns | личные местоимения | жіктеу есімдік |
| polysyllable | многосложное слово | көп буынды сөз |
| plural form | множественное число | көпше түрі |
| Present Continuous (Progressive) Tense | настоящее продолженное время | созылыңқы осы шақ |
| Past Continuous (Progressive) Tense | прошедшее продолженное время | созылыңқы өткен шақ |
| principal clause | главное предложение | негізгі сөйлем |
| parts of speech | части речи | сөз таптары |
| principal parts of the sentence | главные члены предложения | сөйлемнің тұрлаулы мүшелері |
| possessive pronouns | притяжательные местоимения | тәуелдік есімдік |
| possessive case | притяжательный падеж | тәуелдік септік |
| punctuation mark | знак препинания | тыныс белгісі |
| passive voice | страдательный залог | ырықсыз етіс |
| pronunciation | произношение | оқылуы |
| regular verb | правильный глагол | дұрыс етістік |
| irregular verb | неправильный глагол | бұрыс етістік |
| relative pronouns | относительные местоимения | қатыстық есімдік |
| reflexive pronouns | возвратные местоимения | өздік есімдік |
| reciprocal pronouns | взаимные местоимения | ортақ есімдік |
| special question | специальный вопрос | арнайы сұрақ |
| subordinate clause | подчиненное предложение | бағыныңқы сөйлем |
| subjunctive mood | сослагательное наклонение | шартты рай |
| syllable | слог | буын |
| subject | подлежащее | бастауыш |
| supposition | предположение | болжау |
| simple attribute | простое определение | дара анықтауыш |
| superlative degree | превосходная степень | күшейтпелі шырай |
| syntax | синтаксис | синтаксис |
| suffix | суффикс | жұрнақ |
| sentence | предложение | сөйлем |
| singular form | единственное число | жекеше түрі |
| translation | перевод | аудармасы |
| types of sentences | виды предложений | сөйлем түрлері |
| voiceless sound | глухой звук | қатаң дыбыс |
| uncountable nouns | неисчисляемые существительные | саналмайтын зат есім |
| verb | глагол | етістік |

**Grammar explanations**

Существительное (The Noun)

Существительные в английском языке, как и в русском языке, обозначают предметы и вещества (a table — стол; a house — дом; water — вода), живые су­щества (a **girl** — девочка; a cat — кошка), явления и отвлеченные понятия (rain — дождь; peace — мир; beauty — красота; work — работа; darkness — темнота).

Существительные в английском языке имеют ряд свойств, которые отличают их от существительных в русском языке. Основные из них следующие:

1. Перед существительными обычно ставятся особые служебные слова — артикли a, an, the:

I am a student. Я студент.

Open the door, please. Откройте дверь, пожалуйста.

2. Большинство существительных не изменяются по падежам, и только одушевленные существительные имеют два падежа: общий (a boy, boys) и притяжательный (boy's, boys'):

What is the boy's name? Как имя мальчика?

Where are the girls' hats? Где шляпы девочек?

3. Существительные могут употребляться в качестве беспредложного определения к другим существительным. В этом случае они часто соответствуют по значению прилагательным в русском языке:

You must wear your school uniform. Вы должны носить школьную

форму.  
 Which do you like best: winter or sum- Какие виды спорта вы любите

mer sports? больше: зимние или летние?

Множественное число имен существительных (Plural of Nouns)

Множественное число имен существительных в английском языке образует­ся путем прибавления окончания –s, например:

а реn [ə 'реn] ручка pens [penz] ручки a desk [ə 'desk] парта desks [desks] парты a boy [ə 'boI] мальчик boys [boIz] мальчики

Существительные, оканчивающиеся на -s, -ch, -х, -sh, -ss, -о, во множествен­ном числе принимают окончание -es, например:

а bush [ə 'bu∫l куст bushes ['bu∫Iz] кусты a glass [ə 'glα**:**s] стакан glasses ['glα**:**sIz] стаканы a dress [ə 'dres] платье dresses ['dresIz] платья a box [ə 'boks] коробка boxes ['boksIz] коробки a match [ə 'mæt∫] матч matches ['mæt∫Iz] матчи a hero [ə 'hIərou] герой heroes ['hIərouz] герои

Существительные, оканчивающиеся на -f, -fæe, обычно меняют во множествен­ном числе -f, -fe на -v и принимают окончание -es, например:

a knife [ə 'naIf] нож knives [naIvz] ножи a wolf [ə 'wulf] волк wolves [wulvz] волки

Существительные, оканчивающиеся на **-у** с предшествующей согласной, ме­няют во множественном числе букву у на i и принимают окончание -es, например:æ ????

a factory [ə'fæktər I] завод, фабрика factories ['fæktər Iz] заводы, фабрики

Некоторые существительные образуют множественное число путем изменения корневой гласной буквы, не принимая никаких окончаний, например:

a man [ə'mæn] мужчина men [men] мужчины

a woman [ə'wumən] женщина women ['wI mI n] женщины

a tooth [ə'tu: θ] зуб teeth [ti: θ] зубы

a foot [ə'fut] ступня, нога feet [fi:t] ступни, ноги

Некоторые существительные в английском языке употребляются только в  
единственном числе, например:

advice [əd'va Is] совет, советы permission [pə'mI∫ən] разрешение  
 information [I nfə 'meI∫ən] сведение, progress ['prougres] успех, успехи  
 сведения

iron ['aIən] железо water ['wotə] вода

money ['m∧nI] деньги weather ['weðə] погода

news [nju:z] новость, новости work [wə: k] работа

Падежи имен существительных

Существительные в английском языке имеют только два падежа: общий падеж (the Common Case) и притяжательный падеж (the Possessive Case).

Притяжательный падеж выражает принадлежность и таким образом соответ­ствует в русском языке родительному падежу. В форме притяжательного падежа употребляются в основном существительные, обозначающие одушевленные пред­меты.

Притяжательный падеж существительных в единственном числе образуется прибавлением апострофа и буквы s (-'s) к форме существительного в общем падеже. Окончание -'s произносится [s] после глухих согласных, [z] после звонких соглас­ных и гласных и [Iz] после шипящих и свистящих, т. е. так же, как и окончание -s при образовании множественного числа существительных.

Единственное число

Общий падеж Притяжательный падеж

aunt [∧nt] тётя aunt's house ['∧nts 'haus] дом тети dog [dog ] собака dog's name ['dogz 'neI m] кличка собаки father ['f α: ðə] отец father's book ['fα:ðəz 'buk] книга отца James [dʒeI mz] Джеймс James’s friend ['dʒeI mzIz frend] друг Джеймсa George [dʒo:dʒ] Джорж George's son ['dʒo:dʒIz s∧n] сын Джорд­жа

Притяжательный падеж существительных во множественном числе, оканчи­вающихся на -(e)s, образуется путем добавления только апострофа -'.

Множественное число

Общий падеж Притяжательный падеж

horses ['ho:sIz] лошади horses' races **[**'ho:sIz 'reIsIz] скачки лошадей

dogs [dogz] собаки dogs' masters [dogz' 'mα:stəz] хозяева собак

Существительные, не имеющие окончания -(e)s во множественном числе, об­разуют притяжательный падеж множественного числа так же, как существитель­ные в единственном числе — путем прибавления окончания -s', например: chil­dren's literature ['t∫Ildrənz 'lItərIt∫ə] детская литература.

Артикль (The Article)

В английском языке существует два артикля: неопределенный а(аn) и опре­деленный the.

Неопределенный артикль а(аn) произошел от числительного one — «один> и поэтому употребляется только с исчисляемыми существительными в единствен­ном числе.

Неопределенный артикль имеет две формы: 1) а, стоящую перед существи­тельными, начинающимися с согласного звука или имеющими впереди себя оп­ределение, начинающееся с согласного звука, например: a book [ə'buk] книга, a big apple [ə big 'æpl] большое яблоко; 2) an, стоящую перед существительными, начинающимися с гласного звука или имеющими впереди себя определение, на­чинающееся с гласного звука, например: an egg [ən 'еg] яйцо, an old man [ən 'ould 'mæn] старик.

Неопределенный артикль на русский язык не переводится и передает лишь указание на принадлежность предмета к какому-либо классу однородных предме­тов. Например: a book значит какая-то, одна из, любая, какая-нибудь книга. Если мы говорим Give me a book ['gIv mi: ə 'buk], то мы хотим подчеркнуть, что нам нужна книга, а не тетрадь, ручка, карандаш и т. д.

Определенный артикль происходит от указательного местоимения that [ðæt] «тот» и имеет одну форму the, которая произносится [ðə] перед существительными, начинающимися с согласного звука, например: the room [ðə 'rum] (эта) комната, the big apple [ðə 'bIg 'æр1] (это) большое яблоко, и [ði:] перед существительными, на­чинающимися с гласного звука, или перед определением, начинающимся с глас­ного звука, например: the evening [ði:'i:vnIŋ] (этот) вечер, the old house [ði:'ould 'haus] (этот) старый дом.

Определенный артикль может употребляться как с исчисляемыми, так и с неисчисляемыми существительными в единственном и множественном числе.

Определенный артикль употребляется, когда нужно выделить какой-либо предмет, лицо или явление из класса ему подобных, например: Giye me the book ['gIv mi: ðə 'buk]. Дайте мне эту книгу (ту, которая лежит на столе).

Употребление неопределенного артикля

Неопределенный артикль употребляется:

1) с существительным — именным членом составного сказуемого:

Mike is a student. Миша студент.

2) с существительным-подлежащим после оборота there is (there was, there will be):

There's a good library in our school. В нашей школе есть хорошая библио­тека.

3) с дополнением после глагола have:

I have a sister. У меня есть сестра.

Употребление определенного артикля с исчисляемыми существительными

Определенный артикль употребляется, когда собеседникам (или пишущему и читателю) известно из окружающей обстановки (ситуации), предшествующего опыта или словесного окружения (контекста), о каком или каких именно предме­тах (лицах, явлениях) идет речь.

1. Определенный артикль обычно употребляется, когда перед существительным стоит:

а) порядковое числительное:

Gagarin was the first man to fly into Гагарин был первым человеком, ко-  
 space. торый полетел в космос.

б) прилагательное в превосходной степени:

This is the shortest way to the town. Это кратчайший путь к городу.

Jack is the best runner in our school. Джек — лучший бегун в нашей школе.

в) одно из прилагательных — following, last, next, same:

Copy the following sentences. Спишите следующие предложения.

You've made the same mistake. Вы сделали ту же самую ошибку.

The next stop is ours. Следующая остановка наша.

Примечание: Существительные с прилагательными next и last употребля­ются без артикля в сочетаниях next door (в соседнем доме), next week (на будущей неделе), next month, next summer, next year, next Sunday (в будущее воскресенье), next time (в следующий раз), last week (на прошлой неделе), last month, last year, last Saturday и т. п.:

Tom lives next door to me. Том живет рядом со мной.

I'll be on duty next week. Я буду дежурить на следующей неделе.

I was there last night. Я был там вчера.

2. Определенный артикль употребляется, когда за существительным следует  
определение:

Неге is the book *I bought yesterday*. Вот книга, *которую я вчера купил*.

Can you show me the way *to the sta*- Вы можете мне показать дорогу *на*

*tion*? *станцию*?

The trees *planted* by us have taken root. Деревья, *посаженные* нами, прижились.

This is the house *where I was born*. Вот дом, *где я родился*.

3. Определенный артикль употребляется в тех случаях, когда из предшествующего  
контекста ясно, о каком предмете (лице, явлении) или каких предметах (лицах, явлениях)  
идет речь:

We've received a letter from an English schoolgirl. **The girl** writes in **the letter** that she wants to correspond with Russian schoolchildren.

Petrov mis-spelt a word in his sentence. Who has noticed **the mistake**?

В данных примерах из первых предложений ясно, о каких именно предме­тах (лицах, явлениях) идет речь (the letter, the girl, the mistake).

4. Определенный артикль употребляется с существительными, обозначающими предметы, единственные в своем роде: **the sun, the moon,** the earth, the ground, the **world**, **the sky, the** air и некоторые другие:

Our scientists were the first to pho- Наши ученые были первыми, кто сфотогра-

tograph the far side of the Moon. фировал обратную сторону Лу­ны.

Our sportsmaen are among the best in Наши спортсмены одни из лучших в мире.

the world.

5. Определенный артикль может употребляться с исчисляемыми существи­тельными в единственном числе, не выделяя предмет из ряда ему подобных, а обозначая весь класс однородных предметов:

**The rose** is a beautiful flower. Роза — красивый цветок.

**The dog** is the friend of man. Собака — друг человека.

Отсутствие артиклей перед исчисляемыми существительными

Артикли не употребляются с исчисляемыми существительными в следующих случаях:

1. если существительное является обращением:

**Children,** bequiet! Дети, тише.

1. перед словами **Father, Mother** в высказываниях членов семьи:

Ann, hasn't **Father** come yet? Аня, папа еще не пришел?

3) перед названиями дней недели, месяцев и времен года:

**Spring** has comeat last! Весна наконец пришла.

**July** is a summer month. Июль — летний месяц.

We do not go to school on Sunday. Мы не ходим в школу в воскресенье.

4) перед существительными **breakfast, dinner, lunch, supper:**

I have **breakfast** at seven. Я завтракаю в семь часов.

Will you stay to **dinner**? Вы останетесь обедать?

**Употребление артиклей с неисчисляемыми существительными**

Неисчисляемые существительные (отвлеченные и вещественные) употребля­ются, как правило, без артиклей:

I like **music**. Я люблю музыку.

**Mathematics** and **physics** are my Математика и физика — мои

favou­rite subjects. любимые предметы.

Give me a glass of **water**, please Дайте мне стакан воды, пожалуйста..

We struggle for peace. Мы боремся за м ир.

Если отвлечённые ивещественные существительные имеют при себе ограничитель- ные определения или уточняются ситуацией, они употребляются с определённым артик- лем:

Pass me the bread, please. Передайте мне хлеб, пожалуйста.

Артикли с именами собственными

Имена собственные употребляются, как правило, без артиклей: Zaure, Almaty, Kazakhstan, Australia, Gogol Street:

Astana is the capital of Kazakhstan. Астана – столица Казахстана.

I live in Gogol Street. Я живу на улице Гоголя.

С определённым артиклем употребляются следующие имена собственные:

1. названия морей, океанов, рек, каналов, проливов, горных цепей, пустынь, групп островов: the Caspian Sea, the Pacific Ocean, the Ili, the Lena, the Beringov Channel, the Ili Alatau, the Karakum:

Almaty is situated in the vicinity of the Ili Alatau.

Алматы расположен в предгорьях Заилийского Ала-Тау.

1. Названия стран света: the North, the South, the West, the East:

They live somewhere in the North. Они живут где-то на севере.

1. Фамилии во мнoжественном числе для обозначения всей (всех членов) семьи:

Next year the Askarovs will go to Siberia. В следующем году Аскаровы поедут в Сибирь.

Прилагательное (The Adjective)

В английском языке, как и в русском, прилагательные обозначают признаки, качества или свойства предметов.

По значению прилагательные делятся на две группы: качественные и относи- тельные.

Качественные прилагательные обозначают признаки и качества предметов, следовательно, имеют степени сравнения.

Степени сравнения имён прилагательных и наречеий

Односложные и двусложные прилагательные образуют степени сравнения путём прибавления суффиксов:

-er в сравнительной степени, -est в превосходной степени.

|  |  |  |
| --- | --- | --- |
| Положительная степень | Сравнительная степень | Превосходная степень |
| short [∫o:t] корткий  big [ bIg] большой  long [ loŋ] длинный  easy ['i:z I] лёгкий | shorter ['∫o:tə] короче  bigger ['bIgə] больше  longer ['loŋgə] длиньше  easier ['i:z Iə] легче | shortest ['∫o:tIst] самый  короткий  biggest ['bIg Ist] самый  большой  longest ['loŋgIst] самый  длинный  easiest ['i:z I Ist] самый  лёгкий |

ə ε θ α ә ⋀ ' ŋ I g ∂

При образовании простых форм сравнительной и превосходной степеней не­обходимо соблюдать следующие правила:

1) удваивать в написании конечную согласную в сравнительной и превосходной степенях односложных прилагательных, оканчивающихся на одну согласную с предшествующим кратким гласным звуком:

hot — hotter — hottest

big — bigger — biggest

2) изменять букву у на i в сравнительной и превосходной степенях прилагательных, оканчивающихся на -у с предшествующей согласной:

lazy — lazier — laziest

happy — happier — happiest

3) опускать немое e в сравнительной и превосходной степенях:

large — larger — largest

simple — simpler — simplest

Некоторые прилагательные (наречия) образуют формы степеней сравнения от других корней:

good хороший better ['betə] лучше, best [best] самый лучший, наилучший, лучше всего

well хорошо

bad плохой

badly плохо worst [wə:st] самый худший, наихудший, хуже всего

many много, much много

little маленький, мало — less [les] меньше, least [li:st] наименьший, меньше всего

Прилагательные, состоящие более чем из двух слогов, образуют степени срав­нения путем постановки перед ними слов mоге [mо:] (в сравнительной степени) и most [moust] (в превос ходной степени).

|  |  |  |
| --- | --- | --- |
| Положительная степень | Сравнительная степень | Превосходная степень |
| interesting интересный  beautiful красивый | more interesting  более интересный  more beautiful  более красивый | most interesting  самый интересный  most beautiful  самый красивый |

Русскому обороту «такой же ... как» соответствует английский оборот as ... as, обороту «не такой ... как» — оборот not so ... as, оборотам «более ... чем», «ме­нее .. чем» — конструкция с than:

This flower is as beautiful as that one. Этот цветок такой же красивый, как и тот.

(Слово one заменяет слово «цветок».)

This story is not so interesting as that one. Этот рассказ не такой интересный, как тот

(рассказ).

This exercise is easier than that one. Это упражнение легче, чем то (упраж­нение).

Относительные прилагательные обозначают признаки и качества предметов через отношение к другим предметам, например: golden hair золотистые волосы, political economy политическая экономия.

В значении относительных прилагательных в английском языке выступают существительные-определители:

a school library школьная библиотека

a gold watch золотые часы

the autumn sky осеннее небо

a London street лондонская улица

**Числительные (Numerals)**

В английском языке, как и в русском, различаются количественные числи­тельные (Cardinal Numerals) и порядковые (Ordinal Numerals).

Количественные числительные обозначают число предметов и от­вечают на вопрос:

How many? (Сколько?) – I have two pens and three pencils.

Порядковые числительные обозначают порядок при счете и отвечают на вопрос:

Which? (Который по порядку?)

The school year begins on the first of September. July the twenty-second is my mother's birthday.

Образование количественных и порядковых числительных

1. Присоединяя суффикс -teen к соответствующим простым числительным, можно  
образовать числительные от 13 (thirteen) до 19 (nineteen):

four + teen = fourteen

six + teen = sixteen

seven + teen = seventeen

*Примечание*: Числительные thirteen и fifteen имеют изменения в основе (срав­ните: three, five).

2. Суффикс -ty, присоединяясь к простым числительным, образует десятки: от 20 (twenty) до 90 (ninety):

seven + ty = seventy

six + ty = sixty

nine + ty = ninety

Примечание: Числительные twenty, thirty и fifty имеют изменения в основе (сравните: two, three, five), а в числительном forty выпадает буква u.

1. В составных числительных единицы непосредственно присоединяются к десяткам и пишутся через дефис, а между десятками и сотнями стоит союз and: thirty-seven, eight hundred and fifty-four.
2. Перед числительными hundred, thousand, million ставится неопределенный артикль а или числительное one; a (one) hundred, a (one) thousand, a (one) million.

5. Числительные **hundred, thousand** и **million** не принимают окончания **-s**,  
когда перед ними стоят числительные **two, three, four** и т. д.: **two hundred, three  
thousand, four million.**

**Hundred, thousand** и **million** могут, однако, принимать окончание **-s**, когда они выражают неопределенное количество сотен, тысяч, миллионов. В этом слу­чае они превращаются в существительные и после них употребляется существи­тельное с предлогом **of**:

Hundreds of students were present at the Сотни студентов присутствовали на

meeting. собрании.

Thousands of people met the Russian Тысячи людей встречали президента

President. России.

6. Порядковые числительные, кроме **first, second, third,** образуются от соответ-ствующих количественных числительных путем прибавления суффикса -**th** или -**eth**.

Суффикс -**eth** прибавляется к числительным, оканчивающимся на -**ty**, причем буква у при этом заменяется на **i**: thirt**y** — thirt**i**eth, sixt**y** — sixt**i**eth, etc.

В остальных случаях прибавляется суффикс -**th**: six — six**th**, seven — seven**th**, hundred — hundredth, etc.

Составные порядковые числительные образуются от соответствующих коли­чественных числительных путем замены последнего количественного числительно­го порядковым: twenty-one — twenty-**first**, thirty-seven — thirty-**seventh**.

**Числительные (Numerals)**

|  |  |
| --- | --- |
| Количественные (Cardinal Numerals) | Порядковые (Ordinal Numerals) |
| 1 one [w∧n] | the first [ ðə fə**:**st] |
| 2 two [tu**:**] | the second [ðə 'sekənd] |
| 3 three [ θr i**:**] | the third [ ðə ' θə**:**d] |
| 4 four [fo**:**] | the fourth [ ðə fo**:** θ] |
| 5 five [faIv] | the fifth [ ðə f If θ] |
| 6 six [sIks] | the sixth [ ðə s Iksθ] |
| 7 seven [ 'sevən] | the seventh [ ðə 'sevnθ] |
| 8 eight [eIt] | the eighth [ ði**:** 'eItθ] |
| 9 nine [naI n] | the ninth [ ðə 'naI nθ] |
| 10 ten [ten] | the tenth [ ðə tenθ] |
| **11 eleven [I'levən]** | **the eleventh [ði: I'levənθ]** |
| 12 twelve [twelv] | the twelfth [ ðə 'twelfθ] |
| **13 thirteen [ 'θə**:'ti:**n]** | the thirteenth [ ðə 'θə**:**' ti:nθ] |
| **14 fourteen** ['fo**:'ti:n]** | the fourteenth [ ðə "fo**:**'ti:nθ] |
| **15 fifteen** ['f **I**f **'ti: n]** | the fifteenth [ ðə 'f If'ti:.nθ] |
| 16 sixteen ['sIks'ti:n] | the sixteenth [ ðə 'sIks'ti**:**nθ] |
| **17 seventeen ['sevn'ti:n]** | the seventeenth [ ðə 'sevn'ti**:**nθ] |
| **18 eighteen [eI 'ti:n]** | the eighteenth [ ði**:** 'eI'ti**:**nθ] |
| **19 nineteen ['naI n'ti:n]** | the nineteenth [ ðə 'naI n'ti**:**nθ] |
| 20 twenty ['twent I] | the twentieth [ ðə 'twentIIθ] |
| 21. twenty-one ['twentI 'w∧n] | the twenty-first [ ðə 'twentI 'fə:st] |
| 30. thirty [ 'θə:tI] | the thirtieth [ ðə 'θə:tII θ] |
| 40. forty ['fo:t I] | the fortieth [ ðə 'fo:tI Iθ] |
| 50. fifty ['f Ift I] | the fiftieth [ ðə 'f Ift I Iθ] |
| 60. sixty ['sIkst I] | the sixtieth [ ðə 'sIkstIIθ] |
| 70. seventy ['sev əntI] | the seventieth [ ðə 'sevnt I Iθ] |
| 80. eighty ['e It I] | the eightieth [ ði**:** 'eItIIθ] |
| 90. ninety [ 'naI ntI] | the ninetieth [ ðə 'naI nt I Iθ] |
| 100. one hundred ['w∧n 'h∧ndr Id] | the hundredth [ **ðə** 'h**∧**ndr **I**d**θ**] |
| 101. one hundred and one  ['w∧n 'h∧ndr Id ənd 'w∧n] | the one hundred and first  [ **ðə** 'w**∧**n'h**∧**ndr **I**d  **ə**nd 'f:st] |
| 110. one hundred and ten  ['w**∧**n 'h**∧**ndr **I**d  **ə**nd 'ten] | the one hundred and tenth  [ **ðə** 'w**∧**n 'h**∧**ndr **I**d **ə**nd 'ten**θ**] |
| 1,000 one thousand ['w**∧**n ' **θ**auz**ə**nd] | the thousandth [ ðə ' θauzəndθ] |
| 1,000,000 one million ['w**∧**n 'm **I**lj**ə**n] | the millionth [ ðə 'mIliənθ] |

Местоимение (The Pronoun)

В английском языке местоимения по своему значению делятся на 9 групп. Каждая группа местоимений имеет свои особенности, как по значению, так и по синтаксическим и морфологическим функциям.

Ниже будут рассматриваться лишь те группы местоимений, которые встреча­ются в данном учебнике.

**Личные местоимения (Personal Pronouns)**

Единственное числоМножественное число

1-е лицо I [aI] я we [wi:] мы

2-е лицо you [ju:] ты you [ju:] вы

3-е лицо he [hi:] он they [ðeI] они

she [∫i:] она

it [It] он, она, оно (о не­одушевленных предметах и

животных)

1. Личное местоимение I всегда пишется с заглавной буквы.

1. В современном английском языке имеется только одно личное местоимение для 2-го лица единственного и множественного числа — you, которое пишется с маленькой буквы. На русский язык **you** переводится «вы» и «ты» в зависимости от контекста.
2. Личные местоимения имеют только два падежа: именительный и объект­ный. Объектный падеж личных местоимений переводится на русский язык одним из косвенных падежей.

*Именительный падеж Объектный падеж*

I me [mi**:**]

you you [ju**:**]

he him [hI m]

she her [hə**:**]

it its [Its]

we us [∧s]

you you [ju**:**]

they them [ðem]

**Притяжательные местоимения (Possessive Pronouns)**

Притяжательные местоимения выражают принадлежность. Каждое личное местоимение имеет соответствующее притяжательное местоимение.

Существуют две формы притяжательных местоимений: первая форма (**Con­joint form**), определяющая существительное, и вторая форма (**Absolute form**), заменяющая существительное.

**Притяжательные местоимения первой формы**

Единственное число Множественное число

1-е лицо my [maI] мойour [аuə] наш

2-е лицо your [jo**:**] твой your [jo**:**] ваш

3-е лицо his [hIz] егоtheir [ðεə] их

her [hə**:**] ее

its [its] его, ее (для неоду­шевленных предметов)

**Притяжательные местоимения второй формы**

Единственное число Множественное число

1-е лицо mine [maI n]ours [auəz]

2-е лицо yours [jo**:**z]yours [jo**:**z]

3-е лицо his [hIz]theirs [ðεəz]

hers [hə**:**z]

its [its]

Притяжательное местоимение первой формы в предложении имеет функцию прилагательного — определения к существительному:

Please give me **my** pen.Пожалуйста, дайте мне **мою** ручку.

**His** story is too long. **Его** рассказ слишком длинен.

Don't take this chair, **its** leg is broken. He берите этот стул, **его** ножка сломана. I have not seen **their** wall newspaper. Я не видел **их** стенной газеты.

Притяжательные местоимения второй формы в предложении заменяют сущест­вительные в функции подлежащего, части составного сказуемого и дополнения:

My room is large, **yours** is larger and Моя комната большая, **ваша** — больше,

**hers** is the largest.а **ее** — самая большая.

Whose school did he visit? He visited Какую школу он посещал? Он посещал

**ours**. **нашу**.

Whose book is it? It is **mine**.Чья это книга? **Моя**.

**Возвратные и усилительные местоимения (Reflexive and Emphatic Pronouns)**

Возвратные местоимения образуются путем прибавления окончания -**self** к личным или притяжательным местоимениям единственного числа и окончания -**selves** к личным или притяжательным местоимениям множественного числа.

Возвратным местоимениям в русском языке соответствует частица -**ся**(-**сь**) в возвратных глаголах и местоимение **себя**, **себе**.

*Единственное число Множественное число*

1-е лицо myself [maI'self] ourselves [auə'selvz]

2-е лицо yourself [jo**:**'self] yourselves [jo**:'**selvz]

3-е лицо himself [hIm'self] themselves [ðem'selvz]

herself [hə**:**'self]

itself [It'self]

I dressed **myself**. Я одел**ся**.

He washes **himself** with cold water. Он умывает**ся** холодной водой.

She said to **herself**. Она сказала **себе**.

I **myself** saw the man. Я **сам** видел этого человека.

**Местоимения many, much, (a) few, (a) little**

В английском языке существует группа неопределенных местоимений: **many, much, few, a few, little, a little.**

Неопределенные местоимения many (много, многие), **few** (мало), **a few** (не­много, немногие, несколько), **much** (много), **little** (мало), **a little** (немного, немнож­ко), выражающие неопределенное количество, различаются в своем употребле­нии.

Местоимения **many**, **few**, **a few** определяют или заменяют исчисляемые сущест­вительные во множественном числе. Эти местоимения отвечают на вопрос: **How many?** (Сколько?)

She bought **many** pencils for her son. Она купила **много** карандашей сыну.  
**Few** students were in the room.В комнате было **мало** студентов.

I saw **a few** buses in the village.В поселке я увидел **несколько** автобусов.

Местоимения **much, little, a little** определяют или заменяют неисчисляемые существительные (имеющие только единственное число). Эти местоимения отве­чают на вопрос: **How much?** (Сколько?)

I don't like to eat **much** bread.Я не люблю есть **много** хлеба.

My brother puts **little** sugar in his tea. Мой брат кладет **мало** сахара в чай.  
Pass me **a little** fish, please.Передайте мне, пожалуйста, **немного**

рыбы.

Слова **much, little, a little** могут относиться не только к существительным, но также к глаголу, причастию и прилагательному и в этом случае имеют значение наречий.

Ann writes not so **much**. Аня пишет не очень **много**.

He looked **a little** tired. Он выглядел **немного** усталым.

Слова **a few**, **a little** имеют положительное значение, обозначая наличие не­большого количества, в то время как **few** и **little** подчеркивают недостаточность количества, т. е. имеют отрицательное значение.

The student has **few** English books.У этого студента **мало** английских книг.

This student has **a few** English books. У этого студента есть **несколько** ан­глийских

книг.

Глагол **(The** Verb)

Времена группы **Indefinite (Indefinite (Simple) Tenses)**

Все времена группы Indefinite употребляются для выражения факта совер­шения действия, а также для выражения обычных, регулярно повторяющихся действий в настоящем, прошедшем и будущем.

**Настоящее** неопределенное время (The Present Indefinite **(Simple)** Tense)

Настоящее неопределенное время (Present Indefinite **(Simple)**) образуется из основной формы глаголы (инфинитива спрягаемого глагола без частицы **to**) для всех лиц, кроме 3-го лица единственного числа.

Для образования 3-го лица единственного числа к основной форме глагола прибавляется окончание -(e)s. Ниже приводим спряжение глагола write в утверди­тельной форме:

I write.We write.

You write.You write.

He (she, it) writes They write.

Окончание -s произносится как звук **[s]** после глухих согласных и как звук [z] после звонких согласных и гласных:

Не works. [hi**:** 'wə**:**ks]

Не sees. [hi**:** 'si**:**z]

He learns. [hi**:** 'lə**:**nz]

Порядок слов в утвердительном предложении

|  |  |  |
| --- | --- | --- |
| Подлежащее | Сказуемое | Остальная часть предложения |
| I | speak | English every day. |

Вопросительная форма образуется при помощи вспомогательного глагола do (does в 3-м лице единственного числа), который ставится перед подлежащим, за ним следует подлежащее и спрягаемый глагол в основной форме. Если в пред­ложении имеется вопросительное слово, то оно ставится в самом начале пред­ложения.

**Порядок слов в вопросительном предложении**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Вопросительное слово | Вспомогательный глагол | Подлежащее | Смысловой глагол | Остальная часть предложения |
| Where | does  do | she  you | live  work | in summer?  in the morning? |

Отрицательная форма образуется из утвердительной путем постановки вспо­могательного глагола **do (does)** и отрицания not перед спрягаемым глаголом в ос­новной форме, например:

Не does not go to school on Sundays. Он не ходит в школу по воскресеньям.

Порядок слов в отрицательном предложении

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Сказуемое | | | Остальная часть предложения |
| Подлежащее | вспомогательный глагол | oтрицание | смысловой глагол |  |
| We | do | not | go | to the cinema every day. |

*Примечание*: **Present Indefinite** **(Simple)** употребляется также для выражения буду­щего действия в придаточных обстоятельственных предложениях времени и усло­вия:

**If** you **study** English hard, you **will learn** it. Если вы **будете заниматься** английским

языком упорно, вы **выучите** его.

**When** we go to the theatre, I **shall meet** you Когда мы **пойдем** в театр, я **встречу** тебя

at the bus stop. у автобусной остановки.

**Прошедшее неопределенное время (The Past Indefinite (Simple)Tense)**

В английском языке все глаголы в зависимости от способа образования вре­мени **Past Indefinite (Simple)** делятся на две группы: правильные глаголы и неправильные. Формы **Past Indefinit**e **(Simple)** глаголов, которые образуются не по правилу, приводятся в таблице неправильных глаголов (см. с. 354), и их следует выучить. Все правиль­ные глаголы образуют **Past Indefinite (Simple)** путем добавления окончания -**ed** к основной форме спрягаемого глагола. Это окончание произносится как звук **[d],** если оно следует за звонким согласным и гласным, как **[t],** если оно следует за глухим со­гласным, и как [Id], если основа спрягаемого глагола оканчивается на звуки **[t]** и [d]:

ask [α:sk]ask**ed** [α:sk**t**]

live [lIv]liv**ed** [lIv**d**]

count [kaunt]count**ed** ['kauntId]

decide [dI'saId]decid**ed** [dI'saIdId]

Глаголы в Past Indefinite **(Simple)** ни по лицам, ни по числам не изменяются.

Отрицательная и вопросительная формы в Past Indefinite **(Simple)** строятся по тому же принципу, что и в Present Indefinite **(Simple)**. Однако надо помнить, что вспомогательный глагол в Past Indefinite **(Simple)** имеет форму did, например:

Did Ann go to the park yesterday? Аня ходила в парк вчера?

We did not see the film yesterday. Мы не видели этот фильм вчера.

Рекомендуем запомнить наиболее распространенные обстоятельства времени, употребляемые с глаголами в Past Indefinite **(Simple)**:

yesterday вчера

at that time в то время

the day before yesterday позавчера

last week (month, year) на прошлой неделе (в прошлом месяце, году)

two days ago два дня тому назад

Будущее неопределенное время (The Future Indefinite **(Simple)**Tense)

Будущее неопределенное время (Future Indefinite **(Simple)**) образуется из вспомога­тельных глаголов shall и will и основной формы спрягаемого глагола.

Вспомогательный глагол shall употребляется в 1-м лице единственного и множественного числа, а в остальных лицах употребляется will:

I shall write. We shall write.

You will write. You will write.

He (she, it) will write. They will write.

Вопросительная форма строится путем постановки вспомогательного глагола shall или will перед подлежащим, например:

Will you go to the cinema tomorrow? Ты пойдешь в кино завтра?

Отрицательная форма образуется путем постановки отрицания not после вспомогательного глагола shall или will, например:

I shall not **gо** to the cinema tomorrow. Я *не* пойду в кино завтра.

Времена группы Continuous (Progressive) (Continuous (Progressive)Tenses)

Настоящее продолженное время (The Present Continuous (Progressive) Tense)

Present Continuous (Progressive) образуется при помощи вспомогательного глагола to be в настоящем времени (am, is, are) и причастия настоящего времени смыслового глагола:

I am reading.

Не is reading.

We are reading.

Вопросительная форма образуется путем постановки вспомогательного гла­гола to be (am, is, are) перед подлежащим, например:

Am I reading?

Is he reading?

Are they reading?

Отрицательная форма образуется путем постановки отрицания not после вспомогательного глагола, например:

I am not reading.

Не is *not* reading.

We are not reading.

Present Continuous (Progressive) употребляется для выражения действия, происходящего в данный момент настоящего времени:

I am speaking now. Я разговариваю сейчас.

The boy is running. Мальчик бежит.

They are doing their laboratory work. Они делают лабораторную работу.

Прошедшее длительное время (The Past Continuous (Progressive) Tense)

Past Continuous (Progressive) образуется при помощи вспомогательного глагола to be в прошедшем времени (was, were) в сочетании с причастием настоящего времени смыслового глагола:

I was speaking.

They were speaking.

Вопросительная форма образуется путем постановки вспомогательного гла­гола to be (was, were) перед подлежащим, например:

Was I speaking?

Were they speaking?

Отрицательная форма образуется путем постановки отрицания not после вспомогательного глагола, например:

I was *not* speaking.

They were *not* speaking.

Past Continuous (Progressive) употребляется:

1) когда точно указано время (или отрезок времени) совершения действия в  
прошлом:

I was writing a letter at 8 o'clock Я писал письмо в 8 часов вчера вечером.

yesterday.

2) когда совершено два действия в прошлом и одно из них имеет продолженную форму (Past Continuous (Progressive)), а другое — законченную форму(Past Indefinite (Simple)):

Pete was going to school when he met Петя шел в школу, когда он встретил  
his friend. своего друга. (Петя шел в школу до встречи с другом и после, поэтому это действие

pассматривается как длительное и выражается глаголом в Past Continuous (Progressive))

She was reading when her mother en- Она читала, когда вошла мама. (Она читала до  
tered the room. прихода мамы в течение какого-то времени,

поэтому это дей­ствие рассматривается как длитель­ное и выражается глаголом в Past Continuous (Progressive).

**Времена группы Perfect (Perfect Tenses)**

Настоящее совершенное время (The Present Perfect Tense)

Present Perfect образуется при помощи вспомогательного глагола to have в настоящем времени (have, has) и причастия прошедшего времени смысло­вого глагола:

I (we, you, they) **have worked**.

He (she, it) **has worked.**

Вопросительная форма образуется путем постановки вспомогательного гла­гола to have перед подлежащим, например:

**Have** I **worked**?

**Has** he **worked**?

Отрицательная форма образуется путем постановки отрицания not после вспомогательного глагола, например:

I **have** ***not*** **worked**.

Не **has** ***not*** **worked**.

**Present Perfect** употребляется для выражения закончившегося действия, когда время совершения его указывается неопределенно, а результат действия налицо в настоящий момент. (В тех случаях, когда время совершения действия в прошлом указывается определенно, употребляется Past **Indefinite**(Simple). **Present** Perfect чаще всего переводится на русский язык глаголами совершенного вида в про­шедшем времени:

I **have written** a letter.Я написал письмо. (Письмо написано,

но не сказано, когда оно написано, поэтому употребляем **Present Per­fect**.)

Мы купили телевизор. (Телевизор куплен.  
We **have bought** a TV set.Он в комнате. Нас интересует результат в   
 настоящем, а время совершения действия

не указано.)

Pete **has broken** his pencil.Петя сломал карандаш. (Результат действия налицо:карандаш сломан и им писать нельзя. Время совер­шения действия не указано.)

**Present Perfect** часто употребляется с наречиями неопределенного времени, которые ставятся между вспомогательным глаголом и причастием прошедшего времени спрягаемого глагола:

Ann **has** ***just*** **come** in.Аня ***только что*** **вошла**.

We **have** ***already*** **seen** this film.Мы ***уже*** **видели** (**посмотрели**)этот фильм.

Наиболее часто употребляемые наречия неопределенного времени:

ever [ 'evə] когда-либо

never ['nevə] никогда

just [dʒ∧st] только что

already [o**:**l'redI] уже (not)

(not) yet [(nоt) jet] еще (не)

Прошедшее совершенное время (The Past Perfect Tense)

Past Perfect образуется при помощи вспомогательного глагола to have в прошедшем времени (had) и причастия прошедшего времени смыслового глагола:

I **had** **worked**.

Не **had** **worked**.

Вопросительная форма образуется путем постановки вспомогательного гла­гола to have (had) перед подлежащим, например:

**Had** I **worked**?

**Had** he **worked**?

Отрицательная форма образуется путем постановки отрицания not после вспомогатель-ного глагола, например:

I **had** ***not*** **worked**.

Не **had** ***not*** **worked**.

Past Perfect употребляется для выражения действия, совершенного к ка­кому-то моменту в прошлом. Этот момент может быть обозначен либо точным указанием времени, либо другим действием в прошлом:

Не **had written** his work by 2 o'clock.Он написал свою работу к двум часам.

We **had finished** our laboratory workМы закончили нашу лабораторную ра-

before the bell rang.боту до того, как прозвенел звонок.

Часто Past Perfect употребляется в придаточных предложениях, когда мы хотим подчеркнуть предшествование одного действия другому:

She **said** (that) she had **read** this Она сказала, что читала эту статью,  
 article.(Сначала она прочла статью, а

потом об этом сказала.)

**Правило согласования времен (The Sequence of Tenses)**

В английском языке существует правило согласования (последовательности) времен в сложноподчиненных предложениях с придаточными дополнительными.

Если в главном предложении глагол стоит в настоящем или будущем време­ни, то в придаточном дополнительном предложении глагол может стоять в любом времени, требуемом по смыслу, и в этом случае правило согласования времен не действует:

Не **says** that Ann **takes** English lessons. Он **говорит**, что Аня **берет** уроки анг­лийского

языка.

Не **says** that they **were** good friends Он **говорит**, что они **были** хорошими

for many years.друзьями в течение многих лет.

Не **says** that they **will go** to Riga in Он **говорит**, что они **поедут** в Ригу летом.  
summer.

Правило согласования времен действует лишь в том случае, если в главном предложении глагол стоит в Past Indefinite (Simple) или Past Continuous (Progressive). В этом случае в придаточном предложении употребляется:

1. для выражения одновременного действия Past Indefinite (Simple) или Past Continu­ous (Progressive):

He said that Ann took English lessons. Он сказал, что Аня берет уроки анг­лийского языка.

2) для выражения действия, предшествующего действию главного предложения — Past Perfect:

Не said that they had beengood friends Он сказал, что они были (когда-то раньше

for many years. хорошими друзьями в течение многих лет.

3) для выражения будущего действия употребляется особое время, которое  
называется Future **(Simple)** in the Past (будущее в прошедшем):

Не said that they would go to Riga Он сказал, что они поедут в Ригу летом.

in summer.

Не said that we should learn English. Он сказал, что мы будем изучать

английский язык.

*Примечание*: Future **(Simple)** in the Past образует утвердительную, вопросительную и отрицательную формы таким же образом, как и Future Indefinite (Simple). Разница заключается лишь в том, что в Future **(Simple)** in the Past вместо глаголов shall и will соответственно употребляются should и would. На русский язык предложения в Future **(Simple)** in the Past переводятся будущим временем. Помните, что Future **(Simple)** in the Past употребляется только в придаточных дополнительных предложениях:

She knew that I (we) should attend Она знала, что я буду посещать эти лекции.

these lectures.

She knew that you (he, she, they) Она знала, что вы будете посещать эти лекции.

would attend these lectures.

**Страдательный залог (The Passive Voice)**

В английском языке существует два залога глагола: действительный (the Active Voice) и страдательный (the Passive Voice). В том случае, когда подле­жащее является действующим лицом, мы имеем дело с действительным залогом, например:

I read a newspaper every day. Я читаю газету каждый день. *(Подлежащее* I *является действующим ли­цом, и поэтому мы говорим, что данное предложение стоит в действи­тельном залоге.)*

Если же подлежащее не является действующим лицом, а действие направлено на подлежащее, то в этом случае мы имеем дело с предложением в страда­тельном залоге, например:

The book was written by my friend. Книга была написана моим другом.

(Здесь действие направлено на под­лежащее the

book.)

Страдательный залог образуется при помощи вспомогательного глагола to beв соответст вующем времени, лице и числе в сочетании с Past Participle (причас­тием прошедшего времени) смыслового глагола.

Формула образования страдательного залога:

|  |  |
| --- | --- |
| to be | Past Participle |
| (в соответствующем времени, | + смыслового глагола |
| лице и числе) |  |
| Например: |  |
| The house was built last month. | Дом был построен в прошлом месяце. |
| The teacher was sent for. | З За учителем послали. |
|  |  |

Как видно из вышеприведенных примеров, глагол в страдательном залоге в английском языке можно переводить на русский язык следующими тремя способами:

1. сочетанием глагола «быть» с краткой формой страдательного причас­тия — в этом сочетании глагол to be в настоящем времени на русский язык не переводится;
2. глаголом, оканчивающимся на -ся (-сь);
3. глаголом в действительном залоге в 3-м лице множественного числа в составе неопределенно-личного предложения.

Страдательный залог употребляется:

1) когда нам неизвестно лицо, совершающее действие, или по каким-то причинам  
 мы не хотим упоминать его:

A lot of books are published in our Очень много книг ежегодно издается  
country every year. в нашей стране.

2) когда предмет действия представляет для нас больший интерес, чем действующее лицо:

The picture was bought yesterday. Картина была куплена вчера.

**Повелительное наклонение (The Imperative Mood)**

Повелительное наклонение выражает просьбу или приказание говорящего. В повелительном наклонении различаются утвердительная и отрицательная формы.

Утвердительная форма глагола образуется из инфинитива без частицы to:

Read! Читай! Читайте!

Write! Пиши! Пишите!

Look at the blackboard! Смотри (смотрите) на доску!

Отрицательная форма глагола *в* повелительном наклонении образуется пу­тем постановки вспомогательного глагола do*,* слитого в одно словос отрица­тельной частицей not — don't:

Don't write now! Не пишите сейчас!

Don't be late! Не опаздывайте!

Don't talk! Не разговаривайте!

Для выражения просьб или приказаний, обращенных к 3-му лицу, употреб­ляется сочетание глагола let с личным местоимением вформе объектного падежа (или существительным вформе общего падежа) и инфинитива без частицы to. Эти сочетания переводятся на русский язык при помощи слов «пусть», «пускай», «давай», «дайте» с местоимением (существительным) и гла­голом:

Let him translate! Пусть (пускай) он переводит!

Дай (-те) ему перевести!  
Let them translate! Пусть (пускай) они переводят!

Дай (-те) им перевести!

Сочетание глагола let с личным местоимением 1-го лица множественного числа в форме объектного падежа и инфинитива без частицы to выражает побуждение к совместному действию:

Let us go! Пойдем (пойдемте)! (Давай (-те) пойдем!)

Let us talk! Поговорим! (Давай (-те) поговорим!)

Отрицательная форма образуется при помощи do not (don't):

Don't let him do it! Пусть он этого не делает!

Don't let them go there! Пусть они не ходят туда!

Глагол to be

Формы глагола to be в настоящем времени (Present Indefinite (Simple))

|  |  |  |  |
| --- | --- | --- | --- |
| Утвердительная форма | Вопроси­  тельная  форма | Отрицательная  форма | Вопросительно-отри­  цательная форма |
| I am  Не (she, it) is  We (you, they) are | Am I?  Is he?  Are we? | I am not  He is not  We are not | Am I not?  Is he not?  Are we not? |

Глагол to be употребляется в качестве смыслового глагола, глагола-связки в составном именном сказуемом и вспомогательного глагола.

1. В роли смыслового глагола to be соответствует в русском языке глаголам «быть»,  
«находиться». В этом случае в настоящем времени to be на русский язык часто не  
переводится, например:

Ann is at school. Аня … в школе.

2. В роли глагола-связки в составном именном сказуемом to be соответствует в русском языке глаголам «быть», «являться», «состоять», «заключаться».  
В настоящем времени связка на русский язык тоже часто не переводится:

I am a student. Я … студент. (Я являюсь студентом.)

3. В роли вспомогательного глагола to be употребляется для образования всех времен  
группы Continuous и страдательного залога (Passive Voice). В этом случае to be не имеет  
самостоятельного значения, а выполняет лишь различные грамматические функции, являясь показателем времени, лица, числа, залога и т. д. Само же действие выражается смысловыми глаголами:

Не is sitting at the table now. Он сидит за столом сейчас.

The house was built last month. Дом был построен в прошлом месяце.

Оборот there is (there are)

В сочетании с вводным словом there глагол to be в личной форме образует оборот there is (there are) со значением «есть», «имеется», «находится».

Глагол to be в этом обороте может стоять в настоящем, прошедшем и будущем времени (Present, Past, Future Indefinite (Simple)) и согласуется с последую­щим существительным.

На русский язык предложения с оборотом there is (there are) рекомен­дуется перево-дить начиная с обстоятельства места:

There is a bus *in the street*. *На улице* (есть, находится, стоит) автобус.

There is a book *on the table.* *На столе* … книга.

There are two pens *on the desk*. *На парте* … две ручки.

There is a man *there*. *Там* … человек.

В вопросительной форме глагол to be ставится перед вводным словом there:

Is there a pencil in your bag? В твоем портфеле есть карандаш?

Are there any cars in the street? На улице стоят машины?

Для образования отрицательной формы отрицание nо ставится после оборо­та there is (there are):

There is no book on the table. На столе нет книги.

There are no buses in the street. На улице нет автобусов.

Однако в кратких отрицательных ответах используется отрицание not:

Is there a clock on the table? На столе есть часы?

No, there is not. Нет.

Are there any exercise-books in your Есть ли у вас в портфеле тетради?

bag?

No, there are not. Нет.

Глагол to have

Формы глагола to have в настоящем времени (Present Indefinite (Simple))

|  |  |  |  |
| --- | --- | --- | --- |
| Утвердительная форма | Вопроситель­ная форма | Отрицательная форма | Вопросительно-отрицательная форма |
| I (you, we, they) have He (she, it) has | Have I?  Has he? | I have not  He has not | Have I not?  Has he not? |

*Примечание*: В отрицательной форме глагол to have может принимать отри­цание nо. Например: I have no books. У меня нет (никаких) книг. Но: I haven't the book. У меня нет этой книги.

Глагол to have употребляется в качестве смыслового, вспомогательного и модального глагола (заменитель глагола must [m∧st] — должен).

1. В роли смыслового глагола to have соответствует в русском языке глаголам «иметь», «**обладать**», например:

I have a book. **У** меня **есть** книга. (Я **имею** книгу.)

She has a watch. **У** нее **есть** часы. (**Она** **имеет** часы:)

2. В роли вспомогательного глагола to have употребляется для образования времен группы Perfect в сочетании с причастием прошедшего времени (Participle II).

Глагол to have в этом случае самостоятельного значения не имеет, а вы­полняет лишь различные грамматические функции, являясь показателем времени, лица, числа и т. д. Само же действие выражается смысловым глаголом, в соче­тании с которым он употребляется, например:

I have read this book.Я **прочел** эту книгу.

Не has bought a watch.Он **купил** часы.

3. В роли модального глагола to have употребляется как заменитель глагола  
must, например:

You will **have** to come here again.Вам **придется** снова прийти сюда.

**Модальные глаголы (Modal Verbs)**

В английском языке имеется группа глаголов, которые называются модаль­ными или недостаточными. В эту группу входят глаголы:

**can** [kæn] могу, умею

**may**  [meI] могу, мне позволено, разрешено

must [m∧st] должен

Недостаточными они называются потому, что имеют только некоторые лич­ные формы и совсем не имеют неличных форм (инфинитива, причастий и ге­рундия), а поэтому не имеют никаких сложных глагольных форм.

Ни один из этих глаголов не имеет окончания **-s** в 3-м лице единственного числа настоящего времени (Present Indefinite). Самостоятельно эти глаголы не употребляются. За ними всегда следует другой глагол в инфинитиве без частицы to:

I **can … speak** English.Я **умею** **говорить** по-английски.

You **may … sit** down.Вы **можете** **сесть** (я вам **разрешаю**).

Глагол **саn**

Глагол сап имеет две формы: 1) форму настоящего времени: I can, he **саn**; 2) форму прошедшего времени: I **could**, he **could**.

Для выражения будущего времени или каких-либо других временных форм глагол саn не употребляется, а употребляется выражение be able to:

Не will be able to read English books Он **сможет** читать английские книги in a month.через месяц.

Глагол сап выражает физическую или умственную способность совершения действия, а также умение совершить какое-либо действие

She can run quickly.Она **может** быстро бегать.

Не can skate.Он **умеет** кататься на коньках.

Can you play tennis?Вы **умеете** играть в теннис?

Yes, I саn. Да.

No, I **can’t** (**cannot)**.Нет.

Глагол may

Глагол **may** имеет также две формы: 1) форму настоящего времени: I **may**, he **may**;

2) форму прошедшего времени: I **might**, he **might**.

Для выражения будущего времени или каких-либо других временных форм глагол **may** не употребляется, а употребляется выражение **be allowed to**:

Ann **will be allowed to** pass her exam Ане **будет разрешено** сдать экзамен по химии.  
in chemistry.

Глагол **may** имеет значение возможности совершения действия в зависимости от разрешения или вероятности его совершения:

**May** I take the book? **Можно** мне взять книгу?

Yes, you **may**.Да, **можете** (я разрешаю).

No, you **can't**.Нет, **нельзя**.

It **may** snow today. **Возможно**, сегодня будет снег.

Глагол **must**

Глагол **must** имеет только однуформу — настоящего времени: I **must**, he must.

We **must** do our lessons every day. Мы **должны** делать уроки ежедневно.

**Must** he go there? **Должен** он пойти туда?

Yes, he **must**.Да.

No, he **needn't**.Нет.

Для выражения прошедшего времени или будущего времени употребляется его заменитель — глагол **have to**:

She **had to** do this work. Она **должна была** (ей **пришлось**)

сделать эту работу.

Не **will have to** read this English book. Он **должен будет** (ему **придется**) чи­тать

эту английскую книгу.

**Инфинитив (The Infinitive)**

Как в русском, так и в английском языке инфинитив имеет свойства глагола и существи- тельного.

Инфинитив обычно употребляется с частицей **to**.

**Черты существительного**

1. Инфинитив может быть подлежащим в предложении:

**То study** well is your duty. **Учиться** хорошо — ваша обязанность.

**То speak** English is not difficult. **Говорить** по-английски нетрудно.

2.Инфинитив может быть прямым дополнением:

Our students like to read. Наши студенты любят читать.

My sister asked me to go there with her. Моя сестра попросила меня пойти туда

с ней.

3. Инфинитив может быть именной частью составного сказуемого:  
 Your task ***is*** **to study** well.Ваша задача — **учиться** хорошо.

**Характеристики глагола**

1.За инфинитивом может следовать прямое дополнение:

Не likes to read English books. Он любит читать английские **книги**.

2.Инфинитив может определяться наречием:

They were surprised **to see** me so early. Они удивились, **увидев** меня **так рано**.

3.Инфинитив может быть частью составного глагольного сказуемого:

Не ***must*** **do** his homework this evening. Он ***должен*** **делать** домашнее задание

вечером.

Ann *began* **to** read English books.Аня ***начала*** **читать** английские книги.

Кроме того, инфинитив имеет еще и другие синтаксические функции.

1.Инфинитив может быть определением, заменяющим целое придаточное опреде-  
лительное предложение.

My friend has brought me an interest- Мой друг принес мне интересную  
ing book **to** read. книгу, **которую я могу почитать.**

2.Инфинитив может быть обстоятельством цели, заменяющим обстоятельст-  
венное предложение цели:

Не gave me his реп **to** write a letter Он дал мне ручку, **чтобы я написал**  
to Mike. **письмо Мише.**

3.Инфинитив может входить в состав сложного дополнения:

I want ***you*** **to** come early today.Я хочу, ***чтобы*** ***ты*** **пришла** рано сегодня.

Герундий (The Gerund)

В английском языке есть неличная форма глагола, оканчивающаяся на -ing, которой нет ни в русском языке, ни в казахском. Эта ing-форма называется the Gerund (герундий). Герундию присущи характеристики как глагола, так и существительного.

**Характеристики существительного**

1. Герундий употребляется в роли подлежащего или дополнения:

Reading is useful. **Чтение** полезно.

I like reading. Я люблю **читать**.

2.Герундий может иметь при себе определение, выраженное притяжательным  
местоимением или существительным:

We were pleased ***with*** Alec's coming. Мы были довольны, ***что*** **пришел Алик**

(**приходом Алика**).

3.В роли определения или обстоятельства герундий всегда употребляется  
с предлогом:

I don't like his ***way*** **of** doing it. Мне не нравится, ***как*** **он это делает**.

**Характеристики глагола**

1.Герундий определяется наречием:

I like your **speaking** English ***so*** *fast*. Мне нравится, что вы ***так быстро***

**говорите** по-английски.

2.Герундий может иметь при себе прямое дополнение:

She likes **reading** *French books*. Она любит **читать** ***французские книги***

***(книги на французском языке).***

Как видно из предыдущих примеров, герундий переводится на русский язык существительным, глаголом в неопределенной форме или придаточным предложением.

Причастие настоящего времени (Present Participle, или Participle I)

Present Participle может быть образовано от любого глагола (кроме модаль­ных и вспомогательных глаголов shall и will) путем прибавления окончания -ing, соответствующего русским окончаниям -ащ(-ящ)ий(ся), -ущ(-ющ)ий(ся):

stand**ing** — сто**ящий**

turn**ing** — враща**ющийся**

burn**ing** — гор**ящий**

Если глагол оканчивается на непроизносимое -**е**, то при образовании Present Participle **е** опускается:

smile — smil**ing**

write — writ**ing**

Если глагол состоит из одного слога, то при образовании Present Participle удваивается конечная согласная:

sit — si**tt**ing

run — running

swim — swi**mm**ing

Present Participle участвует в образовании времен группы Continuous в со­четании с личными формами вспомогательного глагола to be:

They are drawing now.Они чертят сейчас.

В предложении Present Participle может выступать в функции:

1) определения к существительному:

I like to see smiling faces. Я люблю видеть улыбающиеся лица.

2) обособленного определения, заменяющего придаточное определительное  
предложение:

The boy sitting at the table is her Мальчик, сидящий за столом, ее брат,.

brother.

(The boy who is sitting at the table (Мальчик, который сидит за столом,  
is her brother.) ее брат.)

3) обстоятельства (в этих случаях английское причастие настоящего времени переводится русским деепричастием):

She went out smiling. Она вышла улыбаясь.

Причастие прошедшего времени (Past Participle, или Participle II)

Форма Past Participle правильных глаголов образуется путем прибавления суффикса -ed к основной форме глагола, например:

decide — решать decided — решённый

Форма Past Participle неправильных глаголов образуется особыми способами (см. таблицу неправильных глаголов на с. *354*), например:

write — писать written — написанный

Формы Past Participle наиболее употребительных глаголов вам нужно вы­учить.

Past Participle участвует в образовании времен группы Perfect, страдательного залога:

I have just got tickets to the cinema. Я только что купил билеты в кино.

This article was written *by* my friend. Эта статья была написана мо*им* ` друг*ом*.

В предложении Past Participle может выступать в функции:

1) определения к существительному:

The broken cup was on the floor. Разбитая чашка лежала на полу.

2) обособленного определения, заменяющего придаточное определительное  
предложение:

Here is the letter received from Nick. Вот письмо, полученное от Коли.  
(Неге is the letter which I received (Вот письмо, которое я получил от Коли

from Nick yesterday.) вчера.)

Предлоги (Prepositions)

Ввиду того, что в английском языке почти отсутствуют падежи, предлоги показывают не только отношения между существительными и другими частями речи, но и выполняют функцию падежных окончаний.

Запомните значения основных предлогов английского языка:

1. Предлог of выполняет функцию русского родительного падежа:

London is the capital of Great Britain. Лондон — столица Великобритании.

2. Предлог to выполняет функцию дательного падежа при выражении отношения глагола к дополнению:

I gave the pen to my brother. Я отдал ручку брату. 3.Предлог about соответствует форме предложного падежа:

Не told me about that party. Он рассказал мне о том вечере.

4. Предлоги by, with выполняют функцию творительного падежа. By обозначает одушевленного деятеля, with — орудие действия:

We write with a pen. Мы пишем ручкой.

'Poltava' was written by Pushkin. «Полтава» была написана Пушкиным.

Помните, что один и тот же предлог может переводиться на русский язык по-разному. Например:

in

She was in the street. Она была на улице.

We shall meet in a month. Мы встретимся через месяц.

The work was done in May. Работа была сделана в мае.

at

She was at the lesson. Она была на уроке.

I worked at my homework. Я работал над моим домашним заданием.

The teacher was at his desk when I Учитель был за своим столом, когда

came in. я вошел.

At 7 o'clock we shall go to Sochi. В семь часов мы поедем в Сочи.

to

Не goes to school in the morning. Он ходит в школу утром.

"Come to the blackboard!" he said. «Подойди к доске!» — сказал он.

Today we shall go to the meeting. Сегодня мы пойдем на собрание.

Запомните значения следующих предлогов:

on [on] на in [I n] внутри

above [ə'b∧v] над near [nIə] около, возле

under ['∧ndə] под to [tu:] указывает движение к предмету

into [ 'I ntə] внутрь before [bI'fo:] перед, до

out of ['aut' əv] изнутри in front of [I n 'fr∧nt əv] перед

behind [bI 'haI nd] сзади from [frəm] указывает движение от

below [bI'lou] ниже предмета или с поверхности чего-либо

Порядок слов в английском предложении (Word Order)

В английском языке существует твердый порядок слов в предложении. Это объясняется тем, что в языке имеется очень небольшое количество граммати­ческих окончаний и словообразовательных суффиксов и функция слова в предло­жении определяется егоместом в предложении.

Необходимо запомнить следующий порядок слов для утверди­тельного предложения: на первом месте стоит подлежащее (с опреде­ляющими его словами), за ним следует сказуемое, за сказуемым — остальная часть предложения.

|  |  |  |
| --- | --- | --- |
| Подлежащее | Сказуемое | Остальная часть предложения |
| Kate | reads | English books. |

В вопросительном предложении обратный порядок слов: сказуемое или часть сказуемого стоит перед подлежащим. Специальный вопрос начинается с вопроси­тельного слова, а общий вопрос — со сказуемого или части сказуемого.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Вопросительное слово | Вспомога­тельный глагол | Подлежащее | Смысловой глагол | Остальная часть предложения |
| When | do | you | speak | English? |
| Where | do | you | learn | your English? |
| What | did | your friend | do | yesterday? |
|  | will | you | do | tomorrow? |

Если глагол-сказуемое предложения стоит в одном из сложных времен дейст­вительного или страдательного залога, то при образовании вопросительной формы первый вспомогательный глагол занимает место перед подлежащим, например:

John has read this book.

Has John read this book?

Отрицание **not** обычно ставится между вспомогательным и основным глаголом или после первого вспомогательного глагола в сложных глагольных формах, например:

I do **not** speak French.Я **не** говорю по-французски.

They have **not** seen the film yet.Они еще **не** видели этот фильм.

Прямое дополнение обычно стоит за сказуемым или за косвенным дополне­нием, например:

Take **the book**, please.Возьмите **книгу**, пожалуйста.

Ann told me **an interesting story**.Аня рассказала мне **интересную историю** (**рассказ**).

Определение чаще всего занимает место перед определяемым словом.

Обстоятельства могут следовать за дополнениями или стоять в начале предло­жения, например:

We shall play tennis **tomorrow**.Мы будем играть в теннис **завтра**.

**In the morning** we shall go to the park. **Утром** мы пойдем в парк.

**Вопросительные предложения (Interrogative Sentences)**

Наиболее распространенными типами вопросов в английском языке являются общие и специальные вопросы.

**Общие вопросы** (General Questions)

Общими вопросами называются вопросы, цель которых — получить от собе­седника подтверждение или отрицание высказанной в вопросе мысли. Эти вопросы требуют утвердительного или отрицательного ответа (например, **yes** или nо).

Общие вопросы, как правило, не содержат вопросительного слова. Они произносятся с повышающейся к концу предложения интонацией:

Are you a nurse?— Yes, I am.

Have you seen the film? — No, I haven't.

**Разделительные вопросы** (“Tail” Questions)

Особой разновидностью общих вопросов являются разделительные вопросы (**“Tail”** **Questions**), которые состоят из двух частей. Первая — повествовательное предложение в утвердительной или отрицательной форме, вторая — краткий об­щий вопрос, состоящий из подлежащего-местоимения и вспомогательного, или модального глагола, или глаголов **to be, to have** в зависимости от сказуемого первой части. Если первая часть утвердительная (когда ожидается ответ **yes**), то вторая — отрицательная; если первая часть отрицательная (когда ожидается ответ nо), то вторая не содержит отрицания.

Первая часть разделительного вопроса произносится с понижающейся инто­нацией, вторая часть — с повышающейся. Подобные вопросы соответствуют в русском языке вопросам с вводными словами: «Не так ли?», «Не правда ли?», «Правда?». Разделительные вопросы весьма употребительны в разговорной речи:

She is here, isn't she? Она здесь, не так ли?

She isn't here, is she? Ее здесь нет, правда?

You will come, won't you? Ты придешь, не правда ли?

Jim doesn't play tennis, does he?

You don't have to go there now, do you?

Bob spoke English very well, didn't he?

Your brother can repair radio sets, can't he?

You have such a book, haven't you?

На разделительные вопросы обычно даются ответы с краткими утвердитель­ными или отрицательными фразами:

Today is Monday, isn't it? — Yes, it is.

You don't like coffee, do you? — No, I don't.

**Специальные вопросы (Special Questions)**

Специальными вопросами называются такие вопросы, целью которых являет­ся не простое подтверждение или отрицание мысли, высказанной в вопросе, а получение какой-либо дополнительной информации:

What's your name? — My name is Pete.

Where are you going? — To the park.

Специальный вопрос ставится к какому-либо одному члену предложения, Он обязательно начинается с вопросительного слова, которое указывает, к какому именно члену предложения относится вопрос. Специальный вопрос произносится с понижающейся интонацией. Рекомендуем запомнить вопросительные слова, которые употребляются в специальных вопросах:

who [hu**:**] кто?

whom [hu:m] кого? кому?

whose [hu**:**z] чей?

what [wot] что? какой?

which [wIt∫] который?

when [wen] когда?

where [wεə] куда? где?

why [waI] почему?

how [hau] как?

how many ['hau 'menI] **сколько**?

how much ['hau 'm⋀t∫] **сколько**?

**Альтернативные** вопросы (Alternative Questions)

Альтернативные (выборочные) вопросы состоят из двух частей, соединенных союзом or. Эти вопросы по форме совпадают с общими, но предусматривают выбор между двумя возможностями и требуют таких же ответов, как и спе­циальные вопросы.

Первая часть таких вопросов произносится с повышающейся интонацией, вторая — с понижающейся:

Is this Ann or Kate? — Kate.

Do you like tea or coffee? — Tea.

Is this your book or mine? — It's yours.

Will you go to the theatre or to the cinema? — To the theatre.

Сложное дополнение (The Complex Object)

В английском языке существуют так называемые синтаксические комплексы, т. е. сочетания двух или нескольких слов, представляющие собой единое синтак­сическое целое (один член предложения).

Наиболее распространенным синтаксическим комплексом является сложное дополнение (**the Complex Object**).

Каждый синтаксический комплекс состоит из именной части (существитель­ного или местоимения) и глагольной части (инфинитива, герундия или причас­тия) и может быть развернут в целое придаточное предложение, где именная часть комплекса будет подлежащим, а глагольная — сказуемым.

Сложное дополнение может состоять из прямого дополнения и инфинитива:

I want you to visit me in the evening. Я хочу, ***чтобы* вы навестили** меня

ве­чером.

Pete wants me to write the letter. Петя хочет, ***чтобы* я написал** это письмо.

После глаголов see, hear, watch, make, feel, let в сложном дополнении инфинитив употребляется без частицы to:

I saw him come into the house. Я видел, ***что* он вошел** в дом.

Не heard me open the door. Он слышал, ***что* я открыла** дверь.

Сложное дополнение может также состоять из прямого дополнения и прича­стия:

I heard her singing an English song. Я слышал, ***как* она пела** английскую песню.

Не watched them playing in the garden. Он наблюдал за тем, ***как* они играли** в саду.

Разница между ними заключается в следующем.

Сложное дополнение с инфинитивом обозначает однократное действие:

I heard her say these words. Я слышал, ***что* она сказала** эти слова.

We saw her run into the house. Мы видели, что **она вбежала** в дом.

Сложное дополнение с причастием настоящего времени подчеркивает процесс протекания действия:

I saw her crossing the street. Я видел, ***как*** **она переходила** улицу.

I watched the children playing with a ball. Я наблюдал за тем, как **дети играли** в мяч.

Сложное дополнение не имеет точного соответствия в русском языке и пере­водится на русский язык дополнительным придаточным предложением с союзами что, чтобы (если в сложное дополнение входит инфинитив) и как (если в сложное дополнение входит причастие настоящего времени).

**354 страница отдельно**

Таблица времен английского глагола

Действительный залог

|  |  |  |  |
| --- | --- | --- | --- |
| Время  Форма | Present (настоящее) | Past  (прошедшее) | Future (будущее) |
| Indefinite (неопределенная) | I work. I write. | I worked.  I wrote. | I shall work.  I shall write. |
| Continuous (продолженная) | I am working.  I am writing. | I was working.  I was writing. | I shall be working.  I shall be writing. |
| Perfect  (совершенная) | I have worked.  I have written. | I had worked.  I had written. | I shall have worked.  I shall have written. |
| Perfect Continuous (совершенная продолженная) | I have been work­ing.  I have been writ­ing. | I had been working.  I had been writ­ing. | I shall have been working.  I shall have been writing. |

**Список неправильных глаголов**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Infinitive** | **Past Indefinite** | **Past Participle (Participle II**) | **Meaning** |
| 1 | abide | abode | abode | ждать, жить |
| 2 | arise | arose | arisen | возникать, появляться |
| 3 | awake | awoke\* | awaked | проснуться, будить |
| 4 | be [bi:] | was [woz],  were [wə:] | been [bi:n] | быть, находиться |
| 5 | bear | bore | born, borne | нести, терпеть |
| 6 | beat [bi:t] | beat [bi:t] | beaten ['bi:tn] | бить |
| 7 | become [bI 'k∧m] | became [bI 'keI m] | become [bI 'k∧m] | становиться |
| 8 | befall | befell | befallen | происходить, случаться |
| 9 | beget | begot | begotten | вызывать, порождать |
| 10 | begin [bI 'gI n] | began [bI 'gæn] | begun [bI'g∧n] | начинать |
| 11 | begirt | begirt\* | begirt\* | охватывать, окружать |
| 12 | behold | beheld | beheld | заметить, ассматри- вать. Вот! Смотри! |
| 13 | bend | bent\* | bent\* | сгибать(ся), поворачивать(ся) |
| 14 | bereave | bereft\* | bereft\* | лишать, отбирать |
| 15 | beseech | besought | besought | [просить](http://www.lingvo.ua/ru/Search/GlossaryItemExtraInfo?text=beseech&translation=%d0%bf%d1%80%d0%be%d1%81%d0%b8%d1%82%d1%8c&srcLang=en&destLang=ru), [умолять](http://www.lingvo.ua/ru/Search/GlossaryItemExtraInfo?text=beseech&translation=%d1%83%d0%bc%d0%be%d0%bb%d1%8f%d1%82%d1%8c&srcLang=en&destLang=ru) |
| 16 | beset | beset | beset | осаждать, обступать |
| 17 | bestride | bestrode | bestridden | [защищать](http://www.lingvo.ua/ru/Search/GlossaryItemExtraInfo?text=bestride&translation=%d0%b7%d0%b0%d1%89%d0%b8%d1%89%d0%b0%d1%82%d1%8c&srcLang=en&destLang=ru), [охранять](http://www.lingvo.ua/ru/Search/GlossaryItemExtraInfo?text=bestride&translation=%d0%be%d1%85%d1%80%d0%b0%d0%bd%d1%8f%d1%82%d1%8c&srcLang=en&destLang=ru), |
| 18 | betake | betook | betaken | [обращаться](http://www.lingvo.ua/ru/Search/GlossaryItemExtraInfo?text=betake&translation=%d0%be%d0%b1%d1%80%d0%b0%d1%89%d0%b0%d1%82%d1%8c%d1%81%d1%8f&srcLang=en&destLang=ru), [прибегать](http://www.lingvo.ua/ru/Search/GlossaryItemExtraInfo?text=betake&translation=%d0%bf%d1%80%d0%b8%d0%b1%d0%b5%d0%b3%d0%b0%d1%82%d1%8c&srcLang=en&destLang=ru) (к ему-л.) |
| 19 | bid | bid, bade | bid, bidden | [предлагать цену за](http://www.lingvo.ua/ru/Search/GlossaryItemExtraInfo?text=bid&translation=%d0%bf%d1%80%d0%b5%d0%b4%d0%bb%d0%b0%d0%b3%d0%b0%d1%82%d1%8c%20%d1%86%d0%b5%d0%bd%d1%83%20%d0%b7%d0%b0&srcLang=en&destLang=ru) |
| 20 | bind | bound | bound | [подтверждать сделку](http://www.lingvo.ua/ru/Search/GlossaryItemExtraInfo?text=bind&translation=%d0%bf%d0%be%d0%b4%d1%82%d0%b2%d0%b5%d1%80%d0%b6%d0%b4%d0%b0%d1%82%d1%8c%20%d1%81%d0%b4%d0%b5%d0%bb%d0%ba%d1%83&srcLang=en&destLang=ru&author=Administrator) |
| 21 | bite | bit | bitten | кусать(ся) |
| 22 | bleed | bled | bled | кровоточить, сочиться, расплываться |
| 23 | blow [blou] | blew [blu:] | blown [bloun] | дуть |
| 24 | break [breIk] | broke [brouk] | broken ['broukn] | ломать |
| 25 | breed | bred | bres | разводить, вызывать |
| 26 | bring [br I ŋ] | brought [bro:t] | brought [bro:t] | приносить |
| 27 | build [bIld] | built [bIlt] | built [bIlt] | строить |
| 28 | burn [bə:n] | burnt\* [bə:nt] | burnt\* [bə:nt] | гореть |
| 29 | burst | burst | burst | разразиться |
| 30 | buy [baI] | bought [bo:t] | bought [bo:t] | покупать |
| 31 | can [kæn], [transcription], [transcription] | could [kud] | — | мочь, уметь |
| 32 | cast | cast | cast | кидать, отбрасывать, отливать |
| 33 | catch [k**æ**t**∫**] | caught [ko:t] | caught [ko:t] | поймать, застать, успеть |
| 34 | chide | chid | chidden | ругать, бранить, шуметь |
| 35 | choose [t**∫**u:z] | chose [t∫ouz] | chosen ['t∫ouzn] | выбирать |
| 36 | cleave | cleft, clove | cleft, cloven | раскалываться, проникать |
| 37 | cling | clung | clung | цепляться, прилипать, держаться |
| 38 | clothe | clad\* | clad\* | облекать, обряжать |
| 39 | come | came | come | приходить, приезжать |
| 40 | cost | cost | cost | стоить, определять цену |
| 41 | creep | crept | crept | ползти, красться, овладевать |
| 42 | cut [k**∧**t] | cut [k**∧**t] | cut [k**∧**t] | (от-), (по-) резать, |
| 43 | dare | durst\* | dared | осмеливаться, сметь, рисковать |
| 44 | deal | dealt | dealt | раздавать, торговать. обходиться, иметь дело |
| 45 | dig [dIg] | dug\* [d**∧**g] | dug\* [d**∧**g] | копать, рыть, извлекать |
| 46 | dip | dipt\* | dipt\* | макать, нырять, досавать |
| 47 | do [du:] | did [d Id] | done [d**∧**n] | делать, создавать, выполнять |
| 48 | draw [dro:] | drew [dru:] | drawn [dro:n] | рисовать, чертить, копировать, представ- лять, излбражать |
| 49 | dream [dri:m] | dreamt\* [dremt] | dreamet\* [dri:md] | мечтать, помышлять, видеть сон |
| 50 | drink [dr I ŋ∫k] | drank [ dræŋk] | drunk(en) [ dr∧ŋk] | пить, пьянствовать, испить |
| 51 | drive [ draIv] | drove [ drouv ] | driven [ dr Ivn] | ехать, вести, везти, приводить в действие |
| 52 | dwell | dwelt\* | dwelt\* | жить, находиться, задерживаться |
| 53 | eat [ i:t] | ate, eat [ et ] | eaten [ i:tn] | есть,кушать, съедать |
| 54 | engrave | engraved | engraven\* | гравировать, вырезать, запечатлевать |
| 55 | fall [fo:l] | fell [ fel] | fallen [ fo:ln] | падать, снижаться, спускаться |
| 56 | feed [fi:d] | fed[ fed ] | fed[ fed] | питать (-ся), кормить, обеспечивать, подавать |
| 57 | feel [fi:l] | felt[ felt ] | felt[ felt] | чувствовать, (пр-) ощу- пы вать, считать, полагать |
| 58 | fight[faI] | fought[ fo:t] | fought[ fo:t] | драться, бороться, защищать, командовать |
| 59 | find[faI nd] | found[ fnd] | found[ fnd ] | находить, встречать, обнаруживать |
| 60 | flee | fled | fled | убегать, исчезать, избегать |
| 61 | fling | flung | flung | бросать (-ся), кидать (-ся), ринуться |
| 62 | fly [ flaI] | flew[ flu: ] | flown[ fl n] | летать, пролетать, разиеваться |
| 63 | forbear | forbore | forborne | сдерживаться, воздерживаться |
| 64 | forbid | forbade,forbid | forbidden | запрещать, не позволять, препятствовать |
| 65 | forecast | forecast | forecast | предвидеть, предсказы- вать,прогнозировать |
| 66 | forego | forewent | foregone | предшествовать, идти впереди |
| 67 | foresee | foresaw | foreseen | предвидеть, предвос- хищать, предсказываь |
| 68 | fortell | foretold | foretold | предсказывать, прогнозировать |
| 69 | forget[ fə get] | forgot[ fot] | forgotten[ fotn] | забывать, позабыть |
| 70 | forgive | forgave | forgiven | прощать, забывать |
| 71 | forsake | forsook | forsaken | оставлять, покидать, отказываться |
| 72 | foresware | forswore | forsworn |  |
| 73 | freeze | froze | frozen | замерзать, замирать, застывать |
| 74 | get [ get] | got[ ot] | got, gotten[ ot] | получить, приобрести, достать, добираться |
| 75 | gild | gilt\* | gilt\* | (по-) золотить, украшать |
| 76 | gird | girt\* | girt\* | подпоясываться, окружать, блокировать |
| 77 | give[gIv] | gave[ eIv] | given[ Ivn] | (от-) дать, передать |
| 78 | go[ ou] | went[ went] | gone[ on] | идти, ехать, двигаться, уходить, уезжать |
| 79 | grave | graved | graven\* | вырезать, гравировать, произвести впечатление |
| 80 | grind | ground | ground | [молоть](http://lingvopro.abbyyonline.com/ru/Search/GlossaryItemExtraInfo?text=grind&translation=%d0%bc%d0%be%d0%bb%d0%be%d1%82%d1%8c&srcLang=en&destLang=ru), [перемалывать](http://lingvopro.abbyyonline.com/ru/Search/GlossaryItemExtraInfo?text=grind&translation=%d0%bf%d0%b5%d1%80%d0%b5%d0%bc%d0%b0%d0%bb%d1%8b%d0%b2%d0%b0%d1%82%d1%8c&srcLang=en&destLang=ru); [растирать](http://lingvopro.abbyyonline.com/ru/Search/GlossaryItemExtraInfo?text=grind&translation=%d1%80%d0%b0%d1%81%d1%82%d0%b8%d1%80%d0%b0%d1%82%d1%8c&srcLang=en&destLang=ru) (в порошок); [толочь](http://lingvopro.abbyyonline.com/ru/Search/GlossaryItemExtraInfo?text=grind&translation=%d1%82%d0%be%d0%bb%d0%be%d1%87%d1%8c&srcLang=en&destLang=ru); [размельчать](http://lingvopro.abbyyonline.com/ru/Search/GlossaryItemExtraInfo?text=grind&translation=%d1%80%d0%b0%d0%b7%d0%bc%d0%b5%d0%bb%d1%8c%d1%87%d0%b0%d1%82%d1%8c&srcLang=en&destLang=ru) |
| 81 | grow[grou] | grew[ ru:] | grown[ r n] | расти, делаться, становиться |
| 82 | hang[ hæŋ] | hung[ h∧ŋ] \* | hung[ h∧ŋ] \* | вешать, висеть, зависать |
| 83 | have[ hæv] | had[hæd] | had[ hæd] | иметь, обладать, включать, содержать |
| 84 | hear[ hI ] | heard[ hə:d] | heard[ hə:d] | (у-, за-) слышать, |
| 85 | heave | hove\* | hove\* | поднимать, перемещать, снимать |
| 86 | hew | hewed | hewn\* | разрубать, рубить, вытёсывать |
| 87 | hide[ haId] | hid[ hId] | hid, hidden[ hIdn] | прятать, скрывать |
| 88 | hit | hit | hit | ударить ( –ся), попадать, достигать |
| 89 | hold[hld] | held[ held] | held[ held] | хранить, вмещать (со-)держать, |
| 90 | hurt | hurt | hurt |  |
| 91 | inlay | inlaid | Inlaid |  |
| 92 | interweave | interwove | Interwoven |  |
| 93 | keep[ ki:p] | kept[ kept] | kept[ kept] |  |
| 94 | kneel | knelt\* | knelt\* |  |
| 95 | knit | knit\* | knit\* |  |
| 96 | know[ nou] | knew[ nu:] | known[ nn] | знать |
| 97 | lade | laded | laden\* |  |
| 98 | lay | laid | laid |  |
| 99 | lead | led | led | вести |
| 100 | lean | leant\* | leant\* |  |
| 101 | leap | leapt\* | leapt\* | прыгать |
| 102 | learn[ lə:n] | learnt\* [ lə:nt] | learnt\* [ lə: nt] | учить (-ся) |
| 103 | leave[li:v] | left[ left] | left[ left] | покидать, уходить |
| 104 | lend | lent | lent |  |
| 105 | let[ let] | let[ let] | let[ let] | позволять |
| 106 | lie[laI] | lay[ leI ] | lain[ leIn] | лежать |
| 107 | light[laI t] | lit[ lI t]\* | lit\* , lighted[ laItId] | светить |
| 108 | lose[ lu:z] | lost[ lost] | lost[ lost] | терять |
| 109 | make[ meIk] | made[ meId] | made[ meId] | делать |
| 110 | may[ meI] | might[ maIt] | – | (воз-) можно |
| 111 | mean[ mi:n] | ment[ ment] | meant[ ment] | значить |
| 112 | meet[ mi:t] | met[ met] | met[ met] | встречать |
| 113 | melt | melted | molten\* | плавить |
| 114 | methinks | methought | – |  |
| 115 | misgive | misgave | misgiven |  |
| 116 | mislay | mislaid | mislaid |  |
| 117 | mislead | misled | misled |  |
| 118 | misshape | misshaped | misshapen\* |  |
| 119 | misspeak | misspoke | misspoken |  |
| 120 | misspell | misspelt\* | misspelt\* | ощибаться |
| 121 | mistake | mistook | mistaken |  |
| 122 | misunderstand | misunderstood | misunderstood | непонимать |
| 123 | miswrite | miswrote | miswritten |  |
| 124 | mow | mowed | mowen\* |  |
| 125 | must |  |  | должен |
| 126 |  | ought |  | следует |
| 127 | outbid | outbid | outbid, outbidden |  |
| 128 | outdo | outdid | outdone |  |
| 129 | outgo | outwent | outgone |  |
| 130 | outgrow | outgrew | outgrown |  |
| 131 | outride | outrode | outridden |  |
| 132 | outrun | outran | outrun |  |
| 133 | outshine | outshone | outshone |  |
| 134 | overbear | overbore | overbone |  |
| 135 | overcast | overcast | overcast |  |
| 136 | overcome | overcame | overcome | преодолевать |
| 137 | overdo | overdid | overdone |  |
| 138 | overdrive | overdrove | overdriven |  |
| 139 | overeat | overate | overeaten |  |
| 140 | overhang | overhung | ovberhung |  |
| 141 | overhear | overheard | overheard |  |
| 142 | overlade | overladed | overladen\* |  |
| 143 | overlay | overlaid | overlaid |  |
|  | override | overrode | overridden |  |
|  | overrun | overran | overrun |  |
|  | oversee | oversaw | overseen |  |
|  | overset | overset | overset |  |
|  | overshoot | overshot | overshot |  |
|  | oversleep | overslept | overslept |  |
|  | overspread | overspread | overspread |  |
|  | overtake | ocertook | overtaken |  |
|  | overthrough | overthrew | overthrown |  |
|  | partake | partook | partaken |  |
|  | pay[peI] | paid[ peId] | paid[ peId] | платить |
|  | pen | pent\* | pent\* | включать |
|  | put[ put] | put[ put] | put[ put] | класть |
|  | read | read[ red ] | read[ red ] | читать |
|  | rebuild | rebuilt\* | rebuilt\* |  |
|  | rend | rent | rent |  |
|  | repay | repaid | repaid |  |
|  | rid | rid | rid |  |
|  | ride | rode | ridden |  |
|  | ring | rang[ ræŋ], rung[ rŋ] | rung[ rŋ] |  |
|  | rise | rose | risen |  |
|  | rive | rived | riven |  |
|  | run | ran[ ræn] | run [] | бежать |
|  | saw | sawed | sawn\* |  |
|  | say | said[ sed] | said[ sed] | сказать |
|  | see | saw[ so:] | seen [] | видеть |
|  | seek | sought | sought |  |
|  | seethe | seethed | sodden\* |  |
|  | sell | sold[ sld] | sold[ sld] | продавать |
|  | send | sent[ sent] | sent[ sent] | посылать |
|  | set | set[ set] | set[ set] |  |
|  | shine | shone[ n] | shone[ n] | светить |
|  | show | showed[∫d] | showen[∫n] | показывать |
|  | shut [ ∫∧t] | shut[ ∫∧t] | shut[ ∫∧t] | захлопнуть |
|  | sing | sang[ sæŋ] | sung[ sŋ] | петь |
|  | sink | sank[ sæŋk] | sank[ sæŋk] | тонуть |
| 144 | sit [s**I**t] | sat [s**æ**t] | sat [s**æ**t] | сидеть |
| 145 | sleep [sli:p] | slept [slept] | slept [slept] | спать |
| 146 | smell [smel] | smelt [smelt] | smelt [smelt] | пахнуть |
| 147 | speak [spi:k] | spoke [spouk] | spoken ['spoukn] | говорить |
| 148 | spend [spend] | spent [spent] | spent **[spent]** | тратить |
| 149 | stand [st**æ**nd] | stood [stud] | stood [stud] | стоять |
| 150 | strike [stra**I**k] | struck **[str∧k]** | struck **[str∧k]** |  |
| 151 | swim [sw**I** m] | swam [sw**æ**m] | swum **[sw∧m]** | плавать |
| 152 | take [te**I**k] | took [tuk] | taken ['te**I**kn] | брать |

Additional materials

**Prepositions**

**by** (~ a series - ; ~ 2 o’clock; works ~ M. Auezov); **by** means of **for** (**For** it is so …; this is **for** you; ) **to** (I go **to** the university; to come **to** the table; read the text **from** the beginning **to** the end;) **on** (put **on** your coat; put the book **on** the table; to come **on** time; **on** Monday; **on** the 7th of May; **on** week-days) **from** (I came home **from** the University. Take the book **from** the table.) **till** (wait for me **till** 12 o;clock;) **until** (wait **for** me **until** I come back;) **in** (**in** the house; I shall be back **in** an hour; **in** the pocket; to come **in** time; **in** March; **in** 2011) **into** (come **into** the room; put the pen **into** your bag;) **along** (**along** the street; **along with** the rest of the articles; ) **at** 7 o’clock; **at** the bus stop; **at** work; **at** the exams; **at** the table; Are you **at** home now?) **by** (I shall come **by** 2 o’clock; he is Marat **by** the name; she was killed **by** him;) **before** (**before** the lesson, the table; **before** I come; **before** you begin it, you must …;) **beyond** (it is **beyond** the theme; ) **after** (I came to the finish **after** him; **after** the lesson; he came **after** me) **out of** (**out of** the building;) **under** (**under** the table; the problem **under** consideration (study, investigation)**;**) **over** (the lamp is **over** the table; milk spilt **over** the pan; **over** years; **with** (**with** the help of; to be killed **with** a heavy stone; to write **with** a pen; **onto** (the cat fell just **onto** the table;) **near** (**near** the window; there is a garden **near** the house)

**Noun chains**

control operations computer system a storage device input devices output devices high-speed store backing store construction kit access store car speedometer process control gas pressure processing programs adding machines cash register storage unit a reference number any storage location a data item the human operator critical path method analysis high level languages the systems analyst information output information explosion research workers flight plans airline pilots doctors case histories of patients the other computer professionals information-producing equipment complex production programs accounting and calculating machines

**Equivalents of the modal verbs**

to **be** to = **must;** **were** to go; **is** to read; **am** to do; **~~will~~****~~be~~** ~~to do it~~; to **have** to; **has** to face; will **have** to take the exam; to **have got** = **must**; **has** **got** to go to Kaskelen; to be **able** to = **can (could**); **am** **able** to read; **was able** to sing; **will be able** to speak; to be **allowed** to, to be **permitted** to **= may (might);** **am allowed** to go home; **was permitted** to rewrite; **shall be allowed** to see the film

**Conversion**

**to** form – **a** form **to** change – **a** change **to** speed – **a** speed **to** need – **a** need **to** excess – **an** excess **to** sense – **a** sense **to** process – **a** process **to** display – **a** display **to** control – **a** control **to** slow – slow **to** measure – **a** measure **to** address – **an** address **to** face – **a** face **to** issue – **an** issue **to** correct – correct **to** cause – **a** cause **to** design – **a** design **to** load – **a** load **to** plan – **a** plan **to** base – **a** base **to** try – **a** try **to** attempt – **an** attempt **to** measure – **a** measure

**Synonyms**

store – memory – storage different – various – distinct digit – figure – numeral distinct – separate – individual purpose – target – goal – aim continuously – constantly to cause – to make fast – quick – rapid to obtain – to get – to receive to develop – to work out – to devise at once – immediately – instantly insignificant – negligible – trifling none – nothing, nobody to design – to plan, to devise to decide – to solve to obtain – to get – to receive to have – to possess – to owe day-to-day – every day, daily vast – enormous – very great to perform – to do – to make – to fulfill sequence – set – series large – big to give – to yield – to produce – to provide to use – to exploit – to apply – to employ up-to-date – modern – present – contemporary the finished product – the output – the end product ready – easy files or pockets every – each – everybody – everything

**Antonyms**

**big** – small **large** – little **thin** – thick **long** – short **difficult** – easy, ready **bad** – good **high** – low **far** – near **dark** – light **heavy** – light **tall** – short **bright** – dim **white** – black **near** – far

to **switch** off – to **switch** on **important** – insignificant – negligible – trifling

**The words of the same origin (root) (derivatives)**

**difficult –**difficulty **to differ –** different – difference – differentiate – differentiation **to vary** – varience – variant – variation – variable – variation – varied – variety – various – variability **mean** – meaning – meaningful – meaningless **to** **form** – **a** **form** – formal – formation – formative – formless – formally – formulation – formartiveness –format –formatless –formatter –formatted –formatting –former

**Phrases**

a large number **for** example **as** long **as**  **at** any time **by** watching analog computers hybrid computers **in** terms of rather than **without** limit central processor **in** general and so **on** **in** the same way to make sure that an exacting one taken care **of** **in** particular a number of ready-made put **to** use a great range of taxation **for** payroll **at** once such as **for** instructions to be written **out** that is **say** $100 326 475, … to use … they are designed to none of these to establish the problem to be solved to be designed to **in** fact more and more acute day **by** day to start things **off** this way – **in** this way to **make** correct **decisions** **over** lengths of time **both** digital **and** analog computers a large filing cabinet **without** any discrimination **in** no circumstances the computer itself which are required **either** commercial **or** scientific not only associated operations

**Confusion words**

che**ck** – che**que** f**u**ture – f**ea**ture father – (far) f**a**rther – f**u**rther la**t**er – la**tt**er form – form**er** **the** former – **the** latter th**e**n – th**a**n th**e**se – th**o**se f**or** – f**ro**m to ad**a**pt – to ad**o**pt diff**erent** – diff**icult pre**position – **pro**position some time – sometime – sometime**s** read [i:] – read [e] – read**y** [e]**;**

**Programming word combinations**

internal circuitry items of data numeric code computer languages exacting rules computer code high-level languages hardware a raw material this powerful tool falls on the systems analyst this “raw data” is fed to policy decisions a computer is a device 'pocket' or storage location problem orientated languages to turn a raw material into information dispersed to peripheral units

**Prefixes**

**un**processed **pre**-determined **non**-stop mode **un**known **for**ward **back**ward **im**plicit; **in**direct **pre**position **post**position **under**estimated **over**estimated **in**dependent **ir**regular **il**legal; **sub**conscious **super**natural **anti**fascist **ultra-**vialet **post**war **inter**national **mis**believe **uni**lateral **multi**lateral **mono**linqual **multi**lingual **bi**lingual **sub**way **mono**syllabic **poli**syllabic **co**worker **re-**count **auto**biography **vice-**president

**Suffixes**

address**ée** book**let**  chemist**ry** child**hood** Chin**ese** dramat**ist** friend**ship** free**dom** happi**ness** hero**ism**  machine**ry** organiza**tion** Russ**ian** success**ive** illogical**ity** proper**ty** end**less** speci**al** specif**ic** process**or** destina**tion** Dan**ish** develop**ment** act – act**or** – actr**ess** process – process**ing** – process**es** - process**ed;** programme – programm**er –** programm**ing –**programm**es –** program**s** – programm**ed**

**Possessive case of nouns**

computer**’s** operations; programmer**’s** handbook; the student**’s** textbook; **~~the~~** **Namazbayev’s** book; the door **of** the **room**; windows **of** the **flat**;

**Nouns borrowed from the Latin or Greek languages**

ax**is** – ax**es** antenn**a** – antenn**ae** data **–** dat**um** analys**is –** analy**ses** cris**i**s – cris**e**s criteri**on** - criteri**a** geni**us** – geni**i** formul**a** – formul**ae** radi**us –** radi**i** errat**um** – errat**a** ellips**is** - ellips**es** diagnos**i**s – diagnos**e**s thes**is** – thes**es** stimul**us** – stimul**i** phenomen**on –** phenomen**a**

**Conjunctions**

**if** – если, ли; **until** – до тех пор, пока не; **provided** (**that**) – при условии, что; если; **though, although** – хотя (бы), несмотря на; **after** – после того как; **before** – прежде чем; до того как; перед тем как; раньше чем; **for –** так как; **as** – поскольку; так как; по мере того как; в то время как; как **unless** – если не; **lest** – для того чтобы не; **and** – и, а **but** – но, а **or** – или, тначе **either … or** – или … или; либо … либо **neither … nor** – ни … ни  **both … and** – и … и; как … так и  **whereas** – тогда как; а  **while** – тогда как; а; в то время как; пока  **when** – когда  **while** –  **as soon as** – как только **as/so long as** – пока; до тех пор пока  **since** – так как; поскольку  **because** – потому что; так как  **so (that)** – поэтому **in order to/that** – с тем чтобы **if even** – даже если;  **if** – если  **only if** – если бы **in case (that)** – в случае; на случай (если) **than** – чем

**Infinitive, Participles as an attribute**

operations to be performed – операции, которые будут выполнены (которые нужно выполнить) programs to be worked out – программы, которые должны быть разработаны (которые нужно разработать) the problem to be solved – задача, которая будет решена (которая должна быть решена) the raw material to be required to obtain this – необработанный материал, который требуется, чтобы получить это (который нужен, чтобы получить это) a program to be stored in the computer to perform the tasks – программа, которая должна храниться в компьютере, чтобы выполнить эти задания (которая будет храниться в компьютере, для выполнения этих заданий)

**Substitute words**

**a** very significant **one**; this solution differs from **that of** **yours;** I **know** the language but my father **did** not. These data are very **important** for theory, and less **so** for practice. We have two srudents, **Khabiyev** and **Nazarov**. **The** **former** is a brilliant **boy** **the latter** is a lazy **one.** I **read** such books, but he **won’t**. **This** **point** of view is **that of** a mathematician rather than a physicist. The most extensive **investigation** was **that of Lehman(1).** They obtained **results** similar to **ours.**

**Difficult sentences (constructions, expressions)**

**1. The nearer** the language used by the programmer to the code used by the machine, **the lower** the 'level' of the language. 2. He must be skilled not only **in** **analysing** a problem (in **an** analisa**tion** of a problem) and **devising** a solution (in **the** design**ing** of a solution), but **in** **communicating** his solution to the client or management. 3. He must be skilled not only in analysing a problem and devising a solution, but in communicating his solution to the client or management specify**ing** the requirements, to the users of the information, to **those** involved in collect**ing** raw data and distribut**ing** information and to the other computer professionals involved in the development of the computer side of the system.

4. associated groups 5. detailed descriptions of specifications of individual jobs 6. the overall system 7. Each such job will require a program to be stored in the computer to perform the tasks. 8. keeping this explosion under control 9. Computers, with their ability to process great quantities of facts at vast speeds, are **the** **only** means… 10. order **out of** apparent chaos 11. Computers can give the management the information they need to make correct decisions.

**Part IV**

**Contracted forms**

*a* – adjective – прилагательное – сын есім

*adv* – adverb – наречие – үстеу

*prep –* preposition *–* предлог – емеулік

*int –* interjection *–* междометие – одағай

*n –* noun – существительное – зат есім

*pron –* pronoun – местоимение – есімдік

*sg –* singular – единственное число – жекеше

*pl –* plural – множественное число – көпше

*v –* verb – глагол – етістік

*conj –* conjunction – союз – жалғаулық

*num –* numeral – числительное – сан есім

**vocabulary**

*a v adv n prep v n a pron sg pl v conj num*

***A***

|  |  |  |  |
| --- | --- | --- | --- |
| about *prep* | [ә'baut] | *о, об, про* | *туралы, жөнінде* |
| accept *v* | [ək'sept] | *принимать* | *қабылдау, ұсынысқа келісу* |
| access *n* | ['ækses] | *доступ* | *жол; қатынау;өтетін жер* |
| according (to) *prep* | [ə'kɔ**:**dɪŋ] | *согласно (ч.(к.)-л.)* | *(біреуге , бір нәрсеге) сәйкес, байланысты* |
| account  *n, v* | [ә'kaunt **I** ŋ] | *отчёт, делать ~* | *есеп, түсінік; есеп беру* |
| accurately *adv* | ['ækjərətlɪ] | *точно* | *дәл, дәлме-дәл* |
| achieve *v* | [ә't∫ i**:**v] | *достигать* | *мақсатқа жету, табысқа жету* |
| acute *a* | [ə'kju:t] | *острый (угол)* | *сүйір(бұрыш)* |
| add *v* | [æd] | *складывать* | *қосу, сала түсу* |
| address *v* | [ә'dres] | *адрес, обращаться к* | *мекенжай; іспен бару; керілеу,қатынасу* |
| add up  *v* | ['æd ʌp] | *прибавлять* | *қосу* |
| advantage *n* | [əd'vɑ**:**ntɪʤ] | *преимущество* | *артықшылық, басымдылық* |
| affect *v* | ['æfekt] | *влиять (на)* | *әсер ету; ықпалын тигізу*  *(-ға,-ге,-қа,-ке,-а,-е)* |
| again *adv* | [ә'gen] | *снова, опять* | *тағы, және, екінші рет* |
| agree *v* | [ә'gri:] | *соглашаться* | *келісу* |
| airline *n* | ['εәla**I**n] | *авилиния* | *әуе жолы* |
| alone *a, n* | [ә'loun] | *отдельный, один* | *жалғыз; жеке* |
| all (the other) *a, pron* | [ɔ**:**l(ði**:**'ʌðə)] | *все, всё* | *барлығы, бәрі* |
| already *adv* | ['о**:**lred **I**] | *уже* | *әлдеқашан, осыдан бұрын* |
| also *adv* | ['о**:**lsou] | *тоже, также, ещё* | *сонымен қатар, сондайақ, тағы* |
| although *conj,adv* | ['о**:**lðou] | *хотя, не смотря на* | *бір жағынан; қарамай* |
| always *adv* | ['ɔ**:**lweɪz] | *всегда* | *әрдайым, әрқашан* |
| amount *n* | [ә'maunt] | *количество* | *саны,мөлшер, өлшем; барлығы* |
| analyst  *n* | ['ænәl**I**st] | *аналитик* | *аналитик* |
| and *conj* | [ænd] | *и* | *және* |
| another  *a, pron* | [ə'nʌðə] | *ещё один, другой* | *тағы бірі; басқа* |
| any *a, pron* | ['enɪ] | *любой,*  *что-либо* | *әрбір, кез келген* |
| anything *pron* | ['en **I** θ **I** ŋ] | *что-либо* | *бір нәрсе, бірдеме* |
| application *n* | [æplɪ'keɪʃən] | *приложение* | *қосымша, үстеме;қолдану;өтініш,сұрау* |
| appropriate *a* | [ə'prəuprɪət] | *подходящий* | *қолайлы* |
| apparent *a* | [ə'pærənt] | *очевидный* | *анық, айқын; анығы* |
| architect *n* | ['α**:**k **I** tekt] | *архитектор* | *сәулетші* |
| art *n* | ['α**:**t] | *искусство* | *көркемөнер,шеберлік* |
| as *conj* | [æz] | *поскольку, так как* | *сондай, қалай, ретінде; себебі,өйткені* |
| ask *v* | ['α**:**sk] | *спрашивать* | *сұрау, сұрап білу* |
| associate *v* | [æ'souʃɪeɪt] | *связывать* | *байланыстыру,қарым-қатынас* |
| astonishingly *adv* | [ə'stɔnɪʃɪŋli ] | *удивительно, изумительно* | *таңырқарлық, тамамша* |
| attempt  *n* | [ә'tempt] | *попытка* | *әрекет,әрекет жасау; талпыныс* |
| attention *n* | [ә'ten∫әn] | *внимание* | *назар, назар аудару* |
| available *a* | [ә'veilәbl] | *имеющийся*  *(в наличии)* | *бары, бар болуы;ашық* |

***В***

|  |  |  |  |
| --- | --- | --- | --- |
| base *n* | [be **I** s] | *база (данных)* | *база; деректер базасы; тірек ету* |
| basic *a* | [ be **I** s **I** k] | *основной* | *негізгісі; базистік* |
| basically *adv* | [ ่be **I** s **I** kәl **I**] | *в основном* | *негізгі* |
| because (of) *conj* | [b**I** ่kɔ**:**z (әv)] | *так как, потому что; (из-за)* | *өйткені; себебі; сондықтан;осыған бола* |
| become *v* | [b**I** k∧m] | *становиться* | *болу, жарау; жарасу* |
| between *prep* | [b**I** ่twi**:**n] | *между (двумя)* | *арасында, өзара* |
| big (**g**er) *a* | [่ b**I**g(ә)] | *большой, (больше)* | *үлкен, биік (үлкенірек)* |
| bill *n* | [b**I**l] | *расчет, вычисление; сумма* | *есеп,есептеу,есептелім;қосынды* |
| blunt (pencil) *a* | [blʌnt] | *тупой* | *доғалы; тік* |
| booking *n* | [่buk**I** ŋ] | *бронирование; заказ* | *тапсырыс беру, алу (билет)* |
| both *pron* | [bouθ] | *оба, обои, обе* | *екеуі де* |
| box *n* | [bɔks] | *блок; поле; окно; рамка* | *блок,біртекті заттар; өріс; терезе; жақтау;шеңбер* |
| buy *v* | [ba **I**] | *покупать* | *сатып алу* |
| byte *n* | [ba**I** t] | *байт (единица информации)* | *байт (информациалық бірлік)* |

***С***

|  |  |  |  |
| --- | --- | --- | --- |
| cabinet *n* | ['kæb**I** n**I** t] | *корпус; систем. блок; шкаф* | *тұрқы, сырты; жүйелік блок;шкаф* |
| calculation *n* | [ **͵**kælkju**:** ่le**I**∫әn] | *вычисление; расчёт* | *есептеу,есептелім, есептеп шығару;есеп* |
| can *v* | [kæn] | *мочь; уметь; может быть; возможно* | *қабілетті болу; мүмкін болу* |
| car *n* | [kα**:**] | *автомашина* | *автокөлік* |
| carefully *adv* | [ ่kεәful**I**] | *внимательно; осторожно* | *ұқыпты; ықыласпен; абайлап* |
| carry out *v* | ['kær **I** aut] | *выполнять* | *жүзеге асыру, орындау* |
| cash (register) *n* | [kæ∫ ( ่redʒ**I**stә)] | *наличные деньги* | *қолдағы ақша; бар болушылық* |
| catalogue *n* | [ kætәlɔg] | *каталог; каталогизировать* | *тізімдеме; белгілі тәртіппен, тізім жасау* |
| cause *n v* | [kɔ**:**z] | *причина; вызывать* | *себеп, дәлел, себеп болу* |
| causing *a* | [ ่kɔ**:**z **I** ŋ] | *вызывающий* | *себеп болып жатушы; шақырып алушы* |
| certain *a* | [ ่sә**:**t(ә)n] | *определённый; конкретный* | *белгілі, әйгілі; нақты* |
| change *v* | [ ่t∫e**I** ndʒ] | *(из)менять* | *өзгерту, ауыстыру; айырбастау* |
| chaos *n a* | [่ke**I** әs] | *хаос; хаотический* | *былық, астан-кестен, апыр-топыр* |
| character *n* | [่kærәktә] | *символ; знак; буква; признак* | *бейне,символ; таңба, әріп; белгі* |
| cheap(er) *a* | [ ่t∫i:p (ә)] | *дешёвый(ле)* | *арзан, (арзанырақ)* |
| check *n* | [t∫ek] | *остановка; проверка; контроль* | *тоқтату ; тексеру; бақылау; қадағалау* |
| cheque *n* | [t∫ek] | *(магазинный) чек* | *чек (дүкен чегі)* |
| circuit *n* | [่sә**:**k**I** t] | *итерация; линия связи; схема* | *шыр айналу; байланыс тізбегі;схема* |
| circuitry *n* | [่sә**:**k**I** tr **I**] | *схемы; цепи; схематика* | *схемалар, кесте,сұлба;тізбелер, тізбектер,тізбе* |
| circumstance *n* | [่sә**:**kәmstәns] | *обстоятельство* | *жағдай* |
| clear-cut | [่kl**I**ә ่kʌt] | *чётко (хорошо) сделанный* | *анық, дәл, түзу (орындалған)* |
| clerical *n* | [ ่kler **I**kәl] | *канцелярская (конторская) работа* | *кеңсе жұмысы* |
| client *n* | [ ่kla**I**әnt] | *клиент; пользователь* | *тапсырыс беруші; пайдаланушы* |
| code *n* | [koud] | *код; программа* | *код; программа* |
| collected *a* | [kә ่lekt **I** d] | *собранный; накопленный* | *жиналынған, топталған* |
| collectively *adv* | [kә ่lekt **I** vl**I**] | *в общем; в целом; вместе* | *бірге, біргесіп* |
| combining *v, n* | [kәm ่ba**I** n**I** ŋ] | *комбинировать; объединять; комбинирование* | *бірігу, бірлесу; қосу, құрамдастыру; қосылыс; бірлестік* |
| communicate *v* | [kә ่mju**:**n**I**ke**I** t] | *связываться, взаимодействовать; передавать данные* | *байланыстыру; жалғану; бірлесіп әрекет ету; деректерді жіберу* |
| complete *a, v* | [kәm ่pli**:**t] | *полный; завершать* | *түгел; аяқталу, бітіру* |
| completed *a* | [kәm ่pli**:**t **I**d] | *полный; завершённый* | *түгел, аяқталған; біткен* |
| complex *a* | [่kɔmpleks] | *сложный (составной)* | *күрделі (құранды)* |
| comprehensible *a* | [ **͵**kɔmpr**I** ่****hensәbl] | *понятный; полный* | *түсінікті; толық,түгел* |
| concept *n* | [ **่**kɔnsept] | *понятие; концепция; принцип* | *түсінік; негізгі пікір; түпкі түйін* |
| concerned  with | [kәn ่sә**:**nd w **I** ð] | *имеющий отношение к ч.-л.* | *бір нәрсеге қатысы бар* |
| consequence *n* | [ ่kɔns **I**kwәns] | *следствие* | *нәтиже; нәтижесінде* |
| consider *v* | [kәn ่s**I**dә] | *рассматривать; считать; полагать* | *ойлау; есептеу, санасу; жорамалдау; шамалау* |
| consist (of) *v* | [kәn่s**I**st (әv)] | *состоять (из)* | *құрылу, тұру(-дан, -ден,*  *-тан, -тен, -нан, -нен)* |
| constantly *adv* | [ ่kɔnst*ә*ntl **I**] | *постоянно; всегда; часто* | *жиі; үнемі; әрқашан* |
| construction *n* | [ kәns ่trʌk∫(ә)n] | *конструкция; структура* | *құрылыс; құрлысы, құрамы;құрылым* |
| consult *v* | [kәn ่sʌlt] | *консультироваться* | *кеңесу, ақылдасу; анықтау* |
| contain *v* | [kәn ่te**I** n] | *содержать; делиться без остатка* | *болу; ұстау; тура бөліну* |
| containing *a* | [kәn ่te**I** n**I** ŋ] | *содержащий* | *ұстаушы, болушы* |
| content *n* | [ ่kɔntent] | *содержание; информация; суть; основное содержание; объём; ёмкость* | *мән, мәніс; негізгі мазмұн мағына, маңызы, мәні; көлем, мөлшер; мәлімет, ақпарат* |
| continually *adv* | [kәn ่ti**:**njuәl**I**] | *непрерывно; постоянно* | *қайта-қайта, үнемі, әрдайым* |
| continuously *adv* | [kәn ่ti**:** njuәsl**I**] | *непрерывно; постоянно* | *үзіліссіз, тұрақты; үнемі, қашан да, әрқашан* |
| control *v* | [kәn ่troul] | *управлять; регулировать* | *басқару; реттеу; тәртіптеу* |
| convert *v* | [kәn ่vә**:**t] | *преобразовывать; обращать* | *түрлендіру; айналдыру, өзгерту* |
| converting it into | [kәn่vә**:**t**I** ŋ **I**t **I** ntә] | *преобразование его (её, их) в (во)* | *оны бір нәрсеге түрлендіру (өзгерту)* |
| conversion *n* | [kәn ่vә**:**∫(ә)n] | *преобразование; обращение* | *айналу, айландыру; қатынасу; түрлендіру;*  *өзгеру, өзгеріс* |
| correct *a, v* | [kә ่rekt] | *правильный; исправлять* | *дұрыс, шын; түзету, жөндеу; дұрыстау* |
| (of) course *n* | [(әv) kɔ**:**s] | *курс; ход; (конечно)* | *бағыт; тәртіп; жол, тәсіл; түр (рас)* |
| cover *v* | [ ่kʌvә] | *покрывать; охватывать* | *жайылып кету; жабу; қамту,қапсыру* |
| crucial *a* | [ **่**kru**:** **∫**jәl] | *решающий; критический* | *шешуші; қиын, сыни* |

***D***

|  |  |  |  |
| --- | --- | --- | --- |
| daily *a* | [ ่de **I**l**I**] | *ежедневный; суточный* | *күн сайынғы; күнделікті* |
| data *n* | [ ่de**I** tә] | *данные; информация* | *деректер; берілгендер; мәліметтер; жаңалық* |
| decide *v* | [d**I** ่sa**I**d] | *решать; решиться* | *ұйғару; айқын; белгілі шешім қабылдау* |
| decline *v* | [d**I** ่kla**I** n] | *отклоняться* | *төмендеу; қабылдамау, азаю* |
| demand *v* | [d**I** ่mα**:**nd] | *требовать; запрашивать* | *талап ету; сұрату* |
| depend (on) *v* | [d**I** ่pend (ɔn)] | *зависеть (от)* | *тәуелді болу ( -нан,-нен,-дан,-тан,-тен)* |
| describing *n* | [d**I** ่skra**I** b**I** ŋ] | *описывание* | *суреттеу, сипаттау* |
| design *v* | [d**I** ่za**I** n] | *конструировать; делать проект* | *құрастыру; жоба жасау; жоба* |
| desk *n* | [desk] | *парта; стол канц.* | *парта, жазу үстелі* |
| destination *n* | [ **͵**dest **I** ne**I** ∫(ә)n] | *место назначения; адресат; получатель* | *тағайындалған орын; адресат; алушы* |
| detect *v* | [d**I** ่tekt] | *обнаруживать; выявлять; определять* | *табу; айқындау; анықтау* |
| determine *v* | [d**I** ่tә**:**m **I** n] | *определять* | *анықтау* |
| developing *n,* *v* | [d**I** ่velәp**I** ŋ] | *развитие; разработка; развивать; разрабатывать;* | *даму, өркендеу, тереңдету; өңдеу; істеп қою; құру; зерттеме,талдама* |
| device *n* | [d**I** ่va**I**s] | *устройство; прибор* | *құрылғы; құрал, аспап* |
| dictionary *n* | [ ่d**I** k∫(*ә*)nr **I**] | *словарь (книга)* | *сөздік (кітап)* |
| digital *a,* *n* | [d**I** ่dʒ**I** tәl] | *цифровой; цифра* | *цифрлық; цифр* |
| different *a* | [่d**I**f(ә)rәnt] | *различный; другой; отличный* | *әр түрлі; басқа, өзгеше; өте жақсы, озат* |
| difficult *a* | [ ่d**I**f **I**k(ә)lt] | *трудный* | *қиын, ауыр* |
| directed (to) *a* | [d**I** ่rekt **I**d (tә)] | *направленный (на);* | *нұсқалған (-ға,-ге,-қа, -ке,-а,-е);* |
| direct *v* | [d**I** ่rekt] | *направлять* | *нұсқау; дәлдеу* |
| discrimination *n* | [d**I**s **͵** kr **I** m **I** ne **I** ∫(ә)n] | *дискриминация; различие; распознавание; условный переход* | *кемсітушілік;ерекшелік;*  *айырмашылық; білу, айырып тану; шартты өту* |
| disperse *v* | [d**I**s ่pә**:**s] | *рассеивать; разбрасывать* | *таралу; шашып тастау,*  *жаю ; таратып жіберу* |
| display *v* | [d**I**s่ple**I**] | *выводить на дисплей; показывать* | *дисплейге шығару; білдіру; көрсету* |
| distinct (from) *a* | [ ่d**I**st **I** ŋkt (frәm)] | *отличный (от)* | *анық; бөлек, жеке; белгілі*  *(–нан,-нен;-дан,-ден;-тан, тен)* |
| divide *v* | [d**I** ่va**I**d] | *делить (на части)* | *бөлу; бөлініп шығу (бөлікке)* |
| (a) do-it-yourself *n* | [(ә) ่du**: I** t jo**:** ่self] | *самостоятельное выполнение* | *өз бетінше орындау* |
| drain *n* | [dre**I** n] | *утечка; непроизводительный расход* | *жойылу, кему; дайындалмайтын (өнімсіз, босқа кеткен) шығыс* |
| drawing *n* | [ ่drɔ**:I** ŋ] | *рисование; черчение; иллюстрация ; граф. изображение* | *сурет; иллюстрация; графикалық кескін* |

***E***

|  |  |  |  |
| --- | --- | --- | --- |
| еach *pron* | [่ i**:**t∫] | *каждый* | *әркім, әрқайсысы* |
| either (… or …) *adv* | [ a**I**(i**:)**ðә (ɔ **:)**] | *либо… либо; или … или* | *әйтпесе; де, да; екеудің біреуі* |
| empty *a* | [ ่empt**I**] | *пустой; свободный* | *бос, қуыс; еркін* |
| enable *v* | [**I** ่ne**I**bl] | *разблокировать; давать возможность* | *босату; мүмкіншілік беру* |
| endless *a* | [ **่**endl**I**s] | *бесконечный* | *шексіз, бітпейтін, аяқталмайтын* |
| engaged (in) *a* | [**I** n ่ge**I**dʒd] | *занятый ч.-л.* | *(бір нәрсемен) қолы бос емес; шұғылданатын, айналысатын* |
| enormous *a* | [**I** ่nɔ**:**mә**s**] | *огромный; ужасный* | *өте үлкен, еңгезердей; тым; үрейлі* |
| equivalent *n, (a)* | [**I** ่kw **I** vәl(ә)nt] | *эквивалент (-ный)* | *тең,бірдей; эквивалент(-ті)* |
| essential *a* | [**I** ่sen∫(*ә*)l] | *важный; существенный* | *маңызды, негізгі* |
| establish *v* | [**I**s ่tæbl**I**∫] | *учреждать; устанавливать* | *орнату, орналасу; құру, қою* |
| exacting (~rules) *a* | [**I**g ่zækt **I** ŋ (ru**:**lz)] | *требовательный (строгие правила)* | *қатаң; талғампаз; талап етушi (қатаң ереженi)* |
| exactly *adv* | [**I**g ่zæktl**I**] | точно; верно | *дәл, тура; әрине* |
| examine *v* | [Ig ่zæm I n] | *рассматривать; исследовать; изучать; проверять* | *тексеру, сынау; зертеу; үрену, оқып білү;* |
| example *n* | [**I**g ่zæmpl] | *пример* | *мысал; үлгі* |
| excess *v* | [**I** ่kses] | *превышать;* | *асыру, өсіру* |
| even *a* | [่i**:**v(ә)n] | *ровный; чётный* | *тегіс; бірдей, тең, ұқсас; жұпты* |
| event *n* | [**I** ่vent] | *событие; исход; результат* | *оқиға; жағдай; нәтиже; қорытынды* |
| evry | [ **่**evr **I**] | *каждый; всякий* | *әрбір, әр; әркім, кім көрінген* |
| evryday | [่evr **I**da**I**] | *ежедневный* | *күнделікті* |
| exist *v*  *(=to be, to live)* | [**I**g ่z**I**st] | *существовать, жить, быть* | *болу, бар болу; өмір сүру* |
| expense *n* | [**I**ks ่pens] | *расходы; (за)траты* | *шығыс; шығын; жұмсау* |
| expensive *a* | [**I**ks ่pens**I**v] | *дорогой; затратный* | *қымбат* |
| explain *v* | [**I**ks ่ple**I** n] | *объяснять* | *түсіндіру, ұғындыру* |
| explosion *n* | [**I**ks ่plouʒ(ә)n] | *взрыв; бурный рост* | *жарылу, бұзылу; қарқындап өсуі* |
| extracted (from) *a* | [**I**ks ่trækt **I**d (frәm)] | *извлечённый (из);*  *полученный(из)* | *сығып шығарылған; алынған(-дан,-ден,*  *-тан,-тен,-нан,-нен)* |
| extracting *n* | [**I**ks ่trækt **I** ŋ] | *извлечение* | *шығару; пайда болу* |
| extremely *adv* | [**I**ks ่tri**:**ml**I**] | *чрезвычайно; крайне; очень* | *тіпті, өте, аса* |

***F***

|  |  |  |  |
| --- | --- | --- | --- |
| face *n* | [fe **I** s] | *лицо; циферблат; лицевая сторона; плоская поверхность* | *бет, үсті; циферблат; сыртқы түр; жазық, тегіс бет* |
| (in) fact | [(**I** n) ่fækt] | *(фактически; действительно;)* | *шындық; шынында, расында; шын мәнісінде* |
| fall (on) *v* | [fɔ**:**l (ɔn)] | *пáдать; (нападáть)* | *құлау; (шабуыл жасау)* |
| familiar *a* | [fә ่m **I**ljә] | *знакомый; обычный; известный* | *жақын, таныс; дағдылы; белгілі* |
| fast *a* | [fα**:**st] | *быстрый* | *жылдам, шапшаң; мықты, берік* |
| feat *n* | [fi**:**t] | *подвиг; искусство; ловкость* | *ерлік, батырлық; өнер, шеберлік; ептілік* |
| feed ( into) (to) *v* | [fi**:**d (**I** ntә) (tә)] | *кормить; питать; снабжать; (подавать в)* | *тамақ жегізу;*  *жабдықтау, қамтамасыз ету* |
| (a) few | [(ә) fju**:**] | *мало; (несколько)* | *аз, азғана; (біраз, бірнеше)* |
| field *n* | [fi**:**ld] | *поле; область; зона; группа разрядов* | *өріс; аймақ; аумақ; разряд тобы* |
| figure (out) *n, (v)* | [่f**I**gә(r aut)] | *цифра; число; показатель; рисунок; (вычислять)* | *цифр; сан; көрсеткіш; сурет; (есептеу, есептеп шығару)* |
| file *n* | [fa**I**l] | *файл* | *файл; картотека* |
| fill (up) *n,* *v* | [f **I**l (ʌp)] | *заполнение (-итель); заливка; заполнять* | *толу; толтыру; қосу; толтырғыш* |
| finally *adv* | [ ่fa**I** nәl**I**] | *в конце концов;*  *в заключение; окончательно* | *соңында; ақырында, ең соңында; ақырғы, соңғы* |
| first | [fә**:**st] | *первый; во-первых* | *бірінші; біріншіден, алдымен* |
| fix *n, v* | [f **I**ks] | *исправление; закреплять; настраивать; налаживать* | *түзеу, баптау; бекіту; белгілеу; жөндеу, реттеу; жолға салу* |
| flight *n* | [fla**I** t] | *полёт; авиа полёт; авиа рейс* | *ұшу; әуе ұшу; әуе рейсі* |
| (by) following *a* | [(bai) ่fɔlou**I** ŋ] | *следующий; (следующим; выполняя; последующим)* | *кейінгі, келесі* |
| forecast *n,* *v* | [ ่fɔ**:**kαst] | *прогноз(-ировать); предсказывать* | *болжау, күні бұрын айту* |
| foresee *v* | [fɔ ่si**:**] | *предвидеть; знать заранее* | *алдын ала байқау, бұрын білу* |
| form *n, v* | [fɔ**:**m] | *анкета; бланк; представление; формировать; образовывать* | *сауалнама; ұсыныс; тапсыру, әкеліп өткізу; қалыптастыру* |
| from *prep* | [frәm] | *от; из; с* | *-тан,-тен; -дан,-ден;-нан, -нен* |
| future *a* | [ ่fju**:**t∫ә] | *будущ-ий (-ее, -ая, -ие)* | *келешек, келер* |

***G***

|  |  |  |  |
| --- | --- | --- | --- |
| gadgetry *n* | [ ่gædʒ**I** tr **I**] | *приспособление; способ* | *құрал-сайман; әдіс* |
| garment *n* | [ ่gα**:**mәnt] | *одежда; покров; одеяние* | *киім, зат; үстінгі қабат, жоғарғы бет* |
| general *a* | [ ่dʒenәr(*ә*)l] | *общий; обычный; полный; всеобщий* | *жалпы; негізгі* |
| generate *v* | [ ่dʒenәre**I** t] | *генерировать; создавать; образовывать; порождать; производить* | *жасап шығару, жасау, тудыру, құрастыру; тарату; өндіру, істеп шығару* |
| getting *n,* *v* | [get **I** ŋ] | *получ –ение (-ать)* | *алу; табу; әкелу, апару* |
| give *v* | [g **I**v] | *(от-) давать* | *беру; жолын көрсету* |
| great *a* | [gre**I** t] | *великий; большой* | *ұлы; зор; тамаша; тәжірибелі; күшті* |
| group *n, v* | [gru**:**p] | *группа; группировать* | *топ; топталау, бірігу* |

***H***

|  |  |  |  |
| --- | --- | --- | --- |
| happen *v* | [ ่hæp(ә)n] | *случаться, происходить* | *болып қалу; сәті түсу* |
| hardware *n* | [ ่hα**:**dwεә] | *оборудование; аппаратура* | *құрал-сайман, жабдық* |
| heart *n* | [hα**:**t] | *сердце; суть; основа; основной компонент; ядро* | *жүрек; ядро, өзек, түйін; маңызды құрамдас* |
| high *a* | [ha **I**] | *высокий; старший; сильный;*  *с бόльшим адресом; дорогόй* | *ұзын, жоғарғы, негізгі; қымбат; биік; күшті; қымбат* |
| himself *pron* | [h**I** m ่self] | *себя; собой; себе; сам; самому; самим* | *өзі, өзіне, өзін* |
| hold *v* | [hould] | *вмещать; схватить; завладеть; держать; проводить* | *басып алу; ұстау; өткізу* |
| hooves *n,* *pl* ( hoof *sg*) | [hu**:**vz]  ([hu:f]) | *копыта*  *(копыто)* | *тұяқтар (тұяқ)* |
| horse *n* | [hɔ**:s**] | *лошадь; конь* | *ат; жылқы* |
| how (much)(many) | [hau  (mʌt∫) ('men**I**)] | *сколькo?* | *қанша? неше?* |
| however *conj, adv* | [hau ่evә] | *однако, тем не менее; как (какой) бы ни* | *қалай болса да; қарамастан, дегенмен, болып қалу; сәті түсу* |
| hundred *n* | [ ่hʌndr **I**d] | *сто; сотня* | *жүз* |

***I***

|  |  |  |  |
| --- | --- | --- | --- |
| identif y *v* (-ied *a*) | [a**I** ่dent **I**fa**I (**d)] | *идентифицировать; распозна-вать-нный* | *ұқсастығын, бірдейлігін табу; ұқсату; анықтау; табу; анықталған* |
| if *conj* | [**I** f] | *если; ли* | *егер* |
| illogical *a* | [**I** ่lɔdʒ**I**kәl] | *нелогичный; непоследовательный* | *қисынды емес* |
| immediate *a* | [**I** ่m**I**djәt] | *немедленный; непосредственный* | *тез, жылдам; тікелей, тура* |
| important *a* | [**I** m ่pɔ**:**tәnt] | *важный; значимый* | *маңызды, көңіл аударарлық* |
| indeed *adv* | [**I** n ่di**:**d] | *действительно; в самом деле* | *шышында, расында, әрине* |
| individual *a* | [**I** nd **I ่**v **I** djuәl] | *отдельный; индивидуальный* | *жеке, дара , дербес* |
| industrial *a* | [**I** n ่dʌstr **I** әl] | *промышленный; индустриальный* | *өнеркәсіпті; индустриялы* |
| information *n* | [**I** nfә ่me **I**∫(*ә*)n] | *информация* | *жаңалық, мәлімет, ақпарат* |
| informative *a* | [**I** n ่fɔ**:**mәt **I** v] | *информативный; содержательный* | *ақпараттық; маңызды* |
| initial *a* | [**I** ่n**I**∫**I**әl] | *начальный; исходный* | *басқы; бастапқы; бірінші* |
| input *n* | [ **่I** nput] | *ввод (-ная информация)* | *енгізу; (енгізу мәліметі)* |
| internal *a* | [**I** n ่tә**:**n(ә)l] | *внутренний* | *ішкі* |
| insignificant *a* | [ **͵ I** ns**I**g ่n **I**f **I** k(ә)nt] | *незначительный; неважный* | *есепке алғысыз; маңызсыз* |
| instance *n* | [ ่ I nstәns] | *пример; копия; экземпляр* | *үлгі, мысал; көшірме; дана* |
| integrated *a* | [**I** ntә ่gre**I**t **I** d] | *объединённый; интегральный; встроенный* | *біріктірілген, кірістірілген,*  *құрамдас; жалғастырылған* |
| intelligence *n* | [**I** n ่tel**I**dʒ (ә)n s] | *сведения; информация; интеллект* | *мәліметтер; ақпарат; ой, ақыл, парасат* |
| introduction *n* | [ **͵ I** ntrә ่dʌk∫(ә)n] | *введение;*  *внедрение; объявление о выпуске* | *кіргізу, енгізу; кіріспе; ендіру; тарату; жаңалық кіргізу; шығару туралы жариялау* |
| inventing *n* | [**I** n ่vent**I** ŋ] | *изобретение (процесс)* | *жаңалық ашу; ойлап шығару* |
| invoice *n* | [' **I** nvo **I**s] | *расчёт; счёт; смета; накладная; учёт* | *есеп, есептеу; тіркеме қағаз; есепке алу* |
| involve (in) *v* | [**I** n'vɔlv (**I** n)] | *вовлекать; затрагивать; включать в себя* | *кірістіру, қатынастыру; жанап өту, тиіп кету; қосу, жүргізу* |
| isolation *n* | [a**I**sә ่le **I**∫(ә)n] | *изоляция; развязка; локализация (неисправности)* | *оңашалау, айыру; (жарамсыздықты) оқшалау,*  *шешу, табу; шектеу* |
| issue *n,* *v* | [ ่ **I**∫(s)u] | *выпуск; выдача; вопрос (трудный); выдавать* | *басып шығару; шығару;беру; (қиын) сұрақ; тарату* |
| item *n* | [ ่a**I** tәm] | *элемент; предмет; единица; элементарная группа; позиция* | *элемент, зат, бұйым; бірлік; элементтер тобы; орын* |
| itself *pron* | [**I** t ่self] | *сам; само; сама* | *өзім; өзің, өзі* |

***J***

|  |  |  |  |
| --- | --- | --- | --- |
| job *n* | [dʒɔb] | *задание* | *тапсырма ; жұмыс, іс; орын, қызмет* |
| just as *adv* | [dʒʌst æ(ә)z] | *как раз так* | *тап солай* |
| just *adv* | [dʒʌst] | *как раз; именно; просто* | *дұрыс, дәл, нақ ; жай, оңай* |
| journey *n* | [ ่dʒә**:**n **I**] | *путеществие; поездка; проход* | *сапар, жол; өтетін жер* |
| jotting (down) *n* | [ ่dʒɔt**I** ŋ (daun)] | *краткая беглая запись* | *қысқашасы; қысқаша жазып алу* |

***K***

|  |  |  |  |
| --- | --- | --- | --- |
| kit *n* | [k **I** t] | *комплект; набор; конструктор* | *жинақтама, толым; жиынтық, теру, терім; конструктор, құрылымдаушы* |
| knit *n, v* | [n **I** t] | *вязать, вязание; соединение; стараться; соединяться* | *тоқу, тоқыма; қосылыс, қосылу, бірігу; тырысу; қосу, жалғастыру, біріктіру* |
| knitting *n* | [ ่n**I** t**I** ŋ] | *вязание; старание;* | *тоқу, тоқыма; ынталану, тырысу; қосу; жалғастыру* |
| know(n) *v* | [nou(n)] | *знать; (известный)* | *білу, тану (белгілі, атақты)* |

***L***

|  |  |  |  |
| --- | --- | --- | --- |
| language *n* | [ ่læŋgw**I**dʒ] | *язык* | *тіл* |
| large *a* | [lα**:**dʒ] | *большой* | *үлкен, ірі* |
| last (over) *v* | [lα**:**st (ouvә)] | *длиться; продолжаться* | *созылу, ұзару; ұласады* |
| (the) latter *a* | [(ðә) ่lætә] | *последний (второй из двух перечисленных)* | *соңғы, кейінгі* |
| lawyer *n* | [ ่lɔ**: I**ә] | *юрист; адвокат; законовед* | *юрист(заң қызметкері), кеңес беруші, заң жөнінен ақыл* |
| length *n* | [leŋθ] | *длина; продолжительность* | *ұзындық, қашықтық; кесінді* |
| library *n* | [ ่la**I** brәr **I**] | *библиотека* | *кітапхана* |
| liken (to) *v* | [ ่la**I**kәn (tә)] | *сравнивать; находить сходство (с)* | *cалыстыру; ұқсастық табу (дан,-ден;-нан, -нен)* |
| limit *v* (-ed) (to) | [ ่l**I** m**I** t-**I**d (tә)] | *oгранич-ивать*  *(-енный) чем-либо* | *бір нәрсеге шек қою* |
| line *n* | [ la**I** n] | *линия (связи); шина; строка; серия* | *сызық, тізбек; шек; рет, қатар; жол (кітап, газет туралы)* |
| list *n* | [l**I**st] | *список; перечень; таблица; ведомость* | *тізім; тізбе; кесте; мәлімет* |
| little *a* | [ ่l**I** tl] | *маленький; небольшой* | *кішкентай, қысқа, шағын; аз, аз ғана* |
| load *n,* *v* | [loud] | *загрузка; нагрузка; загружать* | *жүктеме, жүктеу; тиеу; арту, салмақ салу, жасау* |
| locality *n* | [lou ่kæl**I** t **I**] | *место нахождения; локальность* | *тұрған жері; жер, жергіліктік* |
| (pockets or storage) location *n* | [( ่pɔk**I** ts ɔ**:** ่stɔ**:**r **I**dʒ) lou ่ke **I**∫(*ә*)n] | *определение места; расположение; ячейка; адрес ячейки; микрорайон сети* | *орынды анықтау;*  *орналастыру; ұяшық; ұяшық адресі; ықшамаудан желі* |
| looking up *n* | [ ่luk**I** ŋ ʌp] | *поиск* | *бір нәрсені іздеу,қарау* |
| loosely *adv* | [ ่lu**:**sl**I**] | *слабо* | *нашар, әлсіз* |

***M***

|  |  |  |  |
| --- | --- | --- | --- |
| machine *n* | [mә ่∫i:n] | *машина; механизм; устройство* | *машина; механизм; аппарат; құрылғы, құру* |
| machinery *n* | [mә ่∫i**:** n(*ә*) r **I**] | *машины; оборудование; техника* | *машиналар;*  *машинаның бөлшектері; техника* |
| make (up) *v* | [ me **I** k (ʌp)] | *делать;*  *делать монтаж ; (составлять; компоновать)* | *жасау, істеу; (толтыру, реттеу, жөндеу); монтаж жасау* |
| management *n* | [ ่mæn**I**dʒm*ә*nt] | *управление;*  *админитсртирование; организация* | *басқару, меңгеру; басқарма, әкімшілік; ұйым* |
| manager *n* | [ ่mæn**I**dʒә] | *менеджер; организатор; управляющая программа; диспетчер; подсистема управления; устройство управления* | *меңгеруші; басшы;*  *ұйымдастырушы; басқарушы программа; басқару құрылғысы;*  *диспетчер;*  *бағыныңқы жүйе басқаруы; басқару құрылғысы* |
| manufacturer *n* | [ **͵**mænju่fækt∫әrә] | *производитель; изготовитель* | *өндіруші; жасап шығарушы* |
| mean(-s *n*) *a, v* | [ mi**:**n(z)] | *злой; средний; значить; иметь значение; (средство)* | *мейрімсіз, ашулы;*  *орташа, әлсіз; амал; арқылы; мәні болу* |
| meaningless *a* | [ ่mi**:**n**I** ŋl**I**s] | *бессмысленный* | *мәнсіз; мағынасыз* |
| measurable *a* | [ ่meʒ(*ә*)rәbl] | *измеримый* | *өлшенерлік; өлшеуге келетін* |
| measure *n,* *v* | [ ่meʒә] | *мера; оценка; критерий; масштаб; измерять* | *өлшем; бағалау, баға; критерий, белгі; көлем, масштаб; өлшеу* |
| measurement *n* | [ ่meʒәment] | *измерение; замер; размеры* | *өлшеу, өлшем; көлемі* |
| membership *n* | [ ่membә∫**I** p] | *членство; принадлежность; число членов* | *мүше болып тұру; құрамында болушылық; мүшелер саны* |
| modify *v* | [ ่mɔd**I** fa **I**] | *изменять; модифицировать; корректировать* | *өзгерту; анықтау; түзету* |
| money *n* | [ ่mʌn**I**] | *деньги* | *ақша* |
| more (than) *a* | [mɔ**:** (ðæn)] | *более; больше (чем); ещё* | *көбірек, үлкенірек; (-ға қарағанда,гөрі)*  *тағы, қосымша* |
| mostly *adv* | [ ่moustl**I**] | *наиболее; наибольшее; самое большее* | *ең алдымен, көбінесе; бәрінен де, өте* |
| multiply *v* | [ ่mʌlt **I**pla**I**] | *умножать; раз-множить* | *арттыру; көбейту* |

***N***

|  |  |  |  |
| --- | --- | --- | --- |
| need *v n* | [ni**:**d] | *нуждаться; иметь необходимость; нужда; потребность* | *қажеттілік, керектілік; керек болу, мұқтаж* |
| new *a* | [nju**:**] | *новый* | *жаңа* |
| next *a* | [nekst] | *следующий* | *келесі; содан кейін* |
| none *pron* | [nʌn] | *ничто; никто* | *ештеме; ешқандай; ешкім; біреу де емес* |
| nothing *pron* | [ ่nʌθ **I** ŋ] | *ничто* | *ештеме; түкке тұрмайтын зат* |
| nowadays ***adv*** | [ ่nauәde**I**z] | *сейчас; в наши дни* | *қазір; қазіргі күндерімізде* |
| number *n* | [ nʌmbә] | *число; цифра; номер; количество* | *сан; нөмір; өлшем; мөлшер сан* |
| numeric *a* | [nu**:** ่mer **I**k] | *числовой; цифровой* | *сандық, цифрлық.* |

***O***

|  |  |  |  |
| --- | --- | --- | --- |
| obey *v*  obeyed *a* | [ә ่be**I**]  [ә ่be**I**d] | *подчиняться; подчинённый* | *бағыну; тыңдау; айтқанды істеу; қол астындағысы* |
| obeying *a* | [ә ่be**I I** ŋ] | *послушный; выполняющий* | *тіл алғыш; орындайтын* |
| obtain *v* | [әb ่te**I** n] | *получать; добиваться* | *алу; (айтқанына) жету* |
| obviously *a, adv* | [ ่ɔbv**I**әsl**I**] | *очевидный; очевидно* | *сірә; анығы, анық* |
| (at) once ***adv*** | [(әt) wʌns] | *однажды;*  *(сразу же)* | *бір күні, бір кезде; (бірден)* |
| open *v, a* | [oup(ә)n] | *открыть; открытый* | *ашу; ашығы* |
| operate (on) *v* | [ ่ɔpәre**I** t (on)] | *делать операцию (оперироваться); работать (на); выполнять; действовать* | *операция жасау; жұмыс істеу; жүргізу, орындау; қимылға келтіру* |
| operator *n* | [ ่ɔpәre**I** tә] | *оператор* | *оператор (өндірістік әрекеттерді жүргізуші адам)* |
| or *conj* | [ɔ**:**] | *или; либо* | *я болмаса; немесе* |
| order *n* (*v)* | [ ่ɔ**:**dә] | *приказ (-ывать); заказ (-ывать)* | *бұйрық; бұйрық беру; тапсырыс;*  *тапсырыс беру* |
| other *pron* | [ ่ʌðә] | *другой; иной* | *басқа, бөлек; қосымша, басқаша* |
| output *n* | ['аutput] | *вывод (информации); выход продукции* | *(информацияларды) шығару; өнімнің шығарылымдығы; қорытынды* |
| outside *a* | ['аut ่sa**I**d] | *внешний* | *сырт жағы, бет жағы; сыртқы; шеткі* |
| overall *a* | ['ouvәrɔ**:**l] | *общий; полный; суммарный; всеобъемлющий;*  *в общем; в целом* | *ортақ; толық; жалпы алғанда; бүтіндей, тұтас* |
| overdraft *n* | ['ouvәdr**:**αft] | *превышение своего кредита в банке* | *өз қарызының банкте асып кетуі* |
| own *a, v* | [oun] | *(свой) собственный; родной (родственник); иметь; владеть* | *өз, өзінікі;*  *жекеменшік; өзіне меншікті; туған*  *(туыс); болу,*  *бар болу; ие болу; бағындыру* |

***P***

|  |  |  |  |
| --- | --- | --- | --- |
| part *n,* *v* | [pα**:**t] | *часть; доля; раздел; фрагмент; отделять; разделять* | *бөлік, бөлім, тарау; үлес; бөлу, бөлік; фрагмент, үзінді; бөліп алу; бөліп тастау* |
| particular *n,* *a* | [pә่t **I**kjulә] | *частность; подробность; деталь; частный; особый* | *жекелік, ерекшелік; толықтылық; бөлшек; нақтылау; жеке;*  *ерекше* |
| patient *n, a* | [ ่pe **I**∫(*ә*)nt] | *пациент; больной; терпеливый* | *пациент; емделуші; шыдамды, төзімді* |
| pattern *n* | [ ่pætәn] | *образец; шаблон; систематизиро- ванная структура; строение; модель; трафарет; узор; кодовая комбинация; схема; диаграмма; рисунок; образ; изображение* | *үлгі; кескін; жүйелік құрылым; комбинация коды; сұлба; кесте; сурет; кескін, кескіндеу; бейне* |
| payment *n* | [ ่pe**I** mәnt] | *уплата; выплата; платёж* | *төлем, төлеу; ақы; құн; өтем, өтелетін қаржы* |
| payroll *n* | [ ่pe**I** roul] | *платёжная ведомость* | *төлем тізімі* |
| penknife *n* | [ ่penna**I**f] | *перочинный нож* | *бәкі пышақ* |
| people *n* | [ ่pi**:**pl] | *люди; народ* | *халық* |
| perform *v* | [pә่fɔ**:**m] | *выполнять; исполнять; производить* | *істеу, орындау; атқару;*  *өнім шығару* |
| performing *n* | [pә่fɔ**:**m**I** ŋ] | *выполняя; во время выполнения* | *орындап отырған кезінде* |
| peripheral *n, a* | [pe่r **I**f(*ә*)rәl] | *внешнее (периферийное) устройство; внешний; периферийный* | *сыртқы құрылғы; сыртқы* |
| (by) performing | [(ba**I**) pә ่fɔ**:**m**I** ŋ] | *(посредством; с помощью) выполнения; выполняя* | *орындау арқылы;*  *үйретілген, жаттықан* |
| perhaps *adv* | [pә่hæps, præps] | *возможно* | *мүмкіндігінше, мүмкін, ықтимал* |
| picture *n* | [ ่p**I**kt∫ә] | *картина; рисунок; изображение; шаблон; образец* | *сурет; үлгі;*  *бейнелеу; кескін* |
| piece *n* | [pi**:**s] | *кусок; часть; доля* | *бөлік, бөлшек; үзінді* |
| pilot *v, a* | [ ่pa**I**lәt] | *вести; управлять; пробный; эксеперименталь-ный; первоначальный; опытный* | *жүргізу; басқару; меңгеру; байқалатын, сыналатын, тексерілетін; бастапқы; тәжірибелі* |
| place *n, v* | [ple **I** s] | *место; положение; разряд; помещать; размещать;* | *орын; жағдай; орналастыру, қою; белгілеу; топ, класс* |
| place (in) *v* | [ple **I** s (**I** n)] | *поместить*  *(во что-либо, куда-либо)* | *( бір нәрсеге, бір жерге) орналастыру, орнықтыру* |
| play (а part) *n, v* | [ple **I** (ә pα**:**t)] | *игра; выполнять; действовать; воспроизводить; проигрывать (сценарий, пьесу); (играть роль)* | *ойын; орындау, өтеу;*  *әрекет ету; іс қылу; (пьесаны, сценарийді, роль ойнауды) еске түсіру* |
| plug (into files or pockets) *v* | [plʌg (**I** ntә)  fa**I**lz ɔ**:** ่pɔk**I** ts] | *разъём; вилка; штепсель; штекер; (вставлять в гнездо)* | *тығын; кран; аша (электр); қосу, жалғау; ұяға қондыру* |
| politician *n* | [ **͵**pәl**I** t **I**∫(ә)n] | *политик* | *саятсатшы* |
| potential *n,* *a* | [pә ่ten∫(*ә*)l] | *потенциал; потенциальный; возможный* | *мүмкіндік****;*** *шама, әлеует құдіретті; мүмкін, қисыны бар;* |
| power *n* | [ ่pauә] | *мощность; энергия; степень; показатель степени; произво- дительность; вычислительная мощность; способность* | *күш, қуат,*  *үстемділік; дәреже; дәреже көрсеткіші; өнімділік; есептеуіш күш; қабілеттілік* |
| precedent *a* | [pr **I** ่si**:**d(ә)nt] | *предшествующий* | *өткен, болған; алдыңғы* |
| precedent *n* | [่pres**I** d(ә)nt] | *прецeдент* | *оқиға, өткенде орын алған* |
| precise *a* | [pr **I** ่sa**I**s] | *точный; определённый* | *дәл, анық; жинақты, ұқыпты; айқын, белгілі* |
| pre-determined *a* | [pr **I**d**I** ่tә**:**m**I** nd] | *предварительный; предопределённый* | *алдын ала шешу, алдын ала болжау* |
| price *n* | [pra **I**s] | *цена* | *баға, құн* |
| procedure *n* | [prә ่si:dʒә] | *процедура; процесс; методика* | *рәсім, тәртіп; процесс; барыс; әдістеме* |
| prodigious *a* | [prә ่d**I**dʒәs] | *огромный; невероятный; удивительный; изумительный* | *үлкен; тамаша, ғажап; таңырқарлық* |
| produce *v* | [prә ่dju:s] | *производить* | *шығару, өндіру; (өндірілетін)* |
| produced *a* | [prә ่dju:st] | *производимый* |  |
| program(m)е *n*  program(m)er *n* | [ ่prougræm]  [ ่prougræmә] | *программа; прoграммист* | *программа,*  *бағдарлама;*  *программист,*  *бағдарламашы* |
| proof *n,* *a* | [pru**:**f] | *доказательство; проверка; проба; обеспечение защиты; защищённый; непроницаемый* | *дәлел; тексеру; сынақ; қорғауды қамтамасыз ету; қорғалған;*  *өткізбейтін* |
| provide *v* | [prә ่va**I**d] | *обеспечивать; предоставлять; снабжать; предвидеть; предусматривать* | *жабдықтау, дайындап қою; қамтамасыз ету; болжау* |
| provided *conj* | [prә ่va**I**d**I**d] | *при условии; если только* | *шартпен, егерде* |
| purchase *v* | [ **่**pә**:**t∫әs] | *покупать; приобретать; закупать* | *сатып алу; толайым сатып алып қою* |
| purl *n,* *v* | [pә**:**l] | *журчание; журчать* | *сылдыр; сылдырлау* |
| put (to) *v* | [put (tә)] | *положить; поставить; поместить что-либо (на)* | *қою; салу, жатқызу; бір нәрсені орналастыру (-ға,-ге,-қа,-ке,-a,-е)* |
| purpose *n* | [ ่pә**:** pәs] | *цель* | *мақсат* |

***Q***

|  |  |  |  |
| --- | --- | --- | --- |
| quantity *n*  quantities *n* | [ ่kwɔnt **I** t **I** ]  [ ่kwɔnt **I** t **I** (z)] | *количество; величина; величины; количества* | *мөлшер, көлем; шама;*  *мөлшерлер, көлемдер* |
| quickly *adv* | [ ่kw **I**kl**I**] | *быстро* | *жылдам* |
| quite *adv* | [kwa**I** t] | *вполне; довольно; совершенно; совсем; всецело; очень* | *әбден; едәуір; толық, жалпылама* |

***R***

|  |  |  |  |
| --- | --- | --- | --- |
| range *n,* *v* | [re I ndʒ] | *диапазон, область, интервал; размер выборки; широта распределения; отрезок; область значений; заключаться в пределах* | *ауқым; аймақ; аралық; таңдама көлемі; кеңдікті үлестіру; кесінді; мәндер аймағы* |
| raw *a* | rɔ**:**] | *сырой; необработанный; непосредственный* | *шикі; өңделмеген* |
| react *v* | [r **I** ่ækt] | *реагировать; отзываться* | *әсер ету, шағылысу; бірлесіп әрекет ету* |
| reach *v* | [ri**:**t∫] | *достигать; доходить; простираться* | *жету, созылу, естілу* |
| read (into) *v* | [ri**:**d (**I** ntә)] | *вводить данные в* | *деректерді енгізу* |
| ready *a* | [ ่red **I**] | *готовый; подготовленный;*  *лёгкий* | *даяр, дайын; ынталы;*  *дайындалған; жеңіл, оңай* |
| reason *n* | [ ่ri**:**zn] | *причина* | *себеп* |
| refer (to) *v* | [r **I** ่fә**:** (tә)] | *ссылаться (на); обращаться (к); относиться (к); справляться (о ч.-л.)* | *сілтеу, сілтеп жіберу*  *(-ға, -ге,- қа, -ке), жөн сұрау; қатысы болу; айналдыру; қарап анықтау* |
| register *n, v* | [ ่redʒ**I**stә] | *регистр; регистрировать; накапливать; суммировать* | *регистр; журнал, тізім; жазу, тіркеу; жинақтау* |
| related (to) *a* | [r **I** ่le**I** t **I** d (tә)] | *связанный (с); относящийся (к); зависимый (от)* | *байланысты болу; қатысты болу; бағынышты, тәуелді* |
| (by the) relation | [(ba**I** ðә) r **I** ่le**I** ∫(ә)n] | *oтношениe;*  *(по отношению к)* | *бұл жөнінде, бұл турасында* |
| relatively *adv* | [ ่relәt **I**vl**I**] | *относительно* | *салыстырмалы; біршама; туралы, жөнінде* |
| repertoire *n* | [ ่repәtwα**:**] | *набор;*  *совокупность;*  *репертуар* | *жиын; жинақ, жиынтық; теру; репертуар* |
| repetitive *a* | [r **I** ่pet **I** t **I**v] | *повторенный; повторяемый* | *қайталанатын; пысықталған* |
| represent *v* | [ **͵**repr **I** ่zent] | *представлять; изображать* | *көрсету; көз алдына елестеу; бейнелеу* |
| (for this) requirement *n* | [fɔ**:** ð**I**s r **I** ่kwa **I** әmәnt] | *(согласно этому ~) требование* | *шарт, талап; қажеттік (бойынша)* |
| requiring *n,* *a* | [r **I** ่kwa **I**әr **I** ŋ] | *требование; требующий* | *шарт, талап; талап ететін* |
| responsible (for) *a* | [r **I** ่spɔnsәbl (fo**:)**] | *ответственный (за)* | *(бір нәрсе үшін) жауапты, міндетті; сенімді* |
| returned (to) *a* | [r **I** ่tә**:**nd tә] | *возвращённый (к, в)* | *қайтарылынған (-ға,-ге,-қа, -ке,-на,-не)* |
| room *n* | [rum] | *комната; помещение; пространство; место* | *бөлме, пәтер; кеңістік; орын* |
| routine *n* | [ru(**:)** ti**:**n] | *программа; подпрограмма* | *әдеттегі іс; тәсіл;*  *іш программа; бағыныңқы*  *програм ма* |
| rule *n* | [ru**:**l] | *правило;*  *закон;*  *масштаб;*  *линейка* | *ереже; жоба; басқару, үстемдік; сызғыш; шама, өлшем* |

***S***

|  |  |  |  |
| --- | --- | --- | --- |
| (the) same *pron* | [(ðә) seI m] | *oдинаковый;*  *(тот же) самый* | *дәл өзі; бірдей* |
| save *v* | [seI v] | *сохранить; сэкономить; спасти* | *сақтау; үнемдеу; құтқару; жарылқау* |
| scheduling *n* | [ ่∫edjul **I** ŋ] *Br E* [ ่skedjul**I** ŋ] *Am E* | *составление расписания, графика; планирование; диспетчеризация; календарное планирование* | *кесте жасау, кесте; графика; хаттау; жоспарлау; күнтүзбе жоспарлауы* |
| scientific *a* | [**͵**sa**I**әn่t **I**f**I**k] | *научный* | *ғылыми* |
| scope *n, v* | [skoup] | *область, сфера действия; область видимости; контекст; масштаб; размах; область применения; оценка; метка; оценивать; делать отметку* | *аймақ; көруге мүмкіншілігі бар аймақ; ой өрісі; өріс; сала; бағалау; таңба; көлем ; сілтеу, құлаш; өріс алу;*  *тербелеу шегі* |
| seem *v* | [si**:**m] | *казаться* | *көріну, танылу* |
| select *v* | [s**I** ่lekt] | *отбирать; выбирать; выделять* | *таңдап алу; сайлап алу; ерекшелеу* |
| sense *n, v* | [ sens] | *смысл; значение; чувство; ощущение;*  *воспринимать; определять; считывать* | *мағына; сезім, сезу; ес, ақыл; түсіну; анықтау, айыру; сынау, есептеу* |
| sensible *a* | [ ่sensәbl] | *разумный; рассудительный; ощутимый; заметный* | *ақылды, есті; елеулі; көрнекті,*  *айқын* |
| sequence *n,* *v* | [ ่si**:**kwәns] | *последовательность;*  *упор; явочный список значений; натуральный ряд чисел; устанавливать последовательность; упорядочивать; сортировать* | *жүйелік, жүйе, рет; мағыналардың келу тізімі; сандардың натурал- дық қатары; жүйелікті қалыптастыру; іріктеу* |
| (a) series *n* | [(ә) 's**I**әri**:**z] | *последовательность;*  *ряд; серия* | *жүйелік; тізбек; серия* |
| set *n, v* | [set] | *множество; система; набор; совокупность; ряд; серия; комплект; устанавливать; присваивать* | *көптеген, алуан; жүйе; жиынтық, терім; қатар, рет; жинақтама, серия; қою, орнату; иелену,*  *иемдену* |
| to set to work | [tә ่set tә ่wә**:**k] | *приступать к работе* | *жұмысқа кірісу* |
| several *pron* | [่sevr (*ә*)l] | *несколько* | *бірнеше, біраз* |
| sharp *a* | [∫α**:**p] | *резкий; чёткий; острый;* ***символ*** | *анық, айқын; өткір, үшкір; бейне, символ* |
| sharpen *v* | [ ่∫α**:**pәn] | *повысить резкость; заострить;*  *точить (карандаш)* | *анықтығын көтеру; күшейту, сүйрлеу;*  *(қарындашты) өткірлеу* |
| shopping *n* | [ ่∫ɔp**I** ŋ] | *осуществление покупок; хождение по магазинам* | *сауда жасау; дүкенге бару* |
| short *a* | [∫ɔ**:**t] | *короткий; усеченный* | *қысқа, кішкентай; қиылған* |
| significant *a* | [s**I**g'n**I**f**I**kәnt] | *старший; значащий; значительный; важный; существенный* | *үлкен; маңызды, көп мәнді; түбегейлі, мәнді* |
| similar *a* | [ ่s**I**m**I**lә] | *подобный; сходный; схожий* | *ұқсас, сондай, сияқты; лайықты* |
| simple *a* | [s**I** mpl] | *простой; элементарный* | *қарапайым, жеңіл, жай* |
| simplified *a* | [ ่s**I**mpl**I**fa**I**d] | *упрощённый* | *жеңілдетілген, ықшамдалған* |
| simulate *v* | [ ่s**I** mjule**I**t] | *моделировать; имитировать* | *үлгілеу, өңдеу; сылтаурату; бірдеңеге ұқсату* |
| single *a* | [**'**s**I**ŋgl] | *единственный; единый; единичный* | *жалғыз, жалғыз-ғана; біртұтас* |
| since *prep* | [s**I** ns] | *так как, поскольку (иногда с, с тех пор как)* | *-дан, -ден, -тан, -тен, -нан, -нен жалғаулары; содан бері* |
| skilled *a* | [่sk**I**ld] | *квалифицированный; опытный;*  *умелый; искусный* | *шеберлі маман; тәжірибелі; озат, шебер* |
| slavishly *adv* | [ ่slæv**I**∫l**I**] | *несамостоятельно; рабски покорно* | *құлдық* |
| slide (~ rule) *n,v* | [sla**I**d (ru**:**l)] | *слайд; кадр; скольжение; сдвигать; скользить; (логарифмическая линейка)* | *слайд; кадр; қозғалу, өту, жылжу;*  *сырғанау* |
| slow (~ down) *a* | [slou (daun)] | *медленный; (замедлять)* | *баяу, ақырын; жәй* |
| so *adv* | [sou] | *и так; таким образом* | *солай, осылай; бұлай, олай* |
| so far | [sou fα**:**] | *до настоящего времени* | *осы уақытқа дейін* |
| software *n* | [**'**sɔftwεә] | *программное обеспечение; программные средства* | *программалық жасау; программалық құрал* |
| solve *v* | [sɔlv] | *решать (задачи, проблемы)* | *(есептерді, мәселелерді) шешу* |
| solution *n* | [sә ่lu**:**∫(ә)n] | *решение; приложение; законченное решение* | *шешім; қосымша, аяқталған шешім* |
| some *pron* | [sʌm] | *некоторый; несколько; около; приблизительно* | *біраз, біршама; бірнеше; шамамен* |
| something  *pron* | [ ่s∧mθ**I** ŋ] | *что-то; кое-что; что-нибудь; нечто* | *бір нәрсе, бірдеңе; кейбір (нәрсе); бірдеме; әлдебір* |
| sophisticatе *v*  sophisticatеd *a* | [sә**'**่f**I**st**I**ke**I**t]  [sә**'**่f**I**st**I**ke**I**t(**I**d)] | *усложнять; сложный; усложнённый* | *қиындату;(күрделі; күрделендірілген)* |
| special *a* | [ ่spe∫(ә)l] | *особенный; особый; частный; специальный* | *арнаулы, ерекше; арнайы* |
| specification *n* | [ **͵**spes**I**f**I** ่kei∫(ә)n] | *спецификация; определение; описание; детализация; технические требования; техн. задание* | *ерекшелік; анықтау, анықтама; сипаттау,*  *сипаттама; техникалық талап; техникалық тапсырма* |
| specify | [ ่spes**I**fa**I**] | *задавать; уточнять; точно определять* | *тапсырма беру; дәлдеу, анықтау; өзгешіліктерін анықтау* |
| speed *n,* *v, a* | [spi**:**d] | *скорость; быстродействие; ускорять; скоростной* | *шапшаңдық, жылдамдық; тездету; тез, шапшаң* |
| spend *v* | [spend] | *проводить (время); тратить (деньги****,*** *время)* | *(уақыт) өткізу; (ақшаны) жұмсау* |
| spent | [spent] | *провёл; по-, ис-тратил* | *өткізді; жұмсады* |
| spree *n* | [spri**:**] | *веселье; резвость; кутёж; попойка;*  *What a spree! – Как весело!* | *көңіл, шат; алғырлық; шаттық; көңіл көтеріңкілік; Қалай көңілді!* |
| stage *n* | [ste**I**dʒ] | *этап; стадия; ступень; фаза; шаг; каскад; разряд; ячейка* | *кезең;саты; даму сатысы; адым; дәреже; ұяшық* |
| (to) start working | [(tә) stα**:**t ่wә**:**k**I**ŋ] | *начать работать; приступить к работе* | *жұмысты бастау; жұмысқа кірісу* |
| step *n, v* | [step] | *шаг; ступень; этап; стадия;*  *выполнять шаг* | *қадам; жүру, саты; кезең; адым жасау* |
| still *adv* | [st**I**l] | *всё ещё; до сих пор; пока ещё* | *осы кезге дейін; әлі* |
| stitch *v,n* | [st**I**t∫] | *шить; шов* | *тігу; тігіс* |
| stone *n* | [stoun] | *камень* | *тас* |
| storage *n, v* | [ ่stɔ**:**r **I**dʒ] | *запоминание; память; дисковая память; устроойство памяти; запоминать; хранить* | *жадтау; жад; дискілік жад; жад құрылғысы; сақтау; сақтау орны* |
| store *n,* *v* | [stɔ**:**] | *запоминающее устройство; память; запас; склад; вводить в память; запоминать; хранить; записывать; накапливать; помещать* | *жадтайтын құрылғы; жад; қор; жинақ; молшылық; қойма; жадқа енгізу; сақтау; жазылу, жинақтау* |
| stored *a* | [stɔ**:**d] | *соxранённый; записанный* | *сақталған; жазылып алынған* |
| (backing) store *n* | [(' bæk**I** ŋ) stɔ**:**] | *(дополнительный) накопитель; вспомогательное запоминающее устройство; вспомогательная память; внешняя память* | *(қосымша) жинақтаушы; көмекші жадтайтын құрылғы; көмекші жад; сыртқы жад* |
| street atlas *n* | [stri**:**t ( ่ætlәs)] | *карта города* | *қала картасы* |
| stupid *a* | [ **'**stu**:**p**I**d] | *глупый; тупой* | *ақымақ; ақылсыз* |
| subtract *v* | [sәb **'**trækt] | *вычитать; отнимать* | *шегеру, кеміту, алу, шығарып тастау* |
| successive *a* | [s(ә)kses**I**v] | *последовательный* | *кейінгі, келесі* |
| such as | [sʌt∫ әz] | *такой как* | *осындай; сондай* |
| sum *n,* *v* | [sʌm] | *сумма; итог;*  *складывать; суммировать* | *жалпы саны; мән; қосу;жалпы санын жинақтау* |
| (to make) sure (that) | [(tә 'me**I**k) '∫uә (ðæt)] | *(чтобы) убедиться (удостовериться), (что) ...* | *сенімді болу; күдіксіз, анық; көзі жеткен, сенген* |
| switching (off) *n* | ['sw**I**t∫**I** ŋ (of)] | *(отключение; выключение)* | *сөндіру, өшіру, алып тастау* |

***T***

|  |  |  |  |
| --- | --- | --- | --- |
| take *v* | [te**I**k] | *брать; принимать; требовать (времени, терпения)* | *алу; қабылдау; (уақытты, дәтті) талап ету* |
| take care (of) | [te**I**k ่kεә (r әv)] | *заботиться (о); ухаживать (за к.-л.)* | *қамқорлық жасау* |
| take (place) *v* | [te**I**k ple**I**s] | *брать; (происходить; иметь место)* | *алу;*  *(болу, орын алу)* |
| task *n, v* | [tα**:**sk] | *работа; индивидуальное задание; задача; программный модуль; ставить задачу* | *жұмыс ; жеке тапсырма; есеп; міндет; мақсат; программалық модуль; міндет қою* |
| taxation *n* | [tæk ่se**I**∫(ә)n] | *налогообложение* | *салық салушылық* |
| tedious *a* | [ ่ti**:**djәs] | *скушный; утомительный; изнурительный* | *жалықтыратын, қызықсыз, зеріктіретін* |
| tell *v* | [tel] | *рассказывать; говорить; сказать; высказывать; приказывать* | *айту; хабарлау; бұйыру, бұйрық беру* |
| term *n* | [tә**:**m] | *термин; срок; период; семестр; член (мат-ка)* | *термин; мерзім, мезгіл; семестр; шарт; мүше* |
| than *conj* | [ðæn] | *чем* | *-ан, -ше, -ен; қарағанда* |
| then *adv* | [ðen] | *тогда; в то время; затем; потом; к тому же; кроме того; в таком случае* | *сол уақытта, сол кезде; сонан кейін; сондықтан* |
| theory *n* | [ ่θ**I**әr**I**] | *теория* | *теория* |
| there *adv* | [ðεә] | *там; туда; (может не переводиться)* | *сонда , сол жерде* |
| therefore *conj* | [ ่ðεәfɔ**:**] | *поэтому; потому;*  *след-но; вследствие* | *сондықтан; солай болған соң; демек* |
| thing *n* | [θ **I** ŋ] | *вещь; предмет* | *зат, нәрсе* |
| thus *adv* | [ðʌs] | *таким образом; и так* | *осылай, осылайша; сондықтан* |
| time *n* | [ta**I** m] | *время; период* | *уақыт; мезгіл, дәуір, мерзім* |
| together *adv* | [tә ่geðә] | *вместе* | *бірге* |
| tool *n* | [tu**:**l] | *инструментарий; инструментальное средство; инструмент; метод; аппарат* | *аспап; аспапты құрал; әдіс; аппарат* |
| (grand) total *a* | [(grα**:**nd) ่toutәl] | *(всего); в общем* | *жаппай, барлығын қамтитын* |
| transcribe (to) *v* | [træns ่kra**I**b (tә)] | *писать в транскрипции; транскрибировать; переписывать (перевести во что-то, в другой форме)* | *транскрипциясында жазу; транскрипциялау; (басқа түрге) көшіріп жазу* |
| try *v* | [tra**I**] | *стараться; пытаться* | *тырысу, тырмысу; сынап көру; байқау* |
| turn to *v* | [tә**:**n tә] | *обратиться к ч.-л. или к.-л.* | *ықылас білдіру, назар аудару бір нәрсеге әлде біреуге* |
| (computers) themselves *pron* | [(kәm ่pju**:**tәz) ðem ่selvz] | *сами (компьютеры)* | *өздері (компьтерлер)* |

***U***

|  |  |  |  |
| --- | --- | --- | --- |
| unfortunate *a* | [ʌn ่fɔ**:**t∫(ә)n**I**t] | *к сожалению*  *( к несчастью)* | *өкінішке орай* |
| unit *n* | ['ju**:**n**I**t] | *модуль; блок; элемент; звено; компонент; устройство; единица измерения* | *модуль; блок; элемент; құрауыш;* *өлшем бірлігі* |
| unlike *adv* | [ʌn la**I**k] | *в отличие от (ч.-л., к.-л.)* | *(бір нәрседен, біреуден) өзгешелігі, ерекшелігі* |
| up-to-date *a* | [ ่ʌp tә ่de**I**t] | *современный; новейший* | *жаңа, осы күнгі* |
| (to) use *v* | [(tә) ju**:**z] | *использовать; пользоваться* | *қолдану, пайдалану* |
| 1. use | [ju**:s**] | *использование; польза* | *қолдану, пайдалану; пайда* |
| used (to) | [ ju**:s**t (tә)] | *бывало; раньше ...* | *болды; ертерек, бұрынырақ* |
| (to get) used (to) | [(tә get) ju**:s**t (tә)] | *привыкнуть (к ч.-л., к.-л.)* | *(бір нәрсеге, біреуге) дағдылану* |
| useful *a* | [ ่jusful] | *полезный; пригодный; (пригодиться; годиться predicative)* | *пайдалы; жарамды; пайдаға асу, іске жарау;* |
| usually *adv* | [ ่ju**:**ʒuәl**I**] | *обычно* | *әдетте, дағдылы* |

***V***

|  |  |  |  |
| --- | --- | --- | --- |
| value *n* | [ ่vælju**:**] | *ценность; важность; эквивалент; значение; величина; цена; стоимость; значимость* | *құн; бағалы зат; бағасы; маңыздылық; мөлшер; баға;* |
| valve *n* | [ ่vælv] | *клапан;*  *электронная лампа* | *қақпақша, бұранда; электрондық лампа* |
| various *a* | [ ่vεәr **I**әs] | *различный* | *әр-түрлі; жан-жақты* |
| vary *v* | [ ่vεәr **I**] | *меняться; изменяться* | *өзгеру, өзгерту* |
| vast *n, a* | [vα**:**st] | *простор; обширный; безбрежный; огромный; безграничный; многочисленный* | *кеңдік; зор, өте үлкен; көп, сансыз, шексіз* |
| vehicle *n* | [ ่vi**: I**kl] | *ср-во передвижения; (летательный) аппарат; ср-во распространения, выражения; проводник (света, звука)* | *жылжыту құралы; ұшу аппараты; (жарықты, дыбысты) өткізгіш* |
| versatility *n* | [ **͵**vә**:**sә ่t**I**l**I**t**I**] | *многосторонность; изменчивость* | *көп жақтылық; құбылмалылық, айнымалылық* |
| very *adv* | [ ่ver **I**] | *очень* | *дәл, нағыз; ең; өте* |
| voltage *n* | [ ่voult**I**dʒ] | *напряжение тока* | *электр тоғының күші* |

***W***

|  |  |  |  |
| --- | --- | --- | --- |
| wall *n* | [wɔ**:**l] | *стена* | *қабырға* |
| watch *v*  (by) watch(ing) | [ ่wɔt∫]  [ ba**I** ่wɔt∫**I**ŋ] | *смотреть; наблюдать; следить;*  *посредством наблюдения* | *қарау, бақылау, байқау* |
| way *n* | [we**I**] | *способ; средство; метод; образ действия; магистраль* | *әдіс, тәсіл; амал; бейне әрекеті, қашықтық; бағыт* |
| whatever *pron* | [wɔt evә] | *какой-бы ни; любой; всё что; что бы ни; никакой;* | *қандай болмасын; қалай да, қандай болса да; барлығы, бүкіл; ешқандай, ешбір* |
| when *conj* | [wen] | *когда* | *қашан* |
| whether *conj* | [ ่weðә] | *ли* | *ба, -бе, -па, -пе,- ма,-ме* |
| **which *conj, pron*** | [w**I**t∫] | *который; какой; что* | *қайсы, қай, қандай, қалай* |
| (as) well *adv* | [(әz) wel] | *хорошо; (также; тоже в конце предложения )* | *жақсы (дәл осылай, ол да)* |
| wonder *v* | [ wʌndә] | *интересоваться;*  *хотеть знать; удивляться* | *қызығу, көңіл қою; хабардар болу; таңғалу, ғажаптану* |
| word *n* | [wә**:**d] | *слово* | *сөз* |
| work (on) *v* | [wә**:**k] | *работать (над)* | *жұмыс істеу; еңбек ету, орындау ( -ға,-ге, қа,-ке, -дан, -ден, -тан, -тен)* |
| work (out) *v* | [ ่wә**:**k (aut)] | *выражаться; решать; работать; (разрабатывать); эксплуатировать* | *көрсету; білдіру; шешу; жұмыс істеу, еңбек ету; (өңдеу); пайдалану; аяғына дейін апару* |
| written *a* | [ ่r **I** t n] | *написанный* | *жазылған* |

***Y***

|  |  |  |  |
| --- | --- | --- | --- |
| yet *adv* | [jet] | *ещё (в отриц. предл.); уже (в вопрос. предл.)* | *(болымсыз сөйлемде) әлі, тағы да;*  *(сұраулы сөйлемде) әлдеқашан* |

  
Aaron Staley, Computer Engineering Major

Read more about Aaron and other computing majors on our [Faces of Computing](http://computingcareers.acm.org/?page_id=61) page.

[Excerpt from Bureau of Labor Statistics site  
**Occupational Outlook Handbook**](http://www.bls.gov/oco/ocos027.htm):

[About ACM](http://www.acm.org/about_acm/) | [Contact Us](http://www.acm.org/key_people/) | [Membership](http://www.acm.org/membership/) | [Privacy Policy](http://www.acm.org/serving/acm-privacy.html) | [Code of Ethics](http://www.acm.org/serving/ethics.html) | [© 2006 ACM, Inc.](http://www.acm.org)  
Powered by [WordPress](http://wordpress.org).

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Jeff's pictureJeff Offutt  Professor of [Software Engineering](http://www.cs.gmu.edu/programs/masters/swe/)  Mailstop 4A5  [Volgenau School of Info Tech & Engineering](http://ite.gmu.edu/)  [George Mason University](http://www.gmu.edu/)  Fairfax, VA 22030-4444  Office:    4430 [Engineering Building](http://eagle.gmu.edu/map/fairfax.php?building=engineering) Spring 2011 office hours: Wednesdays 2:30-4:00 Email:   Jeff's email, no screen scraping please Phone:   (1) 703-993-1654  Fax:       (1) 703-993-1710  URL:     http://cs.gmu.edu/~offutt/  I can't work my best unless I live my best.  Students who want to email me should read my suggestions [hints for students making requests](http://cs.gmu.edu/%7Eoffutt/for-students.html) first.  My thoughts on [cheating and plagiarism](http://cs.gmu.edu/%7Eoffutt/classes/advice/cheatingphilosophy.html).  [COS bio](http://expertise.cos.com/cgi-bin/exp.cgi?id=510814)       [Google scholar](http://scholar.google.com/scholar?hl=en&lr=&q=a+j+offutt&btnG=Search)  Last Modified: June 2009 | Why study Software Engineering? Because the best job in America in 2011 is being a [software engineer](http://www.braintrack.com/college-and-work-news/articles/software-engineer-ranked-best-job-for-2011-11010502) | Some people try to predict the future, but scientists create it.      [previous quotes](http://cs.gmu.edu/%7Eoffutt/prevquotes.html)  Some of my [picture collections](http://cs.gmu.edu/%7Eoffutt/pictures/).  Ruminations on [9-11](http://cs.gmu.edu/%7Eoffutt/twoDates-911.html).  [Countries](http://www.travbuddy.com/widget_map_display.php?id=4578329) I've visited.   |  |  | | --- | --- | | [book cover](http://cs.gmu.edu/~offutt/softwaretest) Published by [Cambridge University Press](http://www.cambridge.org/catalogue/catalogue.asp?isbn=0521880386), January 2008.  Order from [Amazon](http://www.amazon.com/Introduction-Software-Testing-Paul-Ammann/dp/0521880386/ref=pd_nr_b_4?ie=UTF8&s=books) | [STVR](http://onlinelibrary.wiley.com/journal/10.1002/(ISSN)1099-1689) [Editor's page](http://cs.gmu.edu/%7Eoffutt/stvr/)  Wiley's Software Testing, Verification and Reliability  Useful research in building better software. |   Announcements & Events   |  | | --- | |  | | [Interview in LogiGear's magazine](http://www.logigear.com/825-LogiGear-Magazine/826-Issue-August---2010.html) | |  | | keynote at [Google's Test Automation Conference (GTAC)](http://www.gtac.biz/): "Automatically Generating Test Data for Web Applications," October 2010, Hyderabad India. [Slides](http://cs.gmu.edu/%7Eoffutt/JeffOffutt-GTAC-2010.pptx) [YouTube](http://www.youtube.com/watch?v=lmS5ElMyIHU) | | A [Mutation testing community](http://www.mutationtest.net/) website | | [GMU Software Engineering seminars](http://cs.gmu.edu/~smalek/seminar.html) | |  | |