ҚАЗАҚСТАН РЕСПУБЛИКАСЫ

БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

Т. ХӘКІМОВА

АҚПАРАТТЫҚ ТЕХНОЛОГИЯЛАРДЫ ОҚЫТУДЫҢ ЛОГИКАЛЫҚ БАҒДАРЛАМАЛАУ НЕГІЗДЕРІ

Оқу құралы

Алматы
2013

ÓÄÊ 372.851.02
ÓÄÊ 372.800.4.02

Пікір жазғандар:

физика-математика ғылымдарының кандидаты, доцент Б.А.Урмашев
физика-математика ғылымдарының кандидаты, доцент М.А.Бекпатшаев
физика-математика ғылымдарының докторы, профессор Н.Заурбеков

Хәкімова Т.

Ақпараттық технологияларды оқытуда логикалық программалау негіздері: оқу құралы. Алматы: Қазақ университеті, 2013. – 80 бет.

Оқу-әдістемелік құрал студенттердің жаңа технологияны пайдалана отырып, қазіргі кездегі ЭЕМ-де жоғарғы деңгейде тәжірибе алып, жұмыс істеу қабілетін арттыру мүмкіндігін тудырады және мәліметтерді өңдеудің түйінді мәселелерін өздігінен шешуге көмектеседі. Оқу құралында жасанды интеллектінің негізгі ұғымдары мен даму тенденциялары,Visual Prolog декларативті программалау тілінің зертханалық жұмыстары қарастырылған.

Оқу құралы IBM PS компьютерімен жұмыс істеуді өз бетімен оқып-үйренушілер мен ақпараттық технологиялар пәнінен мемлекетаралық бақылауға даярланатын студенттер үшін де тиімді.

МАЗМҰНЫ

Кіріспе

Соңғы жылдары компьютерлік телекоммуникациялық техниканың және технологияның рөлі мен орны түбегейлі өзгерді. Ақпараттық және телекоммуникациялық технологияларды игеру бүгіндер әркім үшін аса қажет. Қазіргі технологияның дамуы мен оның қолданылуының деңгейі материалдық базасының дамуымен ғана емес, оның жаңа білімді туындату, игеру және қолдана білу қабілетімен де анықталады.

Кейінгі кездері оқыту үдерісіне компьютер, электронды оқулық, мультимедиалық, ғаламтор сияқты жаңа ақпараттық коммуникациялық технологиялардың дидактикасы мен технологиясы кеңінен қолданылып келе жатқаны мәлім. Оған Э.В. Еврейнов, В.А. Каймин, В.В. Гриншкун, О. Околов, Л.Х. Зайнутдинова, П.С. Булкин, Б.И. Волков, Е.Ы. Бидайбеков, Ж.А. Қараев т.б ғалымдар мен әдіскерлердің зерттеулері айғақ бала алады.

Бүгінгі таңда ақпараттық коммуникациялық технологияларды оқу үдерісінде қолдану әлемдік ақпараттық-коммуникациялық білім беру кеңістігіне қосылуды қамтамасыз етеді.

Логикалық программаның идеялық тамыры – математикалық логикада, формуланы және формалді анықтау әдісін қолдана отырып, автоматты түрде нәтиже алуға және есепті формалды сипаттау әдісін ашуға септігін тигізуінде.

Есептің шартынан және құрылымынан есептің шешімін алатын программалық тілді «декларативті тіл» деп атайды. Мұндай тіл процедуралы тіл деп аталады. Процедуралық тілде есептің шешімі алгоритім құруға және жетілдіруге жұмсалады.

Марсель-Экс университетінде Алон Колмероэ және оның тобы теореманы дәлелдейтін Фортранда жасалған программа құрды. Бұл программа қарапайым тілде мәтіндік ақпаратты өңдейтін жүйе. Программа **Пролог** (Programmation en Logiquc) деп аталып, Ковальскидің интерпретаторын қолданды.

Қазіргі кезде Пролог тілін кез келген компьютерге қоюға болады. Пролог – логикалық программалау тілі. Кең таралған түрлері: Arpity-Prolog, МПролог, Turbo Prolog, Visual Prolog. Пролог тілінің нұсқаларында айырмашылық болғандықтан оқу жүйесінде аз қолданылады.

Оқулықтың зерттеу нысаны:

Жұмыстың негізгі нәтижесі – логикалық программа негізінде предикаттарды есептеу теоремасының түбегейлілігін, белгілеулердің толық бір жүйе болатындығын дәлелдеу. Осы алынған нәтиже предикаттарды есептеу тілдік құралы арқылы формалды түрде, осы тілде жазылған пікірлердің барлық итерпретациялық жағдайында ақиқат. Бұл дәлелдеуден кейін пікір жүйелі түрде қалыптасады.

ЭЕМ үшін есепке программа құрудың негізгі қиындығы машина мен адам тілінің айырмашылығында емес, олардың әртүрлі ойлауында.

Логикалық программалаудың программалау тілдерінің негізгі даму жолынан бірншама айырмашылығы бар. Логикалық программалау пікірлерге негізделіп, компьютердің операциялық терминдеріне адамды ойлауға үйретпей, компьютерге адамға тән нұсқаларды орындатады.

Оқулықтың зерттеу пәні:

Visual Prolog тілінің негізгі қолдану ортасы:

- Қолданбалы программаның жылдам көшірмесін алу;
- Сараптау жүйелері және жасанды зерде ортасындағы зерттеулер;
- ЭЕМ-мен табиғи тілде қатынасу (табиғи-тілдік интерфейс);
- Мәліметтер қоры;
- Робот қимылдарының жоспарын және күнтізбе құру;
- Компиляторлар, ассемблер, диассемблер, программа конверторын жазу;
- Автоматтандырылған жобалау жүйелері (САПР).

Оқулықтың мақсаты:

Жасанды интеллект, сараптаушы жүйелер, логикалық программалау жүйесімен танысу және адамның компьютермен сұхбатын ұйымдастыру үшін құрылған декларативті программалау тілдерінің бірі Visual Prolog тілін оқып үйрену болып табылады.

Оқулықтың міндеттері:

- Visual Prolog декларативті программалау тілінің ортасында жұмыс істеудің ерекшелігін анықтау;
- Студенттерге жасанды интеллект пәннің оқу-әдістемелік кешенін дайындауды жетілдіру;
- Visual Prolog тіліне зертханалық жұмыстар жүргізу;
- Осы программа негізінде тест сұрақтарын құрастыру;

Оқулықтың ғылыми болжамы:

Visual Prolog декларативті программалау тілін құруды және үйретуді кеңінен қолға алынса, ойлау қабілетін жетілдіреді және құрылымдалмаған программа жазуға мүмкіндік береді. Дәстүрлі тілде жазылған программалар, алгоритмнің жазылуына қарай мәліметтерді белгілі ретпен өңдейді.

Программалауды үйрену үшін программалау керек және оны жаза білу керек. Ол үшін міндетті түрде программалау тілі және өңдеу ортасын меңгеру керек.

1. ЖАСАНДЫ ИНТЕЛЛЕКТІНІҢ НЕГІЗГІ ҰҒЫМДАРЫ ЖӘНЕ VISUAL PROLOG ДЕКЛАРАТИВТІ ПРОГРАММАЛАУ ТІЛІНІҢ ОРТАСЫ

1.1. Жасанды интеллектінің негізгі ұғымдары мен даму тенденциялары

Қазіргі кезде программистің жұмыс істеу сапасының дәрежесі интеллектуалдық жүктеудің көп бөлігін компьютерлер орындағанда ғана жоғары болады. Бұл аймақта максималды прогреске қол жеткізу үшін

“Жасанды интеллект” әдісі қолданылады. Мұнда компьютер бір типті және қайта-қайта жасала беретін операцияларды ғана орындамайды, сонымен қатар өзі де үйренеді. Бұған қоса толық қанағаттандыратын “жасанды интеллектіні” құру адамзат дамуының жаңа деңгейлерін ашады.

Адамдарды өздерінің ойлау механизмі әрқашан қызықтырған. Адамды ақыл иесі ретінде ерекшелендіріп тұратын олардың интеллектісінің болуы. Адам интеллектісі көптеген компоненттерде тұрады, оның ішінде сыртқы ортамен байланысты болатын сезім мүшелері, үйренуге икемділігі, бағалауға келмейтін білімдер жиыны. Есептерді шешу барысында байқалатын интеллектінің өзіндік белгілері болып үйренуге икемділік, жалпылау, тәжірибе (білім) жинақтау және өзгерістерге бейімделу болып табылады. Интеллектінің осы қасиеттерінің арқасында ми әртүрлі есептерді шеше алады, сонымен қатар бір есептен екіншісін шешуге оп-оңай ауыса алды. Осылайша интеллектісі бар ми алуан түрлі, оның ішінде формалданбаған, стандартты, алдын-ала шешу әдістері жоқ есептерді шығаруға арналған әмбебап құрал болып табылады.

Интеллектісі бар ми интеллектуалды есептерді шешуге бағытталған болса, бұл процесті ойлау немесе интеллектуалды іс-әрекет деп атаймыз. Интеллект және ойлау органикалық түрде теоремаларды дәлелдеу, логикалық талдау, жағдайларды ажырату, іс-әрекетті, ойынды болжау және белгісіздік жағдайында басқару сияқты тапсырмалармен байланысты.

Осылайша, интеллект деп мидың қабылдау, еске сақтау және бағытталған түрде білімді оқу барысында түрлендіруді пайдалана отырып тәжірибе және түрлі жағдайларға байланысты бейімдеу негізінде интеллектуалды есептерді шешу мүмкіндігін атаймыз.

Интеллект (*intelligence*) терминінің өзі лат. *intellectus* – білім, ойлау, адамның ойлау мүмкіндігі деген сөздерінен шыққан.

40-жылдары есептеуіш машиналардың және кибернетикадағы зерттеулер пайда болысымен адамның ойлау табиғаты туралы сұрақ кибернетикалық аспектіге ие болды. Ғалымдар адамның бойында интеллектуалды іс-әрекеттерді машинада құруға бар күштерін салды. Бұл зерттеу бағыты «жасанды интеллект» деген атауға ие болды.

Жасанды интеллект (*artificial intelligence*) – ЖИ (*AI*) автоматты жүйелердің адам интеллектісінің бөлек бір функцияларын атқаруын айтады. Мысалы, ертерек алынған тәжірибе және сыртқы әсерлерді рационалды талдау негізінде тиімді шешімдерді таңдау және қабылдау.

ЖИ өзінің пайда болуы және дамуы жағынан есептеуіш машиналарға тәуелді. Әдетте бұл бағытты информатика және есептеуіш техника салаларына жатқызады. Бұның бәрі Екінші Дүниежүзілік соғыс аяқталған соң барлық ойын есептерді және жұмбақтарды компьютер көмегімен шешуден басталды. Осы алғашқы тәжірибелер негізінде туған іргелі идея күйлер кеңістігінде іздеу деген атқа ие.

Сонымен, интеллект деп мидың қабылдау, еске сақтау және бағытталған түрде білімді оқу барысында түрлендіруді пайдалана отырып

тәжірибе және түрлі жағдайларға байланысты бейімдеу негізінде интеллектуалды есептерді шешу мүмкіндігін атайтын боламыз.

Бұл анықтамада «білім» деп миға сезім мүшелері арқылы түсетін ақпаратты ғана атамаймыз. Мұндай типті білім, әрине, өте маңызды, алайда интеллектуалды іс-әрекет үшін жеткіліксіз. Қоршаған орта объектілері сезім мүшелеріне тек әсер етіп қана қоймайды, сонымен өзара белгілі бір қатынастарда болады. Қоршаған ортада интеллектуалды әс-әрекетті іске асыру үшін білім жүйесінде сол ортаның моделіне ие болу керек. Қоршаған ортаның бұл ақпараттық моделінде реалды объектілер орналасқан, олардың қасиеттері және қатынастары тек көрсетіліп және есте сақталып қоймайды, сонымен қатар ойша «бағытталған түрде түрлендіріледі». Бұнымен қоса, маңыздысы – сыртқы орта моделін құру «тәжірибе және түрлі жағдайларға бейімделу негізіндегі үйрену» арқылы іске асады.

Есептердің белгілі кластарын шешуге байланысты интеллект және ойлау

Интеллектуалдың есепті шешу барысында байқалатын өзіндік белгілері – оқуға, жалпылауға, тәжірибе жинауға және есепті шешу барысында өзгерістерге бейімделуге дағдылану. Интеллектінің осы қасиеттерінің арқасында ми түрлі есептерді шеше алады, сонымен қатар бір есептің шешуінен екіншісіне оңай ауысады. Осылай интеллектісі бар ми көптеген алдын-ала шығарылу әдістері, стандартты шығарылуы жоқ есептерді шеше алатын әмбебап құрал.

Бұл ұғымның басқа да анықтамалары бар. Ғылым, әдебиет, мәдениет мәселелерін талқылауға болатын кез келген материалды жүйе интеллектіге ие. Тьюринг былай түсіндірген: әртүрлі бөлмелерде машина және адамдар бар. Олар бір-бірін көрмейді, бірақ ақпарат алмаса алады (мысалы, электронды пошта арқылы). Егер диалог барысында адамдар машиналармен сөйлесіп отырғандарын байқамаса, онда машинаны интеллектіге ие деп айтуға болады.

Тьюринг берген ойлауды имитациялау жобасы қызығушылық тудырады: «Ересек адам интеллектісін имитациялауға талпынып, біз адам миы қазіргі күйге қалай жеткені туралы көп ойлануымыз керек... Неге бізге ересек адам интеллектісін имитациялайтын программа жазғанша, кішкене бала интеллектісін имитациялайтын программа жазбасқа? Егер баланың интеллектісі дұрыс тәрбие алатын болса, ол ересек адамның интеллектісі болмай ма? Біздің есептеу бойынша оған сәйкес келетін құрылғы оңай программаланып қойыла алады... Осылай, біз мәселені екі бөлікке бөлеміз: «бала-программа» және осы программаны «тәрбиелеу» программасы».

.

Жасанды интеллект жүйесін әр қырынан зерттеу

ЖИ (**Жасанды интеллект**) әр қырынан зерттеу тарихи түрде қалыптасты, олар бір бірінен тәуелсіз түрде дамыды, тек соңғы кезде ғана олардың жақындасуына жол ашылды:

- құрылымдық;
- имитациалық;
- логикалық;
- эволюциялық.

ЖИ (**Жасанды интеллект**) жүйелеріндегі барлық нейрондық зерттеулер спектрі құрылымдық деп аталды. Құрылымдық дедегенімізп ЖИ-ды адам миының құрылымын модельдеу арқылы құру. Ол бейнелерді ажырату есептерінде көп қолданылады.

Адам миы негізінде құрылған модельдер үшін айқындылық қасиеті тән емес. Бұл желілерді адам миымен жақындастыратын тағы бір қасиеті – нейронды желілер қоршаған орта туралы толық мәліметсіз болса да жұмыс істей береді, яғни адам тәрізді, қойылған сұрақтарға тек «иә», «жоқтан» басқа, «нақты білмеймін, бірақ иә сияқты» деген жауаптар бере алады.

Келесі зерттеу жанды мидың құрылымдық және функционалды ерекшеліктерін имитациялық модельдеумен байланысты.

Бұл зерттеулерді тағы «қара жәшік» немесе «нәтижесі бойынша сәйкес келу» деп атайды. Оның мәні келесіде: зерттеуші интеллектінің құрылу және жұмыс істеу принциптерін білмейді, яғни оны «қара жәшік» ретінде қарастырады. Бұл оқулықтың негізгі мақсаты адам интеллектісінің жұмысын ақырғы нәтиже бойынша имитациялайтын кейбір эвристикалық компьютерлік программаларды құру болып табылады. Мұнда ададмдар қандай әдістерді қолданатыны ескерілмейді. ЖИ жүйелерін құрудың мұндай түрі имитациялық деп аталады. Кибернетика үшін классикаалық зерттеу болып табылады.

Осылайша, мұнда адамның басқа қасиеті модельденеді – басқалар не істейді, оның не үшін керектігіне назар аудармай көшіру. Көп жағдайда бұл мүмкіндік біраз уақытты үнемдейді.

Эвристикалық программалау Карнеги университетінің А. Ньюэлл және Г. Саймон есімдерімен байланысты және келесі принципке негізделген, адам миы нәтиже бойынша символдарды басқару туралы қарапайым есептер жиынына, яғни компьютер орындай алатын операциялар келуі мүмкін. Есептердің шешімі мүмкін болатын шешімдер жиыны кеңістігінен эвристикалық ережелер бойынша іздестіріледі. Олар іздестіруді, белгілі бір бағыт бойынша жүруді тездетеді. Эвристикалық іздестіру көлемінде шығарылған типтік есептерге теоремаларды дәлелдеу, түрлі ойындар, жұмбақтарды шешу, геометриялық және шахматтық есептер, әуендерді құру, химиялық құрылымдарды анықтау, т.б. жатады.

Саймон компьютерлер 1990 жылдан соң әлем чемпионы болады деген болжам жасаған болатын, иә ол толығымен орындалды. Эвристикалық іздестіру шеңберінде машиналар тек қарапайым шектеулі есептерді шеше алатын.

ЖИ (**Жасанды интеллект**) имитациялық программаларының келесі жетістіктері, атап айтсақ, *Deep Blue* шахматтық компьютерінің 1997 жылы әлем чемпионы Г. Каспаровты жеңуі, тек эвристикалық іздестірумен ғана

емес, ЖИ-дың басқа синтетикалық салаларының пайда болуымен де байланысты. Оларға мықты көппроцессорлы паралелльді жүйелерге және нейронды акселераторларға негізделген эвристикалық программаларды қолдайтын аппаратты жабдықтау жатады. Мысалы, аталған компьютерде жүрістер генераторы 256 параллельді процессорлар негізінде жүзеге асырылған.

Имитациялық зерттеудің негізгі кемшілігі көптеген модельдердің төмен ақпараттық мүмкіндігі.

Келесі зерттеу логикалық деген атқа ие. Ол неге пайда болды? Адам тек логикалық ойлаумен ғана айналыспайды емес пе? Бұл дұрыс, бірақ адамды жануардан тек оның ойлау қабілеті ғана ерекшелендіреді.

Логикалық зерттеудің негізі болып булеандық алгебра есептелінеді. Әр программист олармен if операторын біле бастағанымен таныс. Өзінің келесі дамуын булеандық алгебра предикаттар есептеуінің негізінде жалғастырды, онда ол символдарды, олардың арасындағы қатынастарды қолдану арқылы кеңейтілді. Кез келген ЖИ жүйесі логикалық принципке негізделген деуге болады және теоремаларды дәлелдейтін машина ретінде қарастырылады. Сонымен қатар мәліметтер деректер қорында аксиомалар, логикалық нәтижелер ережелері түрінде сақталады. Бұған қоса, әр осындай машина мақсатты генерациялау блогына ие, ал жүйе берілген мақсатты теорема ретінде дәлелдеуге тырысады. Егер мақсат дәлелденген болса,онда қолданылған ережелер трассировкасы қойылған мақсатқа жеткізетін әрекеттер тізбегін алуға көмектеседі. Мұндай жүйенің қуаты мақсат генераторының және теоремаларды дәлелдеу машинасының мүмкіндіктерімен анықталады.

ЖИ-ды толық көрсету үшін алгебраның мүмкіндіктері жеткіліксіз, осында ЭЕМ-дердің негізі бит-0 және1 мәндерін қабылдайтын жады ұяшығы екенін еске түсірейік. Осылайша, компьютерде жасауға болатынның бәрін предикаттар логгикасында жүзеге асыруға болады деген болжам жасай аламыз.

Логикалық зерттеу нақтырақ болуы үшін нақты емес логика көмектеседі. Оның негізгі ерекшелігі «иә», «жоқтан» (1/0) басқа «білмеймін» (0,5) сияқты аралық мәндерді қабылдауға болады. Бұл зерттеу адам ойлауына көбірек ұқсайды, себебі иә, жоққа қарағанда білмеймін деген жауап жиі қолданылады. Көптеген логикалық әдістер үшін зор еңбек керек, дәлелдеуді іздестіру кезінде нұсқалар бәрі толық қарастырылады. Сондықтан бұл зерттеу есептеу процесінің эффекттиімдіивті жүзеге асырылуын қажет етеді, және жұмыс сапасының жоғары болуына деректер қорының көлемі үлкен болмаса кепіл беріледі.

Эволюциялық жағынан зерттеу үлкен қарқын алды. Бұл зерттеу бойынша ЖИ жүйелерін құрғанда, бастапқы моделді құруға және қандай ережелер бойынша өзгеретініне баса назар аударылады. Модель әртүрлі әдістер арқылы құрылуы мүмкін.

Эволюциялық модельдер жоқ деп те айтуға болады, тек эволюциялық алгориттмдер ғана бар, бірақ эволюциялық зерттеулер кезінде алынған

модельдер өзіне ғана тән ерекшеліктерге ие, бұл оларды басқа класқа бөлуге мүмкіндік береді.

Барлық аталған зерттеулер бір-бірінен тәуелсіз дамыған, тек соңғы кезде ғана олардың жақындасуына жол ашылғандай. Аралас жүйелер жиі кездеседі, мұнда жұмыстың бір бөлігі бір тип бойынша, екіншісі басқа тип бойынша орындалады.

1.2. Жасанды интеллектіні зерттеудің негізгі бағыттары

Эксперттік жүйелердің жасалуы (ЭЖ) жасанды интеллект мамандарына классикалық дәстүрлі жұмыс болып есептеледі. ANSI C стандартында жазылған *C-PRS* интеллекттік шешкішті *(Procedural Reasoning System in C)* NASA, авиаөнеркәсіп және мобильді роботтар қолданады.

Робототехника

Автономды үй құрылғыларды құру кезінде кем емес кедергілер әскери және ғарыш роботтарын жасаудан кем емес кедергілер болады. Максималды түрде қауіпсіздіктің сұранысына байланысты, өңдеушілерге бұл жай қатты кедергі болады. Шаңсорғыш автономды үй машиналар рыногы даму үстінде. Құрылғылар неше түрлі навигациялық жүйемен және барынша түрлі перефириялық датчиктермен қамтылған. Робот-шаңсорғыштар үй ішінде кез келген траекториямен қозғалып, қоқсықтарды жинай отыра, статикалық заттарға немесе жанды заттарға жақындағана олар қашады. Ақылды шаңсорғыштар өздерінің тұратын орнына қайтып бара алады.

Басқа перспективалы рынок – автономиялық гүлзар шабу (көгалшапқыш). Мысалы, Electrolux фирмасы күн батареясынан зарядталып, тәулік бойы жұмыс істей алатын шапқыш машиналарын шығарады. Интеллектуалды машиналар иелеріне сусындар мен аяқ киімдерін апарумен қоса, түрлі басқа функцияларды орындайды. Probotics фирмасының *Cye* деген роботы компьютерге қосылып, компьютерге орнатылған арнайы программа арқылы қашықтықтан басқарылады. Ыңғайлы виртуалды инструмент арқылы қолданушы үй жоспары бойынша *Cye*-ға пәтер ішіндегі керекті траектория маршрутын белгілеп қоя алады. Роботпен байланыс хаттама бойынша жүзеге асады. Ол хаттамаға 35 команда және роботтың 20 жауап қайтаруы енгізілген. Болашақта *Cye* роботы тек пәтер ішінде ғана емес, аулада да жүре алады.

Cog роботының басқару жүйесі бір жүйе. Көптеген *Cog* түйіндерінде *Motorola* 68 332 16 МГц процессорлары орнатылған. Ол процессорларда *L* (версия *Common Lisp)* интерпретаторы орындалады. Каролина университеті адамдарды түрлі апаттардың салдарынан болған түрлі қоқсықтардың астынан шығарып алатын роботтарды жасау үстінде.

NASA кішкене доп көлеміндегі робот жасап шығарды. Ол робот дауыс командаларын түсінеді, әрі камерамен, температура тетігімен қамтылған.
Огайо штатындағы мемлекеттік университеттің медициналық орталығы хирург-роботты жасап шығарды. Ол робот камерамен және екі қолмен қамтылған. Адам роботты компьютер арқылы басқарады.

Автономды агенттер

Автономды агент технологиясының басты бір артықшылығы дұрыс шешімін нақты білмейтін өндірушіге агент прототипін құрып-ақ мәселені оңай шешуге болады. Ол кейіннен компьютер ортасына жүктеледі. *Microsoft Agent* технологиясы бойынша жұмыс жүріп жатыр. Ол Windows интерактивты персонаждарына кіреді. Онымен араласуға және кеңес сұрауға болады.

Кейбіреулердің ойы бойынша агент Internet қолданушысының орнына барін жасау керек. Қолданушы оған тек керекті файлды немесе ақпаратты жіберуі керек. Ол сол ақпаратты өзі дайын әкелуі керек.

Ми ұқсас-сандық құрылғы

Нероинформатика институты мен Манчестер технологиялық институтының швеция және америка ғалымдары кәгімгі адам миының функцияларын орындайтын техналогия құрды. Ол бір уақытта сандық және аналогтық информацияны қабылдайды. Бұл жаңа технология мықты компьютерлердің шығуына себепкер болды.

Жасанды өмір

Кибернетикалық құрылғыларды жасау мәселесі- мүмкіндігінше электронды немесе тірі ағзаға қарап, оның функцияларын орындайтын технологияны жасап шығару көптеген өндірушілердің назарын өзіне аудартады.

DARPA қаржыландыратын жобалардың бірі – Лего кубиктарын жинайтын жүйе. Ол бейнекамера, манипулятор және компьютерден тұрады.

Microsoft-тың басқа бір жобасы – *Microsoft Ball* тұлғаның эмоциялық жағдайын моделдеуіне арналған. Қолданушымен араласа отырып, оның эмоционалды жағдайын байқау керек. Көптеген эксперименттерге, қолданушылардың бұл программадан алған әсерлеріне қарай, қолданушының разылығы байқалады.

Чат-роботы

Барлық қолданушылар Generic Artificial Consciousness (GAC) жасанды еспен араласып, оған иә немесе жоқ жауабын беретін сұрақтарды қоюға болады. GAC-тың құрушысы, компьютерлік фанат Крис Мак-Кинли 12 жасында микрокомпьютерге шахмат TRS-80 программасын жасап шығарды.

Жасанды интеллект жүйелерінің программалық қамтамасыз етілуі

Интелектті мәселелерді шешу үшін арнайы тілдер жасалып жатыр. ОЛ тілдерге LISP, PROLOG, SMAL TALK және басқалар жатады.

1.2.1. Дәстүрлі бағыттар

- Нақты емес логика;
- Бейнелеулерді өңдеуі ;
- Эксперттік жүйе ;
- Оптималды комбинаторлы проблеманың шешімін табатын интеллектуалды қосымша;
- Қазіргі кездегі ОЖ;
- Әскери технологиялар.

Шығарылатын есептердің формалды еместігін және эвристикалық, қолданылатын білімнің өзіндік мінезін ескере отырып, қолданушы яғни эксперт эксперттік жүйемен қолма-қол диалогтық түрде байланысуы керек.

ЭЖ-нің қорының негізгі күші білім болғандықтан, ЭЖ білімді қабылдап алу қасиетіне ие болуы керек. Білім алу үдерісін келесі түрлерге бөлуге болады:

1) білімді эксперттен алу;

2) жүйенің нәтижелі жұмыс істеуін қамтамасыз ететіндей, білімді ұйымдастыру;

3) білімді түсінікті жүйеге түрде көрсету.

Білімді алу процесі былайша айтқанда «білім инженерінің» (knowledge engineer), яғни күрделі есеп шығаратын, эксперт жұмысының анализі негізінде жүзеге асады.

ЖИ жүйесінде және эксперттік жүйелерде көп жағдайда формалды емес есептер шығарылады, яғни ЭЖ және ЖИ формалды есеп шешуге арналған програмалардың құрылуын өзгертпейді және шек қоймайды. Ньюэллге [1969] және Саймонға қарап [1973], формалды еместерге (ill-structured) келесі мінездемелердің біреуіне немесе бірнешеуіне ие болатындай келесі есептерді қарастырамыз:

1) есептер сандық түрде берілмеуі керек;

2) мақсаттық функцияда анықталғандай мақсаттар терминмен берілмеуі тиіс;

3) алгоритімдік емес шешімі жоқ;

4) алгоритімдік емес шешімі бар, бірақ оны ресурстардың шектелуіне байланысты қолдануға болмайды (уақыт, жады).

Формалды емес есептер келесі ерекшеліктерге ие:

1) қателік, бірмәнділік емес, толық емес және нәтиженің қарама қайшылығында;

2) қателік, бірмәнділік емес, проблемалық аймақ пен шығарылып жатқан есеп туралы толық емес және қарама-қайшы білім;

3) нәтиже іздеу кезінде іздеу аймағынан асып кетуі;

4) динамикалық түрде өзгеретін мәліметтер мен білім. Бір айта кетерлігі, формалды емес есептер өте үлкен және керекті класс болып табылады [Дородницын, 1985].

Эксперттік жүйелер мен жасанды интеллект мәліметтерді өңдеу жүйесінен айырмашылығы – оларда символдықтүрде ұсыну, символдық шығару және эвристикалық нәтиже іздеу қолданылады.

ЭЖ-дің қосымшалар спецификасы басқа жасанды интеллект жүйелерге қарағанда айырмашылығында, біріншіден, экспертік жүйелер тек қана қиын есептер шығаруға қолданылады; екіншіден, эксперттік жүйелер нәтижесінің сапасы және тиімділігі жағынан эксперт адамнан кем емес; үшіншіден, экспертік жүйелердің шешімі қолданушыға түсінікті түрде және деңгейде түсіндіріледі. Эксперттік жүйелердің бұл қабілетті өзінің білімі мен шешімдері туралы ой пайымдайтын мүмкіндік береді.Төртінішіден, эксперттік жүйелер өзінің білім қорын экспертпен диалог кезінде толықтыра алады. Бесіншіден, эксперттік жүйелер шешу үшін қолданылатын есептер ортасы шектелген:

символдар немесе сигналдардың интерпретациясы, диагностика, істерді жобалау, берілген шектер бойынша обьектілердің конфигурациясын құрастыру, жөндеу, инструктаж, жүйелердің іс-әрекетін басқару (интерпретация,алдын ала жобалау, түзету, басқару). Экспертік жүйелер әртүрлі проблемалық аймақтарда қолданылады, мысалға медицина, есептеу техникасы, программалау, генетика, акустика, спектралды анализ, геология, юриспруденция және т.б.

Эксперттік жүйенің практикалық нәтижесі, зерттеу аймағының үлкен жетістіктерге жеткеннін көрсетеді. Бірақ бұл аймақтың ғылыми қоры толығымен жетілмеген және дамудың бастапқы деңгейде тұрғандығын атап айтқан жөн. Әлі күнге дейін базалық принциптердің бар болуы, жаңа қолданбаның құрылуы үлкен еңбекті керек етеді (бірнеше жыл) және әр кезде жақсы жеміс бермейді. Олай болса да, бір қолданбадан екінші қолданбаға берілетін әдістер мен құралдар бар.

Қандай программа жкспертік жүйе деп аталады?

- Білімге ие программа. Бұл кей алгоритмдерді орындай алатын икемдігі, мысалға қасиет табу барысында элементтер тізімін анализдеу. Бұл кез келген адамға сұрақтар тізімін беріп, одан жақсы нәтиже күтуге сай. Бірақ ерте ме, кеш пе ол осы тізімде қарастырылмаған бір қиыншылыққа тап болады.
- Білімге ие программа белгілі анықталған бір аймақта шоғырлануы тиіс. Кездейсоқ терілген аттар, күндер мен оқиға болған жерлер – бұл экспертік анализ жасауға керекті, эксперттік жүйеге негіз болады. Алайда ол білім бола алмайды. Білім белгілі бір ұйымдасу мен итерацияны ұсынады. Білім – яғни бір-бірінің соңынан жүретін, шынжыр түрінде байланысқан бөлек-бөлек мәліметтер жиыны.
- Соңында бұл білімдерден проблема шешімінің табылуы.

Енді бұл ойларды эксперттік жүйенің келесі анықтамасына сәйкес қорытамыз. Эксперттік жүйе – бұл компьютерлерге арналған программа, ол шешім немесе кеңес беру мақсатында белгілі бір аймақты қамтиды. Эксперттік жүйе шешім қабылдайтын адамның ассистенті және адам қатысуын талап ететін функцияларды толығымен орындай алады. Шешімді кім қабылдаса, сол өзінің құқығы бар эксперт бола алады және сол кезде ғана программа өзін-өзі ақтайды. Балама нұсқа – осындай программамен істейтін адам оның көмегімен үлкен жетістіктерге жете алады. Эксперттік жүйенің енгізілуі адам мен машина арасындағы функцияларды дұрыс бөлуде жақсы нәтиже береді.

Резолюция әдісі

Бұл G формуласының логикалық нәтижесі $F_1,F_2,...,F_k$ формуласы болатынын дәлелдеу әдісіне берілген. Бұл әдіс резолюция әдісі деп аталады. Логикалық құралдар туралы есеп есептің орындалуына әкеледі. Расында да, G формуласының логикалық құралы $F_1,F_2,...,F_k$ формуласы болады және $\{F_1,F_2,...,F_k, \neg G\}$жиын формуласы орындалмайды. Резолюция әдісі нақты айтқанда орындалмауын көрсетеді. Бұл – әдістің бірінші ерекшелігі. Екінші

ерекшелігі – ол туынды формуланы емес, дизъюнктарды (немесе элементар дизъюнкцияны) көрсетеді.

Логикалық құралдар. Литерал атомарлы формуланы немесе оның кері түрі, дизъюнкт – литералдар дизъюнкциясы деп аталады. Дизъюнкт бір литералдан тұруы мүмкін. Дизъюнктті литералдар жиыны деп алсада болады немесе дизъюнктті айырмаса да болады, себебі, коммутативті және ассоциативті дизъюнкциядан бір-бірінің көмегімен шығады. Мысалы $X \vee \neg Y \vee X$ и $X \vee \neg Y$ дизъюнкттері тең. Бізге ерекше бос (ішінде литералы жоқ) дизъюнкт керек. Оны «квадратпен» белгілейміз □. Бос дизъюнкт кез келген интерпретацияда жалған деп есептейміз. Бұдан F&□ формуласы □ тең , ал $F \vee$ □ формуласы F-ке тең. Бос дизъюнкт те тура солай, себебі атомарлы формула 0, контекстті резолюция әдісінде □-ты қолдану керек.

Анықтама. L және $\neg L$ литералы қарама-қарсы деп аталады.

Логикалық құралдарда резолюция әдісі резолюция ережесіне негізделген.

Анықтама. Резолюция ережесі логикалық құралдардан келесі ереже шығады: резолюция ережесінен $X \vee F$ и $\neg X \vee G$ дизъюнктінен $F \vee G$ дизъюнкті шығады.

Мысалы, $\neg X \vee Y \vee Z$ и $X \vee \neg Y$ дизъюнкттерден $Y \vee Z \vee \neg Y$ дизъюнкттері шығады. Назар аударар болсақ, бірінші екі дизъюнкттерде тағы бір жұп қарама-қарсы литералдар шығады. Резолюция ережесі тек сол литералдарда қолданады деген тұжырым алсақ, онда Y және $\neg Y$-ке қолданылған резолюция ережесінен $\neg X \vee Z \vee X$ шығады.

Анықтама. S – дизъюнкттар жиыны болсын. S-тың нәтижесі дизъюнкттар тізімі.

$D_1, D_2, ..., D_n$ дегеніміз S-ға қатысты әрбір дизъюнкт тізімі немесе бұдан резолюцияның соңғы ережесі шығады. Егер S-ң соңғы дизъюнкті D болса, D дизъюнкті S-ң нәтижесі.

Мысалы, егер $S=\{\neg X \vee Y \vee Z,\ \neg Y \vee U,\ X\}$, онда $D_1=\neg X \vee Y \vee Z$, $D_2=\neg Y \vee U$, $D_3=\neg X \vee Z \vee U$, $D_4=X$, $D_5=Z \vee U$ – S-дан шыққан нәтижесі. $Z \vee U$ дизъюнкті S-дан шығады.

Резолюция әдісін қолдану келесі тұжырымнан шығады және ол толық резолюция әдісінің теоремасы деп аталады.

Теорема 1. Логикалық құралдар дизъюнкттар S жиыны S-дан бос дизъюнкт шықса, тек сонда ғана орындалмайды.

Дәлелдеу үшін, G формуласы логикалық жиыны $F_1,\ldots,F_k$ формуласына резолюция әдісі келесі түрде қолданылады. Алдымен $T=\{F_1,\ldots,F_k,\neg G\}$ формулалар жиыны құрылады. Одан кейін бұл формулалардың әр қайсысы КНФ-ке келтіріледі және шыққан формулалардан конъюнкция сызылады. S дизъюнкттар жиыны шығады. Нәтижесінде S-дан бос дизъюнктті іздейді. Егер S-дан бос дизъюнкт алсақ, онда G формуласы үшін $F_1,\ldots,F_k$ логикалық формула болады. Егер S-дан алынбаса, онда G формуласы $F_1,\ldots,F_k$ логикалық формуласы шықпайды.

Бұл мысалды кері алсақ, G=Z формула. G=Z формуласы логикалық болады, нәтижесінде $F_1=\neg X \vee Y \to X\&Z$, $F_2=\neg Y \to Z$. $T=\{F_1,F_2,\ \neg G\}$жиындар

формуласы. F_1 және F_2 формуласын КНФ-қа келтіреміз ($\neg G$ формуласында осы форма болады). Нәтижесінде:
F_1 эквивалентті $X\&(\neg Y \vee Z)$,
F_2 эквивалентті $(Y \vee Z)$.
Онда S дизъюнкттар жиыны тең:
$\{X, \neg Y \vee Z, Y \vee Z, \neg Z\}$.
S жиынынан бос дизъюнкт оңай алынады:
$\neg Y \vee Z, \neg Z, \neg Y, Y \vee Z$, У, □.

Бұдан G формуласы F_1, және F_2 логикалық формуласын шығады.

Бірінші қатарлы логикаға көшейік. Айнымалыға байланысты дизъюнктке тапсырыс берейік, ол жалпы кванторлармен байланыста болады, бірақ кванторларды өзіміз жазбаймыз. Бұдан шығатыны – екі бірдей айнымалы түрлі дизъюнкттарда әртүрлі болады.

Байқайтын болсақ, бірінші қатарлы логикада резолюция ережесінің бұл түрі орындалмайды. Расында да S={P(x),ØP(a)} дизъюнкттар жиыны орындалмайды, (себебі х айнымалысы жалпы квантормен байланысты). Осы уақытта егер логикалық құралдар үшін резолюция ережесін қолдансақ, онда S бос дизъюнктін ала алмаймыз. Бұл жағдайда не істеу керек. P(x) дизъюнктін кез келген х үшін P(x) ақиқат, бірақ P(x) және x=a үшін де ақиқат болады. x=a деп алсақ, S'={P(a),ØP(a)} дизъюнкттар жиынын аламыз. S жиыны және S' бір мезетте орындалады (немесе орындалмайды). S' ішінен бастапқы резолюция ережесінің көмегімен тривиалды түрі шығады. Бұл мысалда бірінші қатарлы логикадағы резолюция ережесіне қосымша мүмкіндіктер қосу қажет екендігін көрсетеді.

Қажетті тұжырымдар жасайық.

Анықтама. Ауыстыру деп теңсіздіктер жиыны аталады $s=\{x_1=t_1, x_2=t_2,\ldots, x_n=t_n\}$,

$x_1,x_2,\ldots,x_n$ – әртүрлі айнымалы, $t_1,t_2,\ldots,t_n$ – терм, t_i термінде x_i (1£ i £ n) айнымалысы жоқ.

Егер $s = (x_1=t_1,...,x_n=t_n)$, ал F – дизъюнкт, s(F) арқылы дизъюнктті белгілейміз, F бірқалыпты айнымалыдан шыққан x_1 –ден t1; және т.б. x_n -нен t_n. Мысал, егер $s=\{x_1=f(x_2), x_2=c, x_3=g(x_4))$, $F=R(x_1,x_2,x_3)ÚØP(f(x_2))$, онда $s(F)=R(f(x_2), c, g(x_4))ÚØP(f(c)$. Термдер осы сияқты орындалады.

Ыңғайлы болуы үшін теңсіздігі жоқ бос ауыстыруды енгізейік. Бос ауыстыруды е арқылы белгілейік.

Анықтама. $\{E_1,\ldots,E_k\}$ – литералдар немесе термдер жиыны болсын. Егер $s(E_1)=s(E_2)=\ldots=s(E_k)$ болса, s ауыстыруы осы жиын үшін *унификатор деп аталады*. Егер осы жиын үшін унификатор болса, *унификацияланған жиын* болады.

Мысалы, атомдар формулаларының жиыны
{Q(a,x,f(x)), Q(u,y,z)}
унификациялайды {u=a, x=y,z=f(y)}, ал жиын
{R(x,f(x)), R(u,u)}
унификацияланбайды. Расында да, егер х-ті u-ға ауыстырсақ, {R(u,f(u), R(u,u)} жиыны шығады.

u=f(u) ауыстыру жасау мүмкін емес және ол пайдасыз, R(f(u), f(f(u))) және R(f(u), f(u)), формуласына келеді.

Егер жиын унификацияланған болса, онда ереже бойынша осы жиынның бір ғана емес бірнеше унификаторы болады. Берілген барлық унификаторлардың ішінен жалпы унификатор бөліп алуға болады.

Анықтама. Егер $x=\{x_1=t_1, x_2=t_2,\ldots, x_k=t_k\}$ және $h=\{y_1=s_1, y_2=s_2,\ldots, y_l=s_l\}$ – екі теңдеу. X, h ауыстырса, онда

$\{x_1=h(t_1), x_2=h(t_2),\ldots,x_k=h(t_k), y_1=s_1, y_2=s_2,\ldots, y_l=s_l\}$ (4)

$x_i=x_i$ үшін 1£ i£ k, $y_j=s_j$, егер $y_j\hat{I}\{x_1,\ldots, x_k\}$, 1 £ j £ l үшін осы теңдеулерді сызамыз.

Нәтижесі үшін дизъюнкттің орнына префиксті функционалды жазуды қоямыз, сондықтан x және h орнына h◦x аламыз, x- ті сосын h-ті сызамыз.

Мысал қарастырайық. x = {x=f(y), z=y, u=g(d)}, h = {x=c, y=z} болсын. Онда теңдіктер тізімі (4) келесі түрде болады

{x=f(y), z=z, u=g(d), x=c, y=z}.

Осы тізбектерден h◦x = {x=f(y), u=g(d), y=z} шығады.

Осы тізбектің ассоциативті екенін дәлелдеу қиын емес, сондықтан кез келген x,h,x үшін x◦(h◦z)=(x◦h)◦z орындалады және бос ауыстыру көбейтуге қарағанда нейтралды элемент болады. Соңында s◦e=e◦s=s кез келген ауыстыру үшін шығады.

$s = \{x_1=t_1\}\circ\{x_2=t_2\}\circ\ldots\circ\{x_n=t_n\}$ үшін тізбектер тізімі: $s = (x_1=t_1, x_2=t_2,\ldots, x_n=t_n)$. &s орнына дизъюнкт (және терм) қойсақ, тізбек x_1 t_1-ге ауысады, x_2 t_2 - ге ауысады және т.с.с., x_n t_n-ге ауысады

Анықтама. Унификатор s жиындар литералы немесе термдер Егер кез келген t унификаторы үшін литералдар жиыны бар, x ауыстыруы былай болады t=x○s.

Мысал, {P(x,f(a), g(z)), P(f(b),y,v)} жиыны үшін жалпы унификатор s={x=f(b), y=f(a), v=g(z)} ауыстыруы болады. Егер t орына {x=f(b), y=f(a), z=c, v=g(c)} унификаторын алсақ, онда x={z=c}.

Егер литералдар жиыны унификацияланған болса, жалпы унификатор бар болады. Бұл тұжырымды параграф соңында дәлелдейміз. Ал қазір жалпы унификаторды табу алгоритмін көрсетейік. Алгоритм *унификация алгоритмі деп аталады*. Алгоритмді көрсету үшін жиындар теңдігі қажет.

Анықтама. M – литералдар немесе термдер жиыны. Бірінші сол жақ позициясын бөліп аламыз, ол жерде барлық литералдар бір символдан тұрмайды. Символдан басталатын, сол позицияны алатын әрбір литералдан теңдеу жазып аламыз. (Бұл теңдеу литералдың өзі, атомарлы формула және терм). Теңдеулерден шыққан жиын M-дағы бекітілген жиын деп аталады.

Мысалы, егер M={P(x, f(y), a), P(x,u, g(y)), P(x, c, v)}, бірінші сол жақ позициясында барлық литералдар. Бекітілген жиын f(y), u, c термдерінен тұрады. {P(x, y), ØP(a, g(z))} ол жиын. Егер M={ØP(x, y),ØQ(a, v)}, онда мынаған тең: {P(x, y), Q(a, v)}.

Унификация алгоритмі

1-қадам. k=0, M_k=M, s_k=e аламыз.

2-қадам. Егер M_k жиыны бір литералдан тұрса, онда s_k-ны жалпы унификатор деп алып, жұмысты аяқтау. Басқа жағдайда N_k жиынын M_k-ға байланысты табу.

3-қадам. Егер N_k жиынында v_k айнымалысы және терм t_k , v_k-ға кірмейтін бар болса, онда 4-ке көшеміз, әйтпесе, M жиыны унификатор емес және жұмысты аяқтау деген хаттама береді.

4-қадам. $s_{k+1}=\{v_k, t_k\}\circ s_k$–ға s_{k+1}–ді қойса, s_k-дан v_k t_k –ға алмасады, және $v_k=t_k$ теңсіздігі жазылуы мүмкін. M_k жиыны $v_k=t_k$ алмасуы орындалуы шыққан литералдар жиынын M_{k+1} деп қарастырады.

5-қадам. k=k+1 деп және 2 қадамға өтеді.

M={P(x,f(y)), P(a,u)} болсын. Унификация алгоритмінің жұмысын M жиыны арқылы көрсетейік N_0={x,a} болғандықтан M. алгоритмнің басында s_1={x=a} табылады,. M_1 жиыны {P(a,f(y)),P(a,u)}-ге тең. Содан соң s_2={x=a, u=f(y)} және M_2={P(a,f(u))}-ге ашылады. Себебі M_2 1 литералдан тұрады жұмысты аяқтайды және s_2 береді.

Екінші мысалды қарастырайық. M={P(x,f(y)), P(a,b)} болсын. Алгоритмнің бірінші қадамынан s_1=(x=a) және M_1={P(a,f(y)), P(a,b)}. 3-ші қадамның 2-шісінде M жиыны унификацияланбағаны туралы хаттама жібереді, себебі N_1={f(y),a} – де айнымалы жоқ.

4 қадам орындалғанда, M_k жиынында айнымалылардың бірі өшіріледі (v_k айнымалысы) және жаңа айнымалы пайда болмайды. Бұл дегеніміз унификация алгоритмі әрқашан жұмысты аяқтайды деген сөз, себебі 4 қадам шексіз. Алгоритм 3 қадамда жұмысын аяқтаса, M жиыны унификацияланбаған. Алгоритм 2 қадамда жұмысын аяқтаса, s_k – M жиынының унификаторы. Егер s_k – жалпы унификатор болса, дәлелдеу қиын, бірақ біз мұны дәлелдеп көрейік.

Теорема 2. Егер M – бос емес литералдар және M унификатор болса, онда алгоритм унификациясы 2 қадамда жұмысын аяқтайды да, s_k – M жиынының жалпы унификаторы болады.

Дәлелдеу. Егер t – M жиынының унификаторы. Индукцией по k индукциясы бойынша дәлелдейміз a_k бар егер $t=a_k\circ s_k$.

***Индукция базасы*:** k=0. онда s_k=e және a_k орнына t алуға болады.

Қадам индукциясы: k мағынасы 0£k£ l теңсіздігінде орындалады десек, a_k дегеніміз $t=a_k\circ s_k$.

Егер s_l(M) бір литерал құраса, онда алгоритм келесі 2 қадамда тоқтайды. Сонда s_l жалпы унификатор болады, себебі $t=a_l\circ s_l$.

Егер s_l(M) 1 литералдан артық болса. Онда алгоритм унификациясында жиын N_l – теңеседі. a_l N_l жиынын унификациялауы керек, себебі $t=a_l\circ s_l$ – M жиынының унификаторы. N_l – унификаторының ең болмағанда бір v айнымалысы болуы керек.

Егер t – терм болса, онда унификатор айнымалы v болады, онда N_l жиынына унификатор a_l, сондықтан $a_l(v)=a_l(t)$. Бұдан t-ң ішінде v жоқ. 4 қадам алгоритмінде s_{l+1} –ді алу үшін v=t, т.е. $s_{l+1}=\{v=t\}\circ s_l$ теңдігін қолданған. $a_l(v)=a_l(t)$ бұдан a_l – ң ішінде $v=a_l(t)$ теңдеуі шығады..

Егер $a_{l+1}=a_l\backslash\{v=a_l(t)\}$. Онда $a_{l+1}(t)=a_l(t)$, t-ң ішінде v жоқ. Ары қарай

$a_{l+1}\circ\{v=t\}=a_{l+1}È\{v=a_{l+1}(t)\}=a_{l+1}È\{v=a_l(t)\}=a_l$.
$a_l=a_{l+1}\circ\{v=t\}$. Бұдан,
$t=a_l\circ s_l=a_{l+1}\circ\{v=t\}\circ s_l=a_{l+1}\circ s_{l+1}$.

Кез келген k үшін a_k бар, $t=a_k\circ s_k$. M жиыны унифицияланған, онда алгоритм 2 қадамда жұмысты аяқтау қажет. Сонда соңғы теңдеу s_k M жиынының унификаторы, s_k(M) жиыны жалпы унификатор, себебі туынды унификатор үшін s_k – ға қойылған t бар, ол $t=a_k\circ s_k$.

Теорема дәлелденді.

1.3. Visual Prolog декларативті программалау тілін жоғары оқу орындарында оқыту

Логика элементтерін информатика курсына енгізу маңызды іс. Ол аталған оқу құралының басты жетістігі болып есептеледі. Өйткені заманауи білім берудің негізгі міндеттерінің бірі ретінде жоғары оқу орны оқушыларының логикалық дұрыс ойлауын дамыту аталады. Бүгінде адам санасының мүмкіндіктерін кеңейтетін танымның көптеген әртүрлі әдістері бар: үлгілеу және математикалық әдістер, соның ішінде мүмкін боларлық теориясының әдістері, физикалық және биологиялық эксперименттер, ЭЕМ-де ақпарат өңдеу, т.б. Бірақ осы әдістердің барлығын тиімді пайдалану үшін адамның логикалық ойлауы дұрыс болуы керек, сондықтан да ғылым ғана логикаға дұрыс ойлау заңдарын тануға үйретеді. Әрине, адам логиканың дәл ережелері мен заңдарын білмей-ақ, тек оларды түйсік деңгейінде қолдана отырып та дұрыс ойлай алады. Алайда логикаға ие адам анағұрлым дәл ойлайды, оның аргументациясы сенімдірек екенін де ұмытпау керек. Танымал американдық психолог Дейл Карнеги «Қалай мазасыздықтан арылып, өмір сүруді бастау қажет» [1989, 153 б.] кітабында былай жазады: «Дұрыс және бұрыс ойлау салты арасындағы айырмашылық келесіден тұрады: дұрыс ойлау салты себеп-салдарды талдауға негізделген, ол логикалық конструктивті жоспарлауға жетелейді; бұрыс ойлау салты қиындықтар мен жүйкенің тозуына жиі соқтырады».

Логикалық ойлау туа бітетін қасиет емес, сондықтан оны түрлі әдістермен дамытуға болады және дамытуға тиіспіз. Логиканы жүйелі меңгеру – осы бағыттың анағұрлым тиімді жолдарының бірі.

Педагогикалық құралдарды таңдаудың теориясын әзірлеудің әдіснамалық негізі *жүйелік амал* болып табылады. Жүйелік амал педагогикалық үдерістің біртұтас артуын қамтамасыз етеді, оны оңтайландырудың шарты болады.

Әдістемелік оқыту жүйесі аясында, жұмыстарда қалыптастырылған [Пышкало,1975; Монахов, 1985; Бабанек, 1988] анықтамаларды салыстыру және қорыту негізінде өзара байланысқан бес құрауыштардың жиыны:

- мақсатты;
- мазмұнды;
- операциялық-іс қызметті (әдістер, формалар және оқыту құралдары);
- бақылау-реттеу (оқытудың қойылған міндеттерін шешу барысын оқытушының бір уақыттың бақылауы және оқу операцияларын орындалу дұрыстығын оқушылардың өздері бақылауы);

- бағалау-нәтижелі (оқыту үдерісінде қол жеткен нәтижелерді педагогтардың бағалауы және оқушының өзін-өзі бағалауы, олар қойған оқыту міндеттерінің сәйкестігін орнату, анықталатын ауытқулардың себептерін айқындау, жаңа оқыту міндеттерін қою).

Әдістемелік оқыту жүйесін жүйелік амалды қолдана отырып құрастыру үдерісі реттелетін жүйені құруды қалайды. Бұл зерттеушіден құрамдық құрауыштар туралы тұтас ойды және оларды өзара дұрыс әрекеттесуде қарастыруды талап етеді. Әдістемелік оқыту жүйесінің құрауыштарының өзара әрекеті оның функционалды сызбасында көрнекі орын алады.

Әдістемелік жүйе күрделі динамикалық қалыптан тұрады. Оқытудың мақсаты мен міндеті оның мазмұнын анықтайды. Оқыту мазмұны оқытуды ұйымдастырудың нақты әдістері мен формаларын талап етеді. Оқыту барысы бойынша ауызша, жазбаша, зертханалық-практикалық жұмыстар, сынақтар мен емтихандар көмегімен жүзеге асатын ағымдық және қорытынды бақылау қажет. Бақылау оқу үдерісінде кері байланыстың жұмыс істеуін – оқытушының қиындықтар дәрежесі туралы, оқыту тапсырмаларын кезеңмен шешу сапасы туралы ақпаратты алуын қамтамасыз етеді. Кері байланыс оқу үдерісін ұйымдастыруды, корригирлеуді реттеу, оқыту әдісі мен формасына өзгерістер енгізу, оларды сол жағдай үшін оңтайлысына келтіру қажеттігін тудырады. Қол жеткізілген нәтижелерді бағалау қойылған оқу-тәрбие міндеттеріне сәйкестіктен нақты ауытқушылықтарды анықтап, білім мен ептіліктен анықталған кемшіліктерді ескеретін жаңа міндеттерді жобалауды талап ете алады. Ақыр соңында, әдістемелік жүйенің барлық құрауыштары өзінің жиынымен нақты нәтижелерді алуды қамтамасыз етеді, оның талдамасы білім мен ептілікте анықталған кемшіліктердің орнын толтыру қажеттілігін ескеретін оқытудың қосымша міндеттерін қою жолымен айқындалатын ауытқушылықтардың себептерін анықтауға және жоюға мүмкіндік береді.

Оқытудың әдістемелік жүйесінің өзара ішкі байланыстары оның түрлі жұмыс істеу тәсілдерін таңдаудың барынша кең мүмкіндіктеріне жол береді. Сондықтан әдістемелік жүйені жетілдірудің бағыттарын анықтайтын қағидалар қажет. Бұндай қағидаларды А.М. Пышкало [1975] *оқытудың әдістемелік жүйесін жетілдіру ұстанымдары* деп атады.

Әдістемелік жүйені жетілдірудің орталық ұстанымы *мақсаттылық ұстанымы* болып табылады: әдістемелік жүйені жетілдірудің бағыттары мен нәтижелері және оның құрауыштары көбінесе студенттерді оқыту мақсаттарына сай болуы керек.

Әдістемелік жүйені жетілдіру бағыты жүйелік амалдың мәнінен туындайтын нақты талаптарды ескермеуге тиім емес. Жүйенің құрауыштарының бірінің кез келген өзгерісі міндетті түрде өзгелеріне әсер етеді. Осы ойдан келіп келесі жайт туындайды: А.М. Пышкало *өзара байланыс ұстанымы* деп атаған әдістемелік жүйені жетілдіру ұстанымы: әдістемелік жүйенің құрауыштары өзгерген кезде осыдан туындайтын салдарын қалған басқа да құрауыштары үшін анықтау қажет.

Көрсетілген ұстанымды *толықтық ұстанымы* деп аталатын жүйедегі барлық өзара байланыстарды қарау талабымен толықтыру қажет: әдістемелік жүйені жетілдіру кезінде оның құрылымының әрбір элементіне көңіл бөлген жөн.

Оқытудың әдістемелік жүйесіне ықпал еткен белгілі дидактикалық ұстанымдар:

1. Оқытудың әдістемелік жүйесінің *мақсаттылық* ұстанымы. Ойластырылған мақсат оқытудың әдістемелік жүйесіне идеялық бағыттылық, саналық және шығармашылық сипат береді.

2. Оқыту мазмұнының *ғылыми* ұстанымы [Краевский, Лернер,1983]:

а) білім мазмұнының заманауи ғылым деңгейіне сәйкестігін;

ә) студенттердің танымның жеке және жалпы ғылыми әдістері туралы түсінігін қалыптастыру;

б) таным үдерісінің маңызды заңдылықтарын үйрену.

3. *Қол жетімділік* ұстанымы, осыған сәйкес интеллектуалдық, физикалық және моралды салмақсыз оқушылардың мүмкіндік деңгейінде құрылуы тиіс.

4. Жоспарлаудың түрлі формаларында оқыту үдерісінде іске асырылатын *жүйелік және кезектілік* ұстанымы. Жоспарлау кезінде оқу материалының тараулары арасында логикалық байланыстар орнатылады, тақырыптар мен олардың жеке сұрақтарын игерудің реті, теориялық және зертханалық жұмыстардың кезектілігі, оқу материалын қайталау және меңгеру дәрежесін бақылау белгіленеді. Логикалық байланыстарды сақтаған жағдайда оқу және дүниетаным материалдары үлкен көлемде және анағұрлым берік игеріледі деген заңдылық бекітілген. Жүйелік пен кезектілік аз уақытта үлкен нәтижелерге жетуге мүмкіндік береді.

5. Көрнекілік ұстанымы Я.А. Коменскийдің «дидактиканың алтын ережесіне» негізделеді, оқытуға сезімнің барлық мүшелерін тартуды талап етеді. Оқыту көрнекілігі оқу үдерісінде әртүрлі демонстрацияларды, зертханалық-практикалық жұмыстарды, оқу плакаттарын, сызбаларды, т.б. қолданумен қамтамасыз етіледі.

6. *Оқытудың оңтайлы әдістерін, формаларын және құралдарын таңдау* ұстанымы педагогтарға ғылыми негізде нақты пән үшін сәйкес келетін жұмыс тәсілдерін таңдау құқығын береді. Бұл ұстаным оқытушыларды әдістемелік шығармашылыққа шақырады, озық тәжірибенің жетістіктерін, сондай-ақ оқыту саласындағы өзінің зерттеу іс-қызметін қолдану үшін кең мүмкіндік береді.

7. *Оқыту нәтижелерінің беріктік, ұғынушылық және әрекеттік* ұстанымы, осыған сәйкес оқытудың әдістемелік жүйесін құруда оқу-танымдық іс-қызмет дағдыларының беріктігін, алынған білімнің игерілуін және ептілікті, дағдыларды қамтамасыз ету.

«Логикалық программалау негіздері» пәні бойынша оқу материалын іріктеуге және оқу-әдістемелік құралдарды құруға бағытталған;

1.Бірінші кезең оқытудың мақсаттары мен міндеттерін негіздеумен байланысты, ол қарастырылатын оқу пәнінің нақты бір материалында білім беру мазмұнының жалпы құрылымы – білімді, ептілік пен дағдыларды,

оқыту үдерісінде қалыптасатын жеке тұлғаның шығармашылық, дүниетанымдық және тәртіптік қасиеттерін көрсетеді.

2.Екінші кезең оқыту мазмұнын іріктеумен байланысты.

«Оқыту мазмұнын ғылыми негіздеп іріктеу және мерзімді түзету – дидактиканың анағұрлым маңызды, бітпейтін мәселелерінің бірі. Ғылыми білімді үздіксіз жаңартумен, оларды саралап жіктеумен және ықпалдастырумен, жаңа еңбек объектілері мен құралдарының, технологиялардың, материалдардың, т.б. пайда болуымен қолдау табатын ғылыми-техникалық прогресті жылдамдату жағдайында бұл мәселе, әсіресе құрамына информатика мен есептеу техникасы кіретін ғылымның, техниканың, өндірістің динамикалық, тез дамушы салалары туралы сөз қозғалғанда елеулі күрделене түседі. Осыған орай теориялық негіздеуге және ғылыми ізденістің бағытын белгілеуге мүмкіндік беретін айқын әдіснамалық бағдарларды басшылыққа алу, оқытудың мазмұнын іріктеуде әлі да көп кездесетін эмпирикалық амалды жеңу маңызды», - бұл Б.С. Гершунскийдің [1987] әлі де маңыздылығын жоймаған тезисі.

Осы мәселені шешу үшін өзара байланысқан, бірақ бір-біріне тән емес түсініктер: *«оқыту мазмұны», «оқу материалы»* және *«оқу пәні»* түсініктерін мағынасын дәлдеу қалды.

3. *Үшінші кезең оқу материалын* іріктеумен, жүйелеумен және жіктеумен және *оқу пәнін* құрастырумен байланысты. «Оқу пәні» түсінігі В.В.Давыдов [1972], Л.Клинберг [1984], И.Я.Лернер [1980], М.Н.Скаткин [1984], В.С.Ледкев [1980] жұмыстарында талқыланады. Оқу пәнінің заты *ғылыми білімдердің дидактикалық өтелген жүйесінен* тұраты.

Білімнің ғылыми жүйесінің ғылыми емесінен айрықша өзгешелігі оның құрамында ерекше құрауыштардың бар екендігі болып табылады [Бабанский,1988]:

- ғылымдардың негіздері (оның негізінде жатқан іргелі қағидалар);
- оның сүйенетін санаттары мен ұғымдары;
- қандай да бір құбылыстарды суреттейтін, түсіндіретін және болжайтын теориялары;
- сол ғылымды дамыту үдерісінде орнатылған заңдары мен заңдылықтары;
- ғылыми білімнің тиісті саласындағы іс-қызметтің мазмұны мен түрлерін анықтайтын ұстанымдары мен ережелері;
- өзінің объектісін зерттеу үдерісінде осы ғылым қолданатын әдістер;
- ғылыми білімдердің сол саласының дамуын алдан ала болжауға мүмкіндік беретін идеялар;
- қандай да бір теориялардың, заңдардың, заңдылықтардың негізінде жатқан фактілер.

Меңгерілетін ғылымның логикалық құрылымының құрауыштарын талдау маңызды болып табылады, өйткені соның негізінде тиісті оқу пәнінің құрылымдық құрауыштарын анықтауға және ғылыми негіздеуге болады.

4. *Төртінші кезеңде* ұсынылған оқу-программалық құжатнама мен оқу-әдістемелік құралдарды *тәжірибелік-эксперименталды мақұлдау* жүргізілген және олардың мазмұнына қажетті түзетулер енгізілген.

«Логикалық программалау негіздері» пәнінің мақсаттары мен міндеттері

«Логикалық программалау негіздері» пәні бойынша оқыту мазмұнын іріктеуге бағдар жүйесі ретінде алынған оқыту мақсаттары мен міндеттерінің жиыны суреттеледі.

ЛБН пәнінің программасымен қарастырылатын мақсаттар келесі қағидалар бойынша жүзеге асырылады:

1) түрлі пәндердің тапсырмаларын шешу кезінде сабақ барысында мұғалімдер мен оқушылар үшін пайдалы бола алатын компьютерлерді пайдалану ауқымын кеңейтуге студенттерді барынша ынталандыру;
2) студенттердің компьютер мүмкіндіктерін игеруде және осы мақсатта оларды қолданудың жаңа құралдары мен тәсілдерін меңгеруде сенімділік сезімін дамытуға ықпал ету;
3) түрлі тапсырмалардың типтерін – логика ұстанымдарына сүйенетін амалды адам ойлауының ақпараттық-логикалық үлгілерін құруға декларативті амал қалыптастыру арқылы студенттердің логикалық және ақпаратық мәдениетін арттыруға ықпал ету;
4) студенттердің тапсырмаларды шешу кезінде логикалық программалау тілінің құралдарымен шектелген адам ойлауының ақпараттық-логикалық үлгілерін құру үшін қажетті дағдылар туралы түсінігін қалыптастыру;
5) студенттерді компьютерлік және ақпараттық жүйелерге адамгершілік-жауапкершілік қатынастарына тәрбиелеу;
6) студенттерді тапсырмаларды шешу кезінде компьютерді қолданудың қандай да бір нақты жағдайының артықшылығын, кемшілігін және шектеулерін бағалауға үйрету;
7) студенттердің пән туралы жалпы түсінігін және осы пәннің оқыту әдістемесін ақпараттық технологияларды оқу курсының тарауы ретінде игеруін қамтамасыз ету.

ЛБН пәнінің аясында оқыту міндеттерін анықтау кезінде келесілер қарастырылған:

1. Есептеуіш жүйелердің көмегімен ақпаратты ұсынумен, берумен, сақтаумен және өңдеумен байланысты теоретикалық мәселелерді білу.

2. Программалаудың негізгі ұстанымдарын білу және қиындықтың орташа деңгейінің міндеттеріне қолдана білу. Программаның тиімді және ыңғайлы ұйымдастырылғанын анықтай білу.

3. Мәліметтер мен білімдер құруға және қолдануға байланысты теориялық мәселелерді, эксперттік жүйелерді білу және сәйкес программалық қамтамасыз етуді қолдануға практикалық ептілік.

4. ЭЕМ-де қолданбалы міндеттерді шешу үшін программалық құралдарды пайдалана білу. ЭЕМ көмегімен мәселелердің қандай типтері шешіле алатындығын және осындай мәселелерді шешу үшін қандай құрал-саймандар қажет екендігін білу.

5. Ғылыми теорияның қандай да бір бөлігін түсіну және оны дидактикалық оқу пәнінің бөлігіне айналдыра білу.

6. Жоғары оқу орындарында ақпараттық технологиялар тарауын түрлі деңгейлерде оқытудың мазмұндық және әдістемелік аспектілерін білу: базалық курс, төменгі сыныптар, тереңдетіп оқыту, факультативтер, сыныптан тыс жұмыс.

7. Жоғарғы оқу орындарында ақпараттық технологиялардың негізгі программалық-әдістемелік кешенін қолдану мазмұны мен әдістемесін білу.

8. Информатика мен кибернетика тарихын, олардың даму келешегін білу.

9. Өзінің білімін ақпараттық технологиялар мен оны оқыту әдістемесі саласында әрі қарай жетілдіруге дайындық.

1. ***Білім***

«Логикалық программалау негіздері» пәнін үйрену нәтижесінде студенттер білуі керек: ЭЕМ-ді заманауи және болашақ программалық қамтамасыз етудегі логикалық программалаудың рөлі мен орны: логикалық программалаудың негізгі конструкциялары мен ұстанымдары; Пролог тілінің негізгі түсініктері; прологтық конструкциялардың синтаксисі мен семантикасы; мәліметтердің рұқсат етілетін типтері; Visual Prolog программасының құрылымы; Пролог құралдарымен білімді ұсыну ұстанымдары; ақпараттық-логикалық үлгілер құру кезеңдері; Прологтың декларативті және есептеуіш үлгілері; Прологты программалаудың негізгі әдістері; ұстанымдары; мәліметтердің күрделі құрылымдарын өңдеу; терезелерді, графиктерді және дыбысты қолдану тәсілдері; логикалық тапсырмаларды шешудегі Прологтың мүмкіндіктері, мәліметтер базаларын және білім базаларын басқару жүйесін құру, эксперттік жүйелер құру, өңдеу: табиғи-тілдік конструкциялар; орта оқу орындарындағы информатиканың базалық және факультативті курстарында логикалық программалаудың әдістемелік ұстанымдарын білу.

2. ***Арнайы қабілеттер мен дағдылар***

Пәнді игерген соң студенттердің қолынан мыналар келуі тиіс: Visual Prolog программалау жүйесімен жұмысты бастай және аяқтай білу; қолданушының талабы бойынша турбо-қабықша конфигурациясының баптауын орындау; мәтіндік редактордың құралдарын қолдана отырып программаны теру және редакциялау; ішкі және сыртқы мақсаттық пайымдауларды қолдана отырып диалог режимінде программаны орындау; трассировка режимінде программаны реттеу (программаны сатылап орындау); программаны дискіде мәтіндік файл түрінде сақтау және оны дискіден компьютердің жедел жадына «жүктеу»; программаны және орындалатын файл ды exe типіне құрастыру; тапсырманы дұрыс қоюды сауатты орындау; Visual Prolog-тың синтаксис және семантика ережелеріне сәйкес программаларды құру; программаны орындау нәтижесін экранға және басып шығару құрылғысына берілуін ұйымдастыру; мәліметтердің рұқсат етілетін типтері бойынша негізгі операцияларды орындау; қарапайым мәліметтер базалары мен білім базаларын құру, оларға сұранымдармен байланысу; тапсырмаларды шешудің ақпараттық-логикалық үлгілерін құру; құрылған үлгіні жүзеге асыратын программаны әзірлеу және орындау,

алынған нәтижелерге талдау жасау; ақпаратты автоматтандырылған іздеу тапсырмаларын шешу; әртүрлі пән салаларының білім базаларын өңдеу бойынша қарапайым эксперттік жүйе құру.

3. ***Жалпы кәсіби қабілеттер мен дағдылар***

Логикалық программалауды үйрену студенттердің болашақ мамандар ретінде қажет болатын жалпы кәсіби қабілеттері мен дағдыларының дамуына септігін тигізеді; оқулықтарда жүргізілетін және әдебиеттерде жарияланатын программаларды оқу; кәсіби қарым-қатынастың (ауызша және жазбаша сөз сөйлеу) түрлі тәжірибесін оқу кезеңінде алу; түрлі типтер мен деңгейдегі білім беру ұйымдарының жұмыстарына қатысу. В.А. Каймин мен Ю.С. Завальский [1991, 23 б.] жалпы орта білім беретін және математикалық оқу орындарының оқушыларына арналған «Информатика мен есептеуіш техника негіздері» курсы бойынша эксперименттік программа шығарды. Формальді логиканың жеке элементтерін үйренуді Прологтың негізгі ұғымдарын параллельді үйренумен қоса қарастыра отырып программа Прологты практикалық қолдануға арналған программалау тілі ретінде үйренуге бағытталған.

3. Пән пререквизиттері: Жасанды интеллект, сараптаушы жүйелер, логикалық программалау

4. Пән постреквизиттері: Математикалық логика, информатика, мәліметтер қоры, жүйелік программалау.

5. Күнтізбелік-тақырыптық жоспар

№	Пән тақырыптарының аталуы	апта	Аудиториялық сабақтар		Тапсырма түрі (сипаттамасы)		Барлығы (сағ)
			Дәріс (сағ.)	Пр/сем./зерт х./ студ саб (сағ.)	СОӨЖ	СӨЖ	
1	Кіріспе. Компьютерлік интеллект және роботтық техника	**1**	**1**	**2**	**1**	**2**	**6**
2	Мақсат. Берілгендер. Білім қоры. Факті және ереже ұғымдары	**2**	**1**	**2**	**1**	**2**	**6**
3	Жеңілдету және шығару механизмдері. Талқылаудың тікелей және кері тізбектері	**3**	**1**	**2**	**1**	**2**	**6**
4	Білімді көрсету моделдері. Фрейм. Логикалық модель. Семантикалық тор. Өнімдік модель	**4**	**1**	**2**	**1**	**2**	**6**
5	**2-бөлім.** Сараптаушы жүйе. Жүйе құрылымы	**5**	**1**	**2**	**1**	**2**	**6**
6	Сараптаушы жүйелерді классификациялау. Сараптаушы жүйені дайындау	**6**	**1**	**2**	**1**	**2**	**6**
7	Есепті тұйық формада	**7**	**1**	**2**	**1**	**2**	**6**

	көрсету. Есепті шешіу кезеңдері. Бағалау. Түйістіру						
8	**3-бөлім.** Логикалық программаның өзіндік қасиеттері және негізгі ұғымдары	**8**	**1**	**2**	**1**	**2**	**6**
9	Визуал Пролог ортасы және мәліметтердің берілуі	**9**	**1**	**2**	**1**	**2**	**6**
10	Ортақ тағайындаудағы стандартты предикаттар және пролог программасын басқару	**10**	**1**	**2**	**1**	**2**	**6**
11	Визуал Пролог ортасында рекурсия және циклдерді ұйымдастыру	**11**	**1**	**2**	**1**	**2**	**6**
12	Визуал Пролог логикалық программалау ортасындағы тізімдер	**12**	**1**	**2**	**1**	**2**	**6**
13	Визуал Пролог логикалық программалау ортасындағы жолдар	**13**	**1**	**2**	**1**	**2**	**6**
14	Визуал Пролог логикалық программалау ортасындағы мәліметтер қоры	**14**	**1**	**2**	**1**	**2**	**6**
15	Визуал Пролог жүйесінің графикалық мүмкіндіктері	**15**	**1**	**2**	**1**	**2**	**6**

2.VISUAL PROLOG ДЕКЛАРАТИВТІ ПРОГРАММАЛАУ ТІЛІНІҢ ЗЕРТХАНАЛЫҚ ЖҰМЫСТАРЫ

Қарастырылып отырған зертханалық жұмыстар «Жасанды интеллект негіздері» пәнінде қолданылады. Зертханалық жұмыстар жұмыс программасы негізінде дайындалды. 10-зертханалық жұмыс 15 сағатқа арналған.

2.1. №1-зертханалық жұмыс
Visual Prolog логикалық программалау тілін орнату және танысу

Мақсаты: Программа ортасымен танысу.

Visual Prolog логикалық программалау тілінің ортасы.

Жұмысты бастау үшін міндетті түрде Test Goal деп аталатын визуальды утилитін ашу керек. Бұл жұмыс Project-Test Goal командасының көмегімен немесе <Ctrl>+<G> пернелерімен жүзеге асырылады. Test Goal жақсы жұмыс жасауы үшін арнайы енгізілген жобалар қолданылады. Әрдайым келесі TestGoal-жобасын құрып соны пайдаланған жөн.

Үлгілерді орындауға арналған TestGoal-жобасы.

Кейбір анықталмаған компилятор Visual Prolog Test Goal жобасын қолдануды талап етеді.

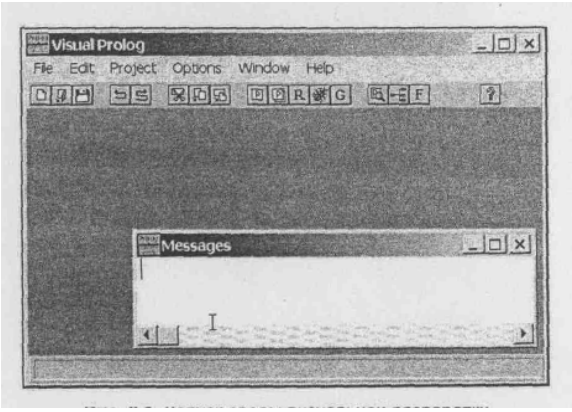
Ол үшін келесі әрекеттерді орындаңыз:

1. Визуалды Visual Prolog өнім орталығын енгізіңіз.

VDE алғаш енгізгеніңізде жоба бірден енбейді және 1-суреттегі құбылысты көресіз.

2. Жаңа жоба құрыңыз.

Project-New Project командасын таңдаңыз, Application Expert сұхбат терезесі пайда болады.

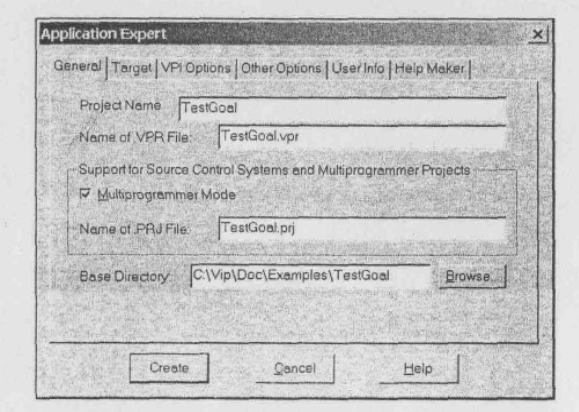


1-сурет. Визуальды орта.

3. Жобаның аты мен негізгі каталогын анықтаңыз. Base Directory келесі түрде ат беріңіз:

C:\Program files\VIP52\DOC\Examples\TestGoal

Бұл анықтама болашақта енгізілетін мәтін үлгілеріне өте ыңғайлы. Project Name алаңында «TestGoal» деп алған жөн. Сондай-ақ Multiprogrammer Mode жалаушасын құрыңыз және Name of.PRJ File алаңында тышқанды басыңыз. Алаңда TestGoal.prj жобасы пайда болады.

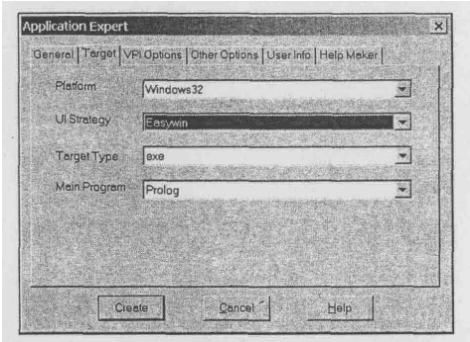


2-сурет. Application Expert диалог терезесінің жалпы құрылымы.

Жобаның мақсатын анықтаңыз.

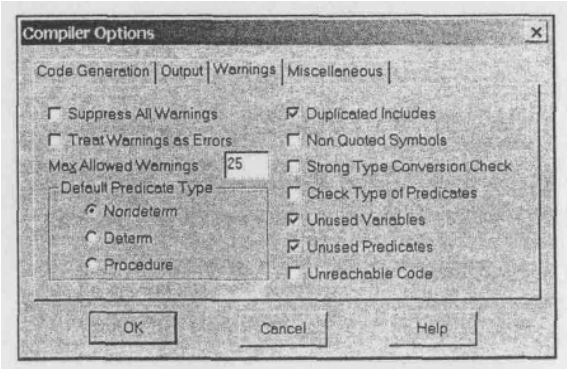
3-суретте көрсетілген Target параметрларын қойған жөн. Енді жоба файлын құру үшін Create тетігін басыңыз.

4. TestGoal жобасын құруда қажетті компилятор опцияларын енгізіңіз. Диалогтық терезе Compiler Options командасын белсендендіру үшін Options-Project-Compiler Options командасын таңдаңыз.
Warnings ашыңыз. Келесі әрекеттерді орындаңыз.
• Nondeterm қосқышын енгізіңіз. Бұл Visual Prolog компиляторы үшін қажет.



3-сурет. Диалогты терезе Application Expert Target құрылымында.

• Non Quoted Symbols Strong Type Conversion Check және Check Type of Predicates жалаушаларын алып тастау.
• Компилятор опциясын сақтау үшін OK тетігі басылады.
Осы әрекеттердің нәтижесінде диалогтық терезе Compiler Options 4-суреттегідей көрінеді.



4-сурет. Компилятор опциясын енгізу.

Редактор терезесінің ашылуы

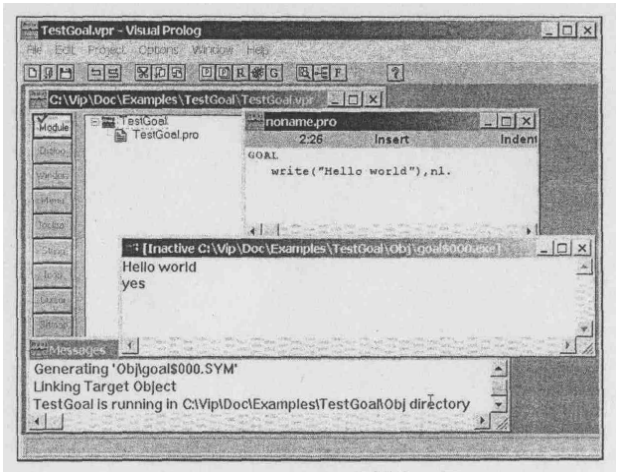
Жаңа редакторлық терезе құру үшін менюдегі File-New командасын қолданамыз. Нәтижесінде жаңа Noname деп аталатын терезе пайда болады. Визуальды орталығында – стандартты мәтін редакторы. Басқа редакторлардағыдай, курсор мен тышқанды пайдалануға болады. Ол Edit менюдағы Cut, Copy және Paste, Undo и Redo командаларын қолдайды. Сондай-ақ, Edit менюында осы әрекеттер үшін «ыстық» комбинация көрсетілген. Айтылған редактор көмекші VDE (редактор терезесіндегі <F1> пернесі) жүйесінде орналасқан.

Программаны енгізу және мәтіндеу.

Сіздің жүйеңіз дұрыс жұмыс жасауын тексеру үшін келесі мәтінді терезеге енгізу қажет:

```
GOAL
write("Hello world"), nl.
```

Пролог терминінде бұл *GOAL* деп аталады және ол программаның орындалуы үшін осы жеткілікті. GOAL орындау үшін Project|Test Goal командасын енгізуіңіз керек немесе <Ctrl>+<G> комбинациясын басыңыз. Егер сіздің енгізген жүйеңіз дұрыс болса, онда монитор экранында 5-суреттегідей көрініс пайда болады.



5-сурет. «Hello world» тест программасы.

Программаның орындалу қорытындысы жоғарғы жақта жеке терезелерде орналасады, (суретте ол Inactive C:\Vip\Doc\Examples\TestGoal\Obj\goal$000.exe деп аталады) және оны басқа GOAL-ды тестілеу кезінде жабу қажет.

Мысалдарды тестілеу.

Мысалдарды C:\Program files\VIP52\DOC\EXAMPLES каталогынан табуға болады..

Test Goal мысалында тестілеу

Визуальді өнім орталығында кез келген мысалды ашып, Test Goal утилитасын пайдалану арқылы тестілеу жүргізіңіз. Ол үшін келесі қадамдарды орындаңыз:

1. Visual Prolog визуальді өнім орталығын енгізіңіз.
2. Арнайы TestGoal жобасын ашу үшін Project|Open Project мәзір командасын қолданыңыз.
3. chCCeNN.pro.-ның кез келген файлын ашу үшін File|Open мәзір командасын қолданыңыз.
4. Project|Test Goal менюінде енгізілген мысалдарды тестілеу үшін мынадай командаларды қолдан: (немесе мына пернелерді басыңыз <Ctrl>+<G>).

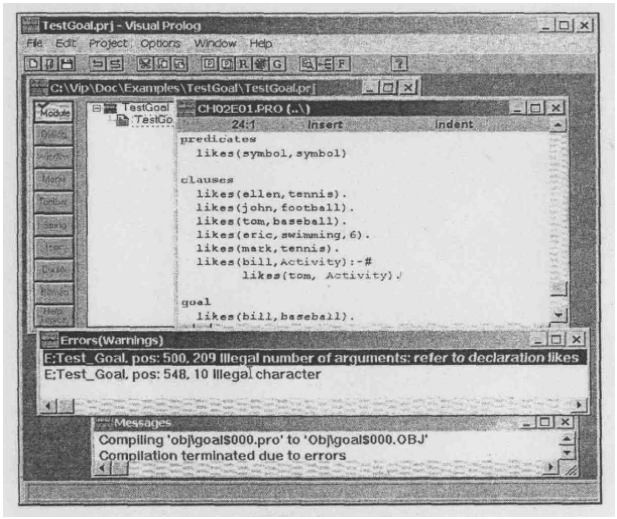
Test Goal GOAL программасында барлық мүмкін болатын қорытындыларды тауып, GOAL программасында қолданылатын барлық өзгерістерді көрсетеді.

Test Goal утилиті құрылымына комментарий

Визуальді өнім орталығының утилитасы GOAL-ды арнайы орындаушы файл программасы ретінде көрсетеді және Test Goal оны орындауға жібереді. Бұл утилита берілген GOAL кодын іштей кеңейтіп, енгізілген программалардың барлық жағдайдан шығуына мүмкіндік жасауына және қолданылған өзгерістердің мағыналарын көрсетеді. Test Goal утилитасы бұл кодты берілген жобаны компилятор опциясына қолданады.

Қателерді өңдеу

Егер сіз программа барысында қателік жіберсеңіз және оны жойғыңыз келсе, онда Errors (Warnings) терезесі пайда болып онда жіберілген қателердің тізімі беріледі (6-сурет).



6-сурет. Қатені жөндеу.

Қатені екі рет бассаңыз оның алғашқы мәтініне қайта келесіз. Visual Prolog интерактивті анықтамалық жүйесін шығару үшін <F1> клавишасын басасыз. Анықтамалық терезе ашылғаннан кейін, Search тетігін басыңыз. Қате нөмірді теріңіз, сонда экранда ол туралы толық ақпарат ала аласыз.

Пролог және құрылым программасының синтаксистік тілі.

Пролог программасы төмендегі кеңейтілген құрылымда болады.

domains
/* ...
Домендерді хабарлау
... */
predicates
/* ...
Предикаттарды хабарлау
... */
goal
/* ...
ішкі мақсат_1, ішкі мақсат_2, және т.б..
... */
clauses
/* ...
Сөйлем (фактілер және ереже)
...*/

Пролог программасының мақсатын анықтайтын ***Clauses бөлімінде*** фактілер мен ережелер бар.

Predicates секциясында предикаттармен осы предикаттардың аргумент типтері хабарланады.

Предикаттардың атауларында алу, жұлдызша пробел символдарын пайдалануға болмайды.

Предикаттар хабарландыруы келесі форманы құрайды:

predicates
predicateName (argument_type1, argument_type2,...,argument_typeN)

Мұндағы: ***argument_type1, ..., argument_typeN*** – стандартты домен немесе ***domains*** секциясындағы хабарланған домендер. Хабарланған домендер аргументтері мен аргумент типінің бейнелері екеуі бір.

Бақылау сұрақтары

1. Visual Prolog тілінің ортасы мен интерфейсі.
2. Тілдің құрауыштары мен утилиттері.

Есеп беру мазмұны және формасы: Тапсырмалар нұсқасын құрастыру. Тапсырманы орындау реті. Орындау нәтижелері. Қортынды.

2.1. №2-зертханалық жұмыс
Visual Prolog логикалық программалау тілінің негізі

Мақсаты: Фактілер, ережелер және сұрауларды орналастыру әдістері.

Ұсыныстар

Пролог тілін құрайтын тек екі түрлі ұғым бар: ереже немесе факті. Бұл ұғымдар прологта ұсыныс (clause) терминімен белгілі. Прологтағы программалардың негізі ұсыныстардан тұрады.

Факті объектілердің арасындағы қатынасты немесе қасиетті көрсетеді. Факті өзіне-өзі жеткілікті. Прологқа айғақтың расталуы үшін қосымша дәлелдеулер керек болмайды және де факті логикалық қорытындының негізі ретінде қолданылады.

Ережелерді қарастырсақ. Проглогта бір фактінің сенімділігінен басқада бірнеше фактілер табуға болады. Ереже – берілгендерді логикалық түрде суреттейтін прологтың құрылымы. Ереже – басқа да қатынастар белгілі болғанда нақтылы қасиет немесе қатынас. Бұл қатынастар үтірлермен бөлінген.

Ережелерге бірнеше мысалдар келтірейік.

1-мысал. Кейбір тағамдардың Диана мәзіріне сәйкес келетіндігіне қорытынды жасай алатын ережелер.

Диана-вегетарианшы ол тек қана өзінің дәрігерінің айтқанымен тамақтанады. (Diane is a vegetarian and eats only what her doctor tells her to eat)

Алдыңғы ереже мен мәзірді біле отырып, Диана мәзірдегі тағамдардың қайсысын таңдайтыны туралы алдын ала қорытынды жасай аласыз. Сіз бұл жұмысты іске асыру үшін тағамның берілген шектерге сәйкес келетіндігін тексеруіңіз керек:

- Food_on_menu көкөніс болып табылады ма?
- Food_on_menu дәрігердің тізіміне кіреді ме?
- шешім: егер екі жауап оң болса, Диана Food_on_menu- тапсырыс бере алады.

Пролгогтағы ұқсас қатынас ережемен анықталуы керек, өйткені қорытынды фактіге сай негізделген. Ережені жазудың бір нұсқасы:
diane_can_eat(Food_on_menu):-vegetable(Food_on_menu),
on_doctor_list(Food_on_menu).

vegetable(Food_on_menu) -дан кейін үтір тұратынына көңіл аударыңыз. Ол бірнеше мақсаттардың конъюнкциясын айқындайды және "және" сияқты оқылады; екі ереже де - diane_can_eat(Food_on_menu) үшін vegetable(Food_on_menu) және on_doctor_list(Food_on_menu) ақиқат болуы керек.

2-мысал. Егер Personl Person2-нің ата-анасы болып табылса, сіз ақиқат фактісін жазғыңыз келеді деп болжаймыз. Керек факті осылай көрінеді:
parent(paul, samantha).

Бұл айғақ бойынша Пол, Самантаның ата-анасы екендігін білдіреді. Бірақ Visual Prolog мәліметтер қорында факілер бар деп болжайық. Ол айғақтар әкелік қатынасты қалыптастырады. Мысалы, "Пол Самантаның әкесі":
father(paul, samantha).

Енді сізде аналық қатынас қалыптастырушы айғақ болсын дейік, "Джулия – Самантаның анасы":
mother(julie, samantha).

Егер сіздерде бірнеше фактілер бар болып және олар аталық-аналық қарым-қатынастарды қалыптастыратын болса, онда мәліметтер қорындағы әрбір ағайынды қатынасқа уақытты өлтірудің қажеті жоқ.

Сіз білетіндей, Personl Persona-ның ата-анасы, егер Personl-Person2-нің әкесі немесе Personl Person2-нің анасы болса, онда неге бұл шектерді біріктіретін ережелерді жазбасқа? Бұл шарттарды табиғи тілде қалыптастырғаннан кейін оларды пролог ережесі бойынша жазу жеткілікті.
parent(Personl, Person2):- father(Personl, Person2).
parent(Personl, Person2):- mother(Personl, Person2).

Бұл пролог ережелері

Personl Person2-нің ата-анасы болады, егер Person1 Person2-нің әкесі болса және

Personl Person2-нің анасы болады, егер Personl Person2-нің анасы болса.

3-мысал. Егер адамға машина (likes) ұнаса және машина сатылса (for sale), онда ол машинаны сатып ала алады.

Бұл ретте қатынас пролог тіліне келесі ереже бойынша ауыстырыла алады:
can_buy(Name, Model):-person(Name),
car(Model), likes(Name, Model), for_sale(Model).

Бұл ережелер келесі қатынастарды сипаттайды:

Name сатып ала алады (can_buy) Model, егер Name адам(person) және Model машина (car) болса, және Model сатылса (for_sale) Name Model –ді ұнатады (likes). Егер бұл ереже 4 шартты қанағаттандырса, онда ол ақиқат болады.

Автокөлік сату проблемасын шешудің жолын іздейтін ch02e02.pro-программасы көрсетілген. Тексеріп көріңіз:

```
predicates
can_buy(symbol, symbol)
person(symbol)
car(symbol)
likes(symbol, symbol)
for_sale(symbol)
clauses
can_buy(X,Y):-person(X), car(Y), likes(X,Y), for_sale(Y).
person(kelly).
person(]udy).
person(ellen).
person(mark).
car(lemon).
car(hot_rod).
likes(kelly, hot_rod).
ikes(judy, pizza).
likes(ellen, tennis).
likes(mark, tennis).
for_sale(pizza).
for_sale(lemon).
for_sale(hot_rod).
```

Джуди және Келли не сатып ала алады? Hot_rod-ты кім сатып алуы мүмкін? Келесі мақсаттарды GOAL: бөлімінде сынап көріңіз.
can_buy(Who, What). can_buy(judy, What). can_buy(kelly, What). can_buy(Who, hot_rod).

Бұл программаға басқа фактілер мен жаңа ережелер қосайық. Өзіңіздің алдында құрған сұраныстарыңыз бойынша программаны тексеріп көріңіз. Қорытындының нәтижелері сіздің күткеніңізді ақтады ма, әлде жоқ па?

Жаттығулар:

1. Visual Prolog-та келесі ережелерге сәйкестендіріп жазыңыз:
eats(Who, What):- food(What), likes(Who, What).
pass_class(Who):- did_homework(Who), good_attendance(Who).
owns(Who, What):- bought(Who, What).
2. Берілген сөйлемдерден Visual Prolog ережелерін құрыңыз:

- адамның асқазаны бос болса, ол аш;

- егер жақсы ақы төленетін және көңілді жұмыс болса, ол барлығына ұнайды;
- автокөлікті кім сатып алса, сол басқарады.

Предикаттар

Прологтағы қатынас предикат деп аталады. Аргументтер – бұл осы қатынастармен байланыстырылатын объектілер; айғақта likes(bill, cindy) қатынас; likes- бұл предикат, ал bill және cindy – аргументтер.

Әртүрлі сандардың аргументтерімен берілген бірнеше предикат мысалдары:

pred (integer, symbol)
person (last, first, gender)
run()
birthday (firstName, lastName, date)

Алдыңғы мысалда көрсетілгендей предикаттарда аргумент болмауы мүмкін, бірақ ондай предикаттардың қолданылуы шекеулі. Mr. Rosemont атын анықтау үшін person(rosemont, male, male) сұрауларын қолдануы мүмкін. Бірақ сұрауларда run аргументінің дәлелдерінсіз не істейді? Программада run ұсынысы бар ма анықтайық. Егер run ереженің басы болса, онда берілген ережені есептеуге болады. Кей жағдайларда бұл пайдалы. Мысалы, егер сіз әртүрлі соған сәйкес жұмыс істейтін программа құрғыңыз келсе, онда бұл run-ның ұсынысында болады.

Айнымалылар

Қарапайым сұрауда, кімнің теннисті жақсы көретіндігін табу үшін айнымалылар қолданамыз.

Мысалы:
likes. (X, tennis)

Бұл сұрауда X әріпін белгісіз адамды табу үшін айнымалы ретінде қолданамыз. Айнымалының аты Visual Prolog кейін (бас немесе кіші) әріптердің кез келген саны тұра алатын (немесе астын сызу символы) цифрлар немесе асты сызулы символдардан басталуы керек. Дұрыс айнымалының аты төменде келтірілген:

My_first_correct_variable_name Sales_10_ll_86

келесі үш - терісі:
1stattempt
second_attempt
"Disaster "

Атауларын әртүрлі регистрдің айнымалы әріптерімен қолданған ыңғайлы:
IncoraeAndExpenditureAccount

Мағынасы түсінікті таңдау айнымалының атын оқу үшін программаны өте ыңғайлы істейді.

Мысалы:
likes(Person, tennis)

жақсы,оған қарағанда
likes(X,tennis).
өйткені Person х-ке қарағанда көбірек мағына береді. Мақсатты енді сынап көріңіз:
GOAL likes (Person, tennis).
Visual Prolog жауап береді:
Pers on=ellen Person=mark
2 Solutions

Өйткені, мақсат екі шешім ellen және marktің мәндерімен айнымалы Personдарды дәйекті түрде салыстыра ала алады.

Айнымалыларды белгілеу

Прологта меншіктеу операторы жоқ. Бұл – прологтың басқа программалау тілдерінен басты айырмашылағының бірі. Прологта айнымалылар ақиқат немесе жалған.

Белгілеуге дейін айнымалы еркін; иемденулерден кейін оның мәнімен байланыста болады. Егер сұраулар бойынша айнымалысы шешім қабылдаса, онда сабақтас болып қалады, содан соң ол прологты босатады және басқа шешімді іздейді.

Мәліметті айнымалың мәнін беріп сақтауға болмайды. Айнымалы үдерістің бір бөлігі ретінде шешімді іздестіру үшін қолданылады.

Ch02e03.pro программасы қарап шыға алады, сонымен қатар айнымалы өз мәндерін алады.

```
predicates
likes(symbol, symbol)
clauses
likes(ellen, reading).
likes(John, computers).
likes(John, badminton).
likes(leonard, badminton).
likes(eric, swimming).
likes(eric, reading).
```

Сұрауларды қарап шығамыз: жүзу мен оқуды жақсы көретін адам бар ма?
likes(Person, reading), likes(Person, swimming).

Пролог программадағы айғақтарды басынан соңына дейін іздестіру арқылы сұраудың екі бөлігін де шеше бастады. Сұраудың бірінші бөлімі:
likes(Person, reading)

Person еркін айнымалы; пролог шешімді табар алдында оның мәні белгісіз. Басқа жағынан, екінші аргумент, reading атымен белгілі. Пролог бірінші сұраулардың бір бөлігіне сәйкес келетін айғақ іздейді. Программадағы бірінші айғақ

likes (ellen, reading)

Демек, пролог еркін Person айнымалысын ellen-ның мәнімен байланыстырады, (айғақтағы reading сұраулардағы reading-ге сәйкес келеді) бірінші сұраулардың бір бөлігін қанағаттандырады айғақты тиісті мәнге ұластырады. Дәл сол уақытта пролог көрсеткіші айғақтардың тізіміне бағытталады, ол іздеу процедурасының қаншалықты алға басқандығын көрсетеді.

Бұдан әрі, (жүзу мен оқуды ұнататын адамға іздеу салу) сұрауларды толық шешу үшін екінші сұраудың бір бөлігі шешілуі керек. Person ellen-мен байланысты болғандықтан, пролог айғақты іздеуі тиіс.

likes (ellen, swimming)

Пролог бұл айғақты программаның басынан іздейді, бірақ сәйкестік жоқ (өйткені программада мұндай айғақ жоқ). Егер Person ellen мәні болса сұраудың екінші бөлімі жалған болады.

Енді пролог айнымалы Personдарды босатады және бірінші сұраулардың бір бөлігінің басқа шешімін табуға талаптанады. Басқа айғақты іздестіру бірінші бөлімнің сұранысын қанағаттандырады, (белгіленген позицияға қайта оралу, қайта іздеу деп аталады) айғақтардың тізіміндегі нұсқағыштан басталады.

Пролог оқуды жақсы көретін және likes(eric, reading) айғағын табатын адамды іздейді. Person айнымалысы қазір eric мәнімен байланысты, және пролог екінші бөлімнің сұранысы бойынша айғақтар программасында қайта іздеуді бастайды.

likes(eric, swimming)

Пролог сәйкестендірілген (программадағы соңғы ұсыныс) және сұраныс толығымен қанағаттандырылады.

Person=eric. 1 Solution.

Анонимді айнымалы

Анонимді айнымалылар біздің программаларымызды ретке келтіруге мүмкіндік береді. Егер сізге нақтылы мәлімет керек болса, анонимді айнымалыларды керек емес мәліметтерді жою үшін пайдалануға болады. Прологта анонимді айнымалылар астын сызу символы арқылы белгіленеді (_).

Келесі "отбасылық" мысал анонимді айнымалыларды пайдалану арқылы іске асады. TestGoalға ch02e04.pro программасының жобасын жүктеңіз.

predicates
male(symbol)
female(symbol)
parent(symbol, symbol)
clauses
male(bill).
male(]oe).

female(sue).
female(tammy).
parent(bill,joe).
parent(sue,joe).
parent(joe,tammy).

Аноним айнымалысы кез келген басқа өзгерістердің орынында қолданылады және оған ешқандай мағына берілмейді.

Мысалы, бізге келесі сұрауда қандай адамдар ата-ана бола алатындығын білу керек бірақ, сізге олардың балалары қызықты емес. Прологтағы әрбір сұраныста сызылған символды падаланған кезде сізге айнымалының мағынасы туралы ақпараттың қажеті жоқ.

goal
parent(Parent, _).

Мұндай сұранысты алғаннан кейін пролог былай деп жауап береді: (Test Goal)
Parent=bill Parent=sue Parent=joe 3 Solutions

Бұл жағдайда пролог үш ата-ананы тауып береді, бірақ ол parent-тың ұсынысында екінші дәлелмен сабақтас мәндерді бермейді.

Анонимді айнымалыларды айғақтар ретінде пайдалануға болады. Прологтің келесі айғақтары:
owns(_, shoes). eats (_).

Табиғи тілде бекітулерді өрнек үшін қолдана алды:

Әрбір адамда аяқ киім бар (Everyone owns shoes). Әрбір адам тамақтана алады (Everyone eats).

Аноним айнымалысы кез келген мәліметтерге қарама-қарсы келеді.

Құрама мақсаттар: конъюнкция және дизъюнкция

Көріп отырғаныңыздай, құралған мақсаттарды іздеудің нәтижесі үшін қолдануға болады, екі ішкі мақсат А және В ақиқат (конъюнкция), ішкі мақсатты үтір арқылы ажыратамыз. Егер іздеудің нәтижесінде ақиқат А және В ішкі мақсат (дизъюнкция) болса бұл ішкі мақсатты нүктелі үтір арқылы ажыратамыз.

Төменде программаның мысалы көрсетілген:

predicates
car(symbol,long,integer,symbol,long)
truck(symbol,long,integer,symbol,long)
vehicle(symbol,long,integer,symbol,long)
clauses
car(Chrysler,130000,3,red,12000).
car(ford,90000,4,gray,25000).
car(datsun,8000,1,red,30000).
truck(ford,80000, 6,blue,8000).
truck(datsun,50000,5,orange, 20000) .

truck(toyota,25000,2,black,25000).
vehicle(Make,Odometer,Age,Color,Price):-car(Make,Odometer,Age,Color,Price);
truck(Make,Odometer,Age,Color,Price).

TestGoal-дың жобасына осы программаны жүктеңіз, содан соң мақсат қойыңыз:
Goal
car(Make, Odometer, Years_on_road, Body, 25000).

Берілген мақсат айтылған ұсыныстардан бағасы $25 000 тұратын машина (car) тұратын машина табуға тырысады. Пролог былай жауап береді:
Make=ford, Odometer=90000,
Years_on_road=4,
Body=gray
1 Solution.

Дегенмен берілген мақсат біршама жасанды. Жасанды болғандықтан тез арада мынандай типті сұрақ қойылады:
Тізімде $25000-дан арзан тұратын машина бар ма?(Is there a car listed that costs less than $25000)

Мұндай мақсатты шешу үшін сіз Visual Prolog-та келесі құралған мақсатты бере аласыз:
Car (Make, Odometer, Years_on_road, Body, Cost) % ішкі мақсат A және Cost < 25000. % ішкі мақсат B.

Бұл конъюнкция болып табылады. Құралған мақсатты шешу үшін пролог ішкі мақсаттарды кезек бойынша рет-ретімен шешуге тырысады. Бастапқыда ол мынаны шешуге талаптанады:
Car (Make, Odometer, Years_on_road, Body, Cost), содан соң
Cost < 25000.
Cost айнымалысы екі ішкі мақсаттарға ұқсас мәндерге ие. Енді осының барлығын Test Goalмен істеп көріңіздер.

Ескерту

Cost < 25 000 ішкі мақсаты Visual Prolog жүйесінде құралған «кем» қатынасына сәйкес келеді. Кем қатынастың басқа да екі сандық объектілерге пайдаланылатын қатынастардан ешқандай айымашылығы жоқ, бірақ, ол қатынастымына символмен (<) және екі объектінің арасында қатыстыра отырып жазған дұрыс.

Келесі өрнек ақиқат болып табыла ма? Табиғи тілде ол былайша сипатталады: Тізімде $25 000-дан кем тұратын автомобиль немесе $20 000-дан кем тұратын жүк машинасы бар ма?

Мына тапсырмада келесі құралған мақсаттарда пролог тиісті нәтижені іздестіруді қайта орындайды:
Car (Make, Odometer, Years_on_road, Body, Cost), Cost<25000% ішкі мақсат A немесе

truck (Make, Odometer, Years_pn_road, Body, Cost), 20000<Cost. % ішкі мақсат B.

Құралған мақсаттағы бұл тип дизъюнкция болып табылады. Берілген мақсат екі альтернативті ішкі мақсаттарды, бір ережеге екі ұсыныстың біріктірілгеніне сәйкес келуі мүмкін деп есептейді. Пролог екі ішкі мақсаттарды да қанағаттандыратын нәтижелердің барлығын іздейді.

Рұқсат етілген құралған мақсатта пролог келесі ішкі мақсаттардан құралған бірінші ішкі мақсатты іздеуге тырысады. («автокөлікті іздеу»):
car(Make, Odometer, Years_on_road, Body, Cost) және
Cost < 25000.

Егер автокөлік табылса мақсат-ақиқат,егер табылмаса(жүк машинасын іздеу) – пролог келесі ішкі мақсаттан тұратын екінші құрама мақсаттарды шешуге тырысады:
truck(Make, Odometer, Years_on_road, Body, Cost),
және
Cost < 20000.

Visual Prolog программасы

Visual Prolog синтаксисі қасиеттер және өзара байланыстарды анықтау үшін жасалған. Біз көбінесе (айғақтар және ереже) ұсыныс, предикаттар, айнымалы және мақсаттарды қарастырдық.

Прологтың басқа болжамдардан айырмашылықтары, Visual Prolog – бақылаушы типтердің компиляторы: әрбір предикат үшін қолданылатын объектілердің типін хабарлайды. Сонымен бірге, Visual Prolog программаларына бұл типтерді орындайтын жылдамдығы машина кодтарына сәйкес келуге мүкіндік береді, ал басқа жағдайда – Pascal тілдеріндегі ұқсас программалардың жылдамдығынан асады.

Енді Visual Prolog программасының негізгі төрт бөлімін талқылаймыз – жарияланатын және сипатталатын предикаттар, сонымен қатар аргумент типтері ,берілген және анықталған ережелер программаның мақсаты болып табылады. Бұдан әрі ережелердің синтаксисін және хабарламаларды толығырақ қарап шығамыз. Соңында, программаның басқа да бөлімдерін қысқаша қорытындылап сипаттаймыз: деректер қоры, тұрақтылар, әр түрлі глобалды бөлімдер және компилятордың нұсқауы.

Visual Prolog программасының негізгі бөлімдері:

Prolog Visual тіліндегі программа төрт негізгі программалық реттен тұрады. Оларға мыналар жатады:

- clauses (ұсыныстар) бөлімі;
- predicates (предикаттар) бөлімі;
- domains (домендер) бөлімі;
- goal (мақсаттар) бөлімі.

Clauses– Visual Prolog программасының негізгі бөлігі; нақ осы бөлімде айғақтар және ережелер жазылады, Visual Prolog операциясы программаның мақсатын шешуге тырысады.

Predicates бөлімінде (Visual Prologта кірістірілген предикаттарды жарияламауға да болады) предикаттарды және домендердің (түрлер) аргументтері жарияланады.

Domains бөлімі сіздер қолданылатын барлық домендерді хабарлау үшін қызмет көрсетеді, олар (стандартты домендерді жариялау міндетті емес) Visual Prolog-тың стандартты домендері бола алмайды.

Goal бөлімі – бұл сізге Visual Prolog-программасының мақсатын сыйғызып алады.

Ұсыныстар бөлімі

Программаны құру барысында сіз clauses-тың (ұсыныстар) бөлімінде барлық айғақтар мен ережелерлерді сыйғыза аласыз. Негізгі тарауда программадағы ұсыныстарға (айғақтар және ережелер) басты назар аудырылады: оларды қалай жазу керектігін білдіреді.

Егер сіз айғақтарды және ережелерді олардың прологта көрсетілуін түсінсеңіз, онда сіз clausestың бөліміндегі әрбір нақты предикат үшін барлық ұсыныс бірге орналасуы керек екендігін білесіз. Бір предикатты анықтайтын ұсыныстардың тізбегі процедура деп аталады.

Предикаттардың бөлімі

Егер clausestың бөліміндегі Visual Prolog тіліндегі программаның меншігіндегі предикаттарды сипаттақ, онда predicatestің (предикаттар) бөлімінде жарияланады; басқа жағдайды Visual Prolog түсінбейді. Сіз предикатты хабарлаудың нәтижесінде домендерге (түрлерге) бұл предикаттың аргументіне жатады.

Visual Prolog кірістірілген предикаттардың толық жиынымен орналасады және анықтама кітабы олардың толық сипатын көрсетеді.

Предикаттар айғақтарға және ережелерге тапсырма береді. Барлық предикаттар predicates бөлімінде олардың аргумент типтерінің (домендер) нұсқауы есептелінеді.

Сіз айғақтарыңыз бен ережелеріңіздің жұмысын істейтін (дәлелдер) объектілер типін жарияласаңыз Visual Prolog жұмысының тиімділігі едәуір өседі.

Қолданбалы предикатты жариялау

Предикаттың жариялануы сол предикаттың атымен басталады, содан соң дөңгелек ашылатын жақша ішінде нөл немесе предикаттың үлкен доменді(тип) аргументі жалғасады:

arguraent_typel OptionalNamel, argument_type2 OptionalName2 predicateName ....., argument_typeN OptionalName3

Әрбір доменді аргументтен кейін, және соңғы аргумент типі жабылатын жақшамен аяқталады. Clauses-тың бөліміндегі ұсыныстан айырмашылығы декларация предикаты нүктемен аяқталмайды, предикаттың доменді аргументі стандартты домен немесе domaines бөлімінде жарияланған домен болатынын атап өтеміз. Аргументтің атын OptionalNameK деп нұсқауға болады. Бұл – программаның оқылуын біраз жеңілдетеді, компилятордың орындалу жылдамдығы білінбейді.

Предикаттардың аттары

Предикаттың аты реттеле орналасқан әріптер, цифрлар және астын сызылған таңбалармен басталуы керек. Предикаттың аты 250 символымен шектеледі.

Домендер бөлімі

Дәстүрлі прологта тек – терм типі бар. Visual Prolog-та доменді барлық аргументтің предикаты ретінде жариялаймыз.

Бірнеше мысалды қарастырайық.

1. Осы мысал предикаттардың домендерін қалай құжаттауға көмектесетінін көрсетеді:

Франк - 45 жасқа толған еркек.

Стандартты домендерді қолана отырып, сіз тиісті домен предикатын қолдана аласыз:

person.(symbol, symbol, integer).

Көп жағдайларда мұндай жарияланулар өте жақсы жұмыс істейді. Дегенмен сіз программа жазылғаннан кейін бірнеше айлардан соң түзеткіңіз келді дейік. Предикаттың ұқсас хабарламасы сізге ештеңе айтпауы мүмкін. Керісінше, төменде көрсетілгендей бұл предикаттың декларациясы осы предикаттың аргументтерін ретке келтіруге көмектеседі:

```
Domains
name,sex = symbol
age = integer predicates
person (name, sex, age)
```

Меншікті домендердің хабарлауының басты артықшылықтарының бірі, Visual Prolog типтердің қатесін зерттеп отыру, мысалы:

same_sex (X, Y):- person (X, Sex, _), person (Sex, Y, _).

name және sex symbol ретінде көрсетілгеніне қарамастан, олар бір-біріне эквивалентті емес. Егер оларды шатыстырсаңыз, Visual Prolog сол қатені анықтауға мүмкіндік береді. Бұл программаңыз өте үлкен және күрделі болған жағдайларда пайдалы.

```
Domains
product, sum = integer
predicates
add_em_up (sum, sum, sum)
multiply_em (product, product, product)
clauses
add_em_up (X, Y, Sum):-Sum=X+Y.
multiply_em (X, Y, Product):-Product=X*Y.
```

Бұл программа екі операцияны орындайды: жинақтайы және көбейтеді.Оған мынадай мақсат беріп көрейік:

add_em_up (32, 54, Sum).

Visual Prolog (Test Goal) былай жауап береді:
Sum=86
1 Solution
сіз программаға екі бүтін санның қосындысын жібердіңіз.

Декларация және ережелер

Visual Prolog-та бірнеше кірістірілген стандартты домендер бар. Оларды domains бөлімінде декларарация типінде және сипатталмаған предикаттардың аргументінде қолдануға болады. Негізгі домендер 4 кестеде сипатталған

4-кесте

Домен	Сипатталуы	Іске асуы
short	Қысқа, таңбалы, сандық	Барлық платформалар (- 32 768–32 767) 16 бит
ushort	Қысқа, таңбасыз,сандық	(0–65 535) 16 биттік платформалар
long	Ұзын, таңбалы, сандық	барлық платформалар (- 2 147 483 648-2 147 483 647) 32 бит
ulong	Ұзын, таңбасыз, сандық	барлық платформалар (0-4 294 967 295) 32 бит
integer	Таңбалы, сандық, тәуелді көлемді платформаға ие	(- 32 768-32 767) 16 биттік платформалар (- 2 147 483 648-2 147 483 647) 32 биттік платформалар
unsigned	Таңбасыз, сандық,тәуелді көлемді платформаға ие	(0–65 535) 16 биттік платформалар (0-4 294 967 295) 32 биттік платформалар
byte		Барлық платформалар (0–55) 8 бит
word		Барлық платформалар (0–65 535) 16 бит
dword		Барлық платформалар (0–4 294 967 295) 32 бит

Сіздің программаңызда идентификаторлар мен жолдар бір-бірін ауыстыра алады, бірақ Visual Prologта олар бөлек сақталады. Идентификаторлар, идентификатор кестелерінде сақталады, ал олардың көрінісінде жолдардың идентификаторы емес, оның кестедегі индексі ғана қолданылады. Бұл идентификаторлардың қарама-қарсы қойылуы өте жылдам орындалатындығын білдіреді. Олар программада біренше жерде кездессе, олардың сақталуы да тығыз болады. Жолдар іздеу кестесінде сақталмайды және қарама-қарсы қоюдың міндеттілігінде Visual Prolog оның символдарын біртіндеп тесереді.

Тапсырмалар

Visual Prolog аяқталған телефондық анықтама секілді программаларды көрсетеді. Тек стандартты домендер пайдаланылғандықтан программаға domains бөлімінің қажеті жоқ.

Predicates

```
phone_number (symbol, symbol)
clauses
phone_number ("Albert " , "EZY-3665" ).
phone_number ("Betty " , "555-5233 " ) .
phone_number ("Carol " , "909-1010 " ) .
phone_number ("Dorothy " , "438-8400 " ) .
goal
```

программаны енгізіп, орындауға жібергеннен кейін рет-ретімен мақсаттарды еңгізіңіз.

```
phone_number ("Carol ", Number) .
phone_number ("438-8400 " Who, ).
phone_number ("Albert ", Number).
phone_number (Who, Number).
```

Енді ұсыныстарды өзгертіңіз. Kim және Dorothy бір телефон нөміріне ие деп есептейік. Бұл айғақты clauses бөліміне енгізіп, келесі мақсатты енгіземіз:
Phone_number ("438-8400" Who,).
Бұл сұраныстан сіз екі нәтиже алуыңыз керек:
Who=dorothy
Who=kim
2 Solutions.

Программасындағы char доменін сипаттау үшін isletter предикаты қолданылады. Тапсырама барысында оған төмендегідей мақсаттар қойылған:
isletter(%).
isletter(Q).
"Yes" немесе "No" сәйкес келетін мағынасына қарай қайтарады
Predicates
isletter(char)
clauses
= > белгісі % таңбасын анықтайды.
% мына теңдік "Алфавиттің алдында қойылады"
isletter (Ch):-‘ a’< = Ch, Ch<=’z’.
isletter (Ch):-‘A’<= Ch, Ch<=’Z’

Программасын енгізіп, Test Goal-да әрбір мақсатты рет-ретімен сынап көріңіз:

a) isletter (‘x’).
b) isletter (‘ 2 ‘).
c) isletter ("Hello ").
d) isletter (a) .
e) isletter (X).

(c) және (d) мақсаттары қателіктің түріне алып келеді, ал (e) мақсаты және "Free variable " (байланыспаған айнымалы) хабарламасын қайтарады. Сіз берілмеген объектінің a немесе z-ке қатынасын тексере алмайсыз.

Бақылау сұрақтары

3. Логикалық программалау тілдерінің мүмкіндіктері.
4. Тілдің құрылымы

Есеп беру мазмұны және формасы: Тапсырмалар нұсқасын құрастыру. Тапсырманы орындау реті. Орындау нәтижелері. Қорытынды.

2.3. №3-зертханалық жұмыс
Қолданбалы предикаттар

Мақсаты: Предикаттарды пролог ортасында қолдану әдістері.

Ереже – бұл шындығы кейбір шарттардан тәуелді болатын бекіту. Ереже программа бөліктерінен тұрады :- егер деп оқылады. Ереже, сонымен қатар айғақ, нүктемен бітеді. Бас айнымалы компоненттер ретінде бола алатын бұрын суреттеп айтылған предикаттардың бірі болып табылады. Ереженің басы бұл ереже анықтауы үшін арналған айғақты суреттеу.

Ереженің денесі ереженің басы шын болды үшін айғақтармен дәйекті түрде келісуі керек үшін мақсаттарды суреттейді. Дене үтірлермен немесе нүктелі үтір айырық термдердің тізімінде болады. Сонымен бірге:
:- орнына if терді пайдалануға,
, орнына and қолдануға,
; орынына or
қолдануға болады.

Тапсырмалар:
2-зерханалық жұмыс тапсырмаларын қарап қарапайым математикалық өрнектерді шешу мысалдарын көрсетіңіз.

Бақылау сұрақтары:

1. Математикалық функциялар қалай сипатталады?
2. Шамалардың қандай түрлері бар?

Есеп беру мазмұны және формасы: Тапсырмалар нұсқасын құрастыру. Тапсырманы орындау реті. Орындау нәтижелері. Қорытынды.

2.4. №4-зертханалық жұмыс
Рекурсия және циклдарды ұйымдастыру

Мақсаты: Турбо Пролог ортасында рекурсивті есептерді шешу әдістерін, арифметикалық амалдардын іс жүзінде қолдану.

Көптеген қарапайым арифметикалық есептеулер итерациалық программа көмегімен берілуі мүмкін. Мысалы, факторларды есептеу циклдардың көмегімен орындалады, яғни тізбектерді сандарға көбейтіп, керекті факториалды алуға болады. Қажетті процедура Паскаль тілінде жазылып, төменде көрсетілген.

1-мысал. Факториалды есептеу.

```
real factorial (N: integer);
var i: integer ;
f: real ;
begin
i:=o; f:=1;
While (i<N) do
begin
i:=i+1; f=f*i
end;
factorial:=f
end;
```

Пролог тілінде итерациялық құрылымдағы мұндай программа жоқ. Рекурсивті және итерациялы алгоритімдерді рекурсия қолданылады. Итерацияның рекурсиядан негізгі мүмкіншілігі тиімділігінде, ең маңызды тиімділігі жадыны қолдануы. Әрбір рекурсивті шақыруда рекурсияны орындау белгілі бір уақытқа дейін аяқталмай, байланысқан ақпараттарды сақтауды талап етіп, мәліметтердің құрылымын анықтау, стек фрагменті деп аталады.

Стек – арнайы мәліметтер құрылымы. Оның элементтері келесі қасиеттерге байланысты болады: соңғы болып стекқа келсе, ол бірінші болып стектан кетеді. Стек элементтері берілуіне байланысты реттелген.

Есептеудегі жады өлшеміне процедураға n рекурсивті қатынау n-ге сызықты байланысты болады. Программаның орындалу кезінде стектің толып кетпеуін қадағалау керек, өйткені нәтижесінде бар мәліметтерді жоғалтып алуымыз мүмкін. Рекурсивті программадан итерациялы программаның айырмашылығы шектелген жады көлеміне байланысты, бірақ итерация санына байланысты емес.

Турбо Пролог программасында қайталанатын іс-әрекеттерде рекурсия қолдануға болады. Турбо Пролог тілінде бүтін санның факториалын есептейтін программа жазайық. 5.4 программа екі ережеден тұрып, оның бірі рекурсивті болады. Байқайтынымыз, рекурсивті ереже үш ішкі мақсаттан тұрып, ал соңғы ішкі мақсат Fact X=X+FactY, X санының факториалын есептеп және X-1 факториал саны FactY айнымалысының мәні болмайынша есептелмейді.

2-мысал. Факториалды рекурсивті жолмен есептеу.

Программаның жұмысын, мысал арқылы түсіндірейік. Мақсат factorial(5,F) түрінде берілген 5-тің факториалын есептейік. Алдымен Пролог бұл мақсатты бірінші мақсаттың тақырыбымен теңестіреді. Теңестіру сәтсіз аяқталып, екінші ереже таңдалады. Бұл жағдайда теңестіру сәтті аяқталып, X

айнымалысының мәні 5-ке тең болады. Содан кейін Y айнымалысының мәні есептеліп, Турбо-Пролог factorial(Y,FactY) ішкі мақсатын есептеуге көшеді, мұндағы X=4 болып, жаңа мәнмен рекурсивті түрде екінші тәртіп қайта шақырылады.

Тапсырмалар:

1. 1-ден 100-ге дейінгі сандардың қосындысын табатын программа жазыңыз.
2. 1-ден 100-ге дейінгі тақ сандардың қосындысын табатын программа жазыңыз.
3. 1-ден 100-ге дейінгі жұп сандардың көбейтіндісін табатын программа жазыңыз.
4. 1-ден N-ге дейінгі сандардың ішінен X-тен санды табыңыз.
5. A және B аралығында жататын теріс сандарды санататын программа жазыңыз.

2.5. №5-зертханалық жұмыс
Тізімдер

Жұмыстың мақсаты: пролог программасында тізімдермен жұмыс жасауға дағдылану.

Қысқаша анықтама

Тізім – бұл бір ізді термадан тұратын, арнайы терманың түрі.
Ол квадрат жақша ішінде үтір арқылы жазылады. Мысалы:[1,2,-5,4,0]. Мұндай тізім келесідей сипатталады:

```
domains
 list=integer*
```

Егер тізім аралас типті болса, онда келесідей сипатталады:

```
Domains
 element=c(char); i(integer)
 list=element*
```

Мұндай жазу келесідей тізімге сәйкес келеді.
[i(15), i(-6), c(‘a’), i(8), c(‘z’)]

Тізім пролог программасының негізгі құрылымы болып табылады. Тізімдердің ыңғайлы жазылуы үшін: басы (head) және аяғы (tail) деген екі түсінік енгізілген. Мына тізім үшін [1,2,3] 1 элемент тізімнің басы болып саналады, ал қалған элементтер [2,3] тізімнің аяғы.

Келесі кестеде тізімге байланысты негізгі мысалдар келтірілген:

Тізім	Басы	Аяғы
[’a’,’b’,’c’]	‘a’	[’b’,’c’]

[1]	1	[]
[ ]	айқындалмаған	айқындалмаған
[[1,2,3],[3,4],[]]	[1,2,3]	[[3,4],[]]

Тізімнің басын аяғынан ажырату үшін | келесі символы пайдаланылады. Мысалы, тізімі үшін [X|Y] X – тізімнің басы,

Y – тізімнің аяғы.

Тізімдермен жұмысқа мысалдар:

а) қосу (X,L,L1) –L1 тізімі X элементінің L тізімінің басына қосылуы арқылы жасалынған.

қосу (X,L,[X|L]). /* тапсырманың шешуі */

б) жатады (X,L) – X элементі L тізіміне жатады.

жатады (X,[X|_]). /* шектеулі шарт */

жатады (X,[Y|T]):-X<>Y, жатады (X,T).

в) тіркеу (L1,L2,L3) - L1 және L2 тізімдері L3 тізімінде қосылады, яғни бірігеді.

L1 L2
H T1
H T3
L3

Келесі тапсырма үшін Пролог программасын пайдаланамыз**.** Элементтер саны берілген тізімнен, жаңа тізім құру керек. Ол тізімде берілген элементтер екі есе үлкейіп жазылған болу керек. Екі тізімнің де стандарт түрде элементтерін үтір арқылы орналастырып, квадрат жақша арқылы өрнектейміз.

```
/* барлық элементтердің екі есе өсу программасы */
domains
 list=real*
predicates
read_list(integer,list)
wr_list(list)
write_list(list)
new_list(list,list)
result
clauses
/*  N элементтен тұратын тізімді енгізу  */
read_list(0,[]).
read_list(N,[H|T]):-readreal(H),N1=N-1,
read_list(N1,T).
/* Тізім элементтерін шығару. Әрбір элементтен кейін үтір қойылады */
wr_list([]).
```

```
wr_list([H|T]):-write(H,’,’),wr_list(T).
/* Стандартты түрде тізімнің шығуы */
write_list(L):-write(‘[’),wr_list(L),cursor(A,B),
B1=B-1, cursor(A,B1),write(‘]’).
/* Қорытынды тізімнің сипатталуы */
new_list([],[]).
new_list([H1|T1],[H2|T2]):-H2=H1*2,new_list(T1,T2).
/*  */
result:-write(“Тізім элементерінің санын енгізіңіз”),nl,
  write(“N=”),readint(N),nl,
  write(“Тізім элементтерін енгізіңіз”)nl,
  read_list(N,L),
new_list(L,L1),
write(“Қорытынды тізім”),
write_list(L),nl,nl,
write(“Жаңа тізім L1=”),
write_list(L1).
  goal
  result.
```

Зертханалық жұмыстың мазмұны

Берілген L1 тізімінен жаңа L тізімін алу. Кезекті элемент, кезекті L1 тізімінің арифметикалық ортасына тең. Егер тізім элементтерінің саны 3-ке бөлінбесе, онда L2 тізімінің соңғы элементі 3-ке бөлінген немесе L1 тізімінің соңғы екі элементінің қосындысының бөліндісіне тең. L1 тізімі экранда жасырын түрде шығады. Программаның орындалу нәтижесінде L1 шығыс тізімі және L2 тізімі стандартты түрде экранға шығуы тиіс.

Тізімдерге тапсырмалар:

1. Тізімнің бірінші элементін анықтау.
2. Тізімнің соңғы элементін анықтау.
3. Тізімнің соңғы элементін өшіру.
4. Тізімде пайда болған бірінші элементті жою.
5. Берілген тізімдегі барлық кіріс элементтерін өшіру**.**
6. Тізім элементтерін кері ретпен шығару.
7. Тізімнің берілген қатар бойынша жазылуын кері бағытта өзгерту.
8. Тізім элементтерінің арифметикалық ортасын табу.

9. Берілген тізімнен полиндром жасау. Полиндром – бұл берілген тізім оңнан солға, солдан оңға қарай оқығанда да бірдей дыбысталатын тізім болып табылады. Полиндромға мысал - [1,2,3,2,1].

10.Тізім элементтерін циклдық қозғау арқылы шығару:

- - бір орын солға (оңға)
- - N орынға солға (оңға)

11. Тізім элементтерін қозғау арқылы шығару(циклдық емес)

- - бір орын солға (оңға)

- - N орын солға (оңға)

12. Берілген тізімді ұзындығы бірдей екі тізімге бөлу.
13. Берілген тізімді екіге бөлу. Оның бірінде берілген санға тең немесе одан кіші элементтер, ал екінші тізімде берілген саннан үлкен элементтер жинақталуы тиіс.
14. Тізімде берілген тақ орнындағы элементтердің және жұп орнындағы элементтердің қосындысын есептеу.
15. Тізімдегі алғашқы оң N элементті шығару**.**
16. Тізімдегі оң және теріс элементтердің санын есептеу.
17. Берілген тізімнің ең үлкен элементінің және ең кіші элементінің айырмасын табу.
18. Тізімнің барлық элементтерін берілген элементтердің арифметикалық ортасына кеміту.
19. X элементіне дейінгі тізімнің басын өшіру. X элементінен бастап тізімнің соңына дейін өшіру.
20.Тізімде берілген X элементтерін Y элементтерімен алмастыру.

Есептің мазмұны:

Бұл есепте келесі пункттер болуы тиіс:

- жұмыстың мақсаты;
- программаның мәтіні;
- тест нәтижелері**.**

2.6. № 6-зертханалық жұмыс
Жолдар

*Жұмыстың мақсаты**:*** Пролог тіліндегі программалардағы жолдары бар жұмыстың жаттығу дағдыларын алу.

Қысқаша анықтама мәліметтер: Прологта жолдармен жұмыс жасау үшін үйреншікті предикаттардың жиыны болуы керек.

а) жолдың ұзындығының анықталуы.

str_len(жол, ұзындық)(string,integer):(i,o),(i,i)

б) жолдардың біріктірілуі.

concat(1-ші және 2-ші беттегі жол 3-ші беттегі жолдарға бірігеді) (string, string, string): (i,i,o)(o,i,i)(i,o,i)(i,i,i)

в) төменгі жолдарды құру:

frontstr(символдар саны, кіру және шығу беттері, қалдық) (integer, string, string, string):(i,i,o,o)

Шеткі жол басқа жолдардан бастапқы символдар арқылы ерекшеленеді. Жол саны белгілі параметрлермен саналады және қалған жолдар біріктіріледі.

г) жолдың бөлінуінің екі түрі бар: біріншісі – символ, екіншісі – қалған жол бөліктері.

frontchar(жол, бірінші символ,
қалдық)(string,char,string):(i,o,o)(i,i,o)(i,o,i)(i,i,i)(o,i,i)

convert(“”,[]).

convert(Str, [H|T]):-frontchar(Str,H,Str1),convert(Str1,T).

Предикат Frontchar символдарды тізімді өрнектеу үшін қолданылады. Бұл процедураның орындалу ережесі төмендегідей:
д) тексеру, кіріспенің атымен жол болып табылады.
isname(жол)(string):(i)
Жол келесі талаптарға сай болуы тиіс:
- жол әріп, цифр және астын сызу символынан тұрады;
- жол әріптен басталады;
- символдардың арасында бос орын болмауы керек.
е)жолдардың атомдардан құрастырылуы:
ronttoken(жол, атом, қалдық)(string,string,string):(i,o,o)(i,i,o)(i,o,i)(i,i,i)(o,i,i)
Атоммен бола алады:
кіріспенің аты;
кіші сандарды көрсету;
жеке символ.
Fronttoken предикат (frontchardың предикаты бар ұқсастық бойынша)жолдарды атом тізіміне өрнектеу үшін қолданылады.
Жолдармен жұмыс жасау кезінде кіріспенің басқа да жолдары қолданылады:

- бас әріптерді кіші немесе керісінше жолдық өрнектеу
- upper_lower(1-ші және 2-ші беттер)(string,string):(i,i)(i,o)(o,i)
- жолдың символға немесе керісінше өрнектелуі str_char(жол,символ)(string,char):(i,o)(o,i)(i,i)
- жолдардың санға немесе керісінше өрнектелуі
- str_int(жол, бүтін сан)(string,integer):(i,o)(o.i)(i,i)
- жолдың нақ санға немесе керісінше өрнектелуі
- str_real(жол, нақты сан)(string,real):(i,o)(o,i)(i,i)
- символдардың санға (ASCIIкоды) немесе керісінше өрнектелуі
- char_int(символ, бүтін сан)(char, integer):(i,o)(o,i)(i,i)
-

Тапсырмалар:
Зетрханалық жұмыс бойынша тапсырма.
Енгізілген жолдарды экран бетіне шығару:

Жолдар бойынша есептер

1. Енгізілген жолда «а» әріпінің қанша рет кездесетінін санау.
2. Соңғы жолда «н» әріпінің екі рет қатар келуін тексеру.

3 Соңғы сөйлемде қанша сөз бар екенін санау.

4. Белгілі бір символдар кезегі келесі бір символды жояды және оны қайталап отырады, басқа символдардан ерекшеленіп тұрады.
5. Грам, граматика, фон сөзіндегі (қосып жасалған программа көмегімен) грамматикалық қателерді жөндеу.
6. Бос орын арқылы бөлінген сөздердің қатарынан «а» әріпінен басталатын сөздердің санын анықтау.

7. Бос орын арқылы бөлінген сөздердің қатарынан бастапқы және соңғы символдары бар бірдей сөздерді таңдау.
8 Бос орын арқылы бөлінген сөздердің қатарында соңғы сөзден басқа әрбір сөзден кейін үтір қою.
9 Символдар жолы берілген. Егер * символы болмаса, жолды өзгеріссіз қалдырамыз, кейде * символын + -ға ауыстыруға тура келеді.
10 Бос орын арқылы бөлінген сөздердің тізбегі берілген. Сөздің бірінші әріпі – бас әріп, қалғаны – кіші әріп. Экран бетіне сөздің бас әрпінен басталатын ең болмағанда бір әріпті шығару.

***Есеп беру мазмұны және формасы*:**
Есептеу нәтижесінде келесі тармақтар көрсетілуі тиіс:
Жұмыстың мақсаты. Есептің қойылу мәтіні. Тестілеудің нәтижелері. Қорытынды.

2.7. № 7-зертханлық жұмыс
Құрама объекттілер

Жұмыстың мақсаты: Құрама объектілері бар пролог программасының жаттығу дағдыларын қалыптастыру.

Қысқаша анықтама мәліметтер

Бекітілген объектілердің өз мәліметтері болады. Жай тип мәліметтері алты стандартты типтермен шектеледі. Келесі бектіулерге қарайық:
Жақсы көреді(петр,музыка).

Екі объекті де (петр,музыка) жай құрлымды және олар өзін-өзі көрсетеді. Өзін-өзі көрсететін кез келген объектілер жай құрылымды объектілер деп аталады. Программа құрлымы сол сияқты жай құрылымды объектілерден тұрса, онда жай құрылымды болады.

Егер объекті басқа бір объектіні немесе объектілер жиынын көрсетсе, онда құрама объекті деп аталады және бұл объектілерді қолданатын программа құрама құрылымды болады.

Программада көрсетілген құрылымдардың сипаттамасы мына түрде болады:
Domains
Жеке_кітапхана=кітап(аты,авторы,шығрылымы)
Шығарылымы= шығарылымы (баспахана, жыл)
Иесі, аты, авторы, баспахана= symbol
жыл= integer
predicates
коллекция (иесі, жеке_кітапхана)
clauses
Коллекция («Иванов», кітап(«прологты қолдану», «Ин, Соломон», шығарылым («Мир»,1993))).

Керекті ақпаратты шығару үшін ыңғайлы құрылымды деректер қоры ұсынылады. Бұл жерде объектілерге сілтеуге және оның компонеттерін

көрсетуге болады. Сіздерді қызықтыратын объектілер құрлымын құруға және компоненттерді нақты сипаттамасыз немесе жартылай сипаттамамен қалдыруға болады.

Келтірілген программаға мүмкін болатын мысалдар:

Ивановтың коллекциясында қандай кітаптар бар?

коллекция («Иванов», X).

Кітаптың авторы кім «прологты қолдану»?

Коллекция (_,кітап(«прологты қолдану»,X,_)).

Ережелер жиынын құруға болады. Ол ережелер деректер қорымен өзара әрекеттесуге ыңғайлы болады.

Мысалы:

Кітап(аты,авторы,шығырылым).

Коллекция(_,кітап(аты,авторы,шығарылым)).

Шығарылым(баспахана,жыл).

Кітап(_,_,шығарылым(баспахана,жыл)).

Жыл_шығарылым(жыл).

Шығарылым(_,жыл).

Бұл ережелерді келесі мысалдарда қолдануға болады.

Кітаптың авторы кім «прологты қолдану»?

Кітап («кіріспені қолдану»,X,_).

Ивановтың коллекциясында 1990 жылғы кітап шығарылымы бар ма?

Коллекция (X,_),шығарылым(_,1990).

Ескерту: құраушы предикаттар белгілі бір ереже құру үшін predicates бөлімінде болуы керек.

Зертханалық жұмыстың тапсырмалар мазмұны

Отбасы туралы деректер қорын құру. Әр отбасы мүшесі бір сөйлеммен көрсетіледі. Отбасы туралы ақпарат құрылым түрінде беріледі. Әр отбасының үш мүшесі болады. Олар: әйелі, күйеуі және балалары. Балаларды тізім арқылы көрсетеді. Әр отбасы мүшесі құрылымды көрсетеді және аты, тегі, туған күні, жұмысы деген төрт компонеттен тұрады. Жұмыс туралы ақпаратта қандай қызмет атқаратындығы көрсетіледі.

Мысалы:

Отбасы(отбасы_мүшесі(«Николай», «Иванов», күн(12,мамыр,1948), жұмыс(инженер,210)), отбасы_мүшесі(«Анна»,«Иванова»,күн(5,қантар1952), жұмыс(дәрігер,190)),

[отбасы_мүшесі(«Инна»,«Иванова»,күн(20,наурыз,19971)

жұмыс(студент,45)), отбасы_мүшесі(«Олег»,«Иванов»,күн(25,маусым,1978), жұмыс(оқушы,0))]).

Сұранысты жеңілдету үшін келесі предикаттарға: күйеуі, әйелі, балалары, отбасы_мүшесі, туған_күні, жұмысы.

Мәліметтер қорынан келесі ақпараттарды алыңдар:

Ивановтар отбасы мүшелерінің аттарын алу.

Қаңтар айында туған әйелдерді табу.

15 жастан кіші балаларды табу.

Кем дегенде екі балада бар отбасы тегін табу.
Күйеуінің тегін алмаған әйелі бар отбасын табу.
Әйелі жұмыс істемейтін отбасын табу.
Әкесі жоқ отбасын табу.
Егіз балалары бар отбасын табу.
1950 жылы дүниеге келген адамды табу.
Балалары жоқ отбасын табу.
Күйеуі жұмыс істемейтін, әйелі жұмыс істейтін отбасын табу.
Ата-анасы мен балаларының жас айырмашылығы 15 жас болатын отбасын табу.
Ивановтар отбасына кіретін кірісті табу.

Есептеудің маңызы: жұмыс мақсаты, тапсырманың құрылуы, мәтіннің программасы, жұмыс нәтижесінің программасы, қорытынды.

2.8. № 8-зертханалық жұмыс
Файлдар

Жұмыстың мақсаты: файлдарды қолданатын пролог програмаларын құруда практикалық дағдыларды қалыптастыру.

Қысқа анықтамалық міндеттер

Програмада файлдарды қолданғанда оған файлдық воменнің сипаттамасын қосу керек. Ол былайша беріледі:
file=datafile, мұндағы
file – доменнін стандартты типі(файлдық),
datafile – файлдың логикалық аты.
Файлды сипаттағанда бірнеше логикалық атыңды нұсқауға болады, бірақ сипаттаудын өзі жалғыз болуы қажет, мысалы:
file=datafile1;datafile2;datafile3

Файлдармен жұмыс кезінде келесі енгізілген Пролог предикаттары пайдалануы мүмкін: енгізу шығару логикалық құрылғылар переадресатциясының предикатары readdevice және writedevice, файлды жою предикаттары deletefile, файлады сақтау save, файлдың атын өзгерту renamefile, файлдың бар болуын тексеру existfile, мәліметтерді ішкі файлдық буферден берілген файлға жіберу flush, жинақтаушыларды орнату және жолдары disk, ағымдағы каталогты шығару dir, оқуға арналған файлдарды ашу openread, жазбалар openwrite, оқу/жазу openmodify, толықтырулар (жазуға деиін) openappend, файлды жабу closefile, файл соңына тексерулер eof, орнатулар немесе файл типін оқу filemode, орнатулар немесе файлдағы нұсқағыш орнын оқу filepos, файлдағы жолды оқу file_str.

Мысалы, файлға мәліметтер жаз әрекеттерінің реттелгені келесідей :
- файлды openwrite предикаты көьегімен ашу ;
- файлды writedevice предикатымен жазу құрылғысы ретінде тағайындалу;
- файлға жазудың өзі, мысалы, write немесе writef предикатор көмегімен;

- программаның тағайындалуына жауап беретін басқа кез келген предикаттар мен ережелердің қолданылуы;
- файлады close file предикатымен жабу.

Осы әрекеттердің барлығы программада келесі түрде сипатталуы мүмкін:

```
openwrite(datafile1,”File1.dat”),
writedevice(datafile1),
< файлға жазу >
< басқа кез келгенережелермен  предикаттары >
closefile(datafile1).
```

Осындай схема арқылы файлдан мәліметтерді оқу және файлға мәлімет жазу алдындағы әрекеттер сипатталуы мүмкін.

Енгізілген предикаттарда файлдармен жұмыс жасау үшін компьютердің стандартты құрылғыларының логикалық атаулары қалданылуы мүмкін keyboard (пернетақта), screen (дисплеи экран), printer (басып шығарғыш құрылғы).

Пернетақтадан мәліметтерді оқу және оларды файлға жазуды жүзеге асыратын программа мына түрде болуы мүмкін:

```
domains
file=datafile
kstr,fstr=string
predicates
readin(kstr,fstr) /* оқу-жазу */
create_a_file /* файлды құру */
goal
create_a_file.
clauses
create_a_file:-
nl,nl,write(“файл атын енгізу”),nl,nl,
readln(Filename),
openwrite(datafile,Filename),
writedevice(datafile),
readln(Kstr),/* пернетақтадан бірінші жолды енгізу*/
concat(Kstr,”\13\10”,Fstr),
readin(Kstr,Fstr),
closefile(datafile).
/* келесі жолдарды енгізу және оларды файлға жазу*/
readin(“stop”,_):-!.
readin(_,Fstr):- write(Fstr),readln(Kstr1),
concat(Kstr1,”\13\10”,Fstr1),
readin(Kstr1,Fstr1).
```

Келтірілген программада
Kstr –пенрнетақтадан енгізетін жол,
Fstr – файлға шығарылатын жол.

Filename файлының құрылымына “\13\10” символдарымен толықтырылған жолдар. Жолды “\13\10” символдарымен толықтыру readln предикатының жолдары оларды файлдан оқудан ажыратып алу қажет. Програма жұмысы “stop” жолын енгізуде анықталады.

Зертханалық жұмыс бойынша тапсырма мазмұны.

Сөздер реттілігі, (мысалы әйел адам есімі), бос орындармен бөлінген, енгізілген жолдар аргументін енгізілген жолдар сөздер болып табылатын берілген функция атауы бар сөйлем.

Әрбір сөйлем жеке жолда орналасып, нүктемен аяқталуы керек.

2.9. № 9-зертханлық жұмыс
Visual Prolog файл жүйесі

Бұл бөлімде Visual Prolog файл жүйесін және файлдармен айналып тұрған стандартты предикаттарды көрсетіледі. Енгізу /шығару қайта тағайындалумен – әртүрлі құрылымдарымен енгізу /шығару тиімді байланыс әдісімен таныстыру.

Файл жүйесі ептеген ерекшеліктерімен Visual Prolog әртүрлі нұсқаларында бірдей жұмыс істейді

Visual Prolog *current_read_device* (оқығанның ағымды құрылымы), оған енгізу оқылады және жазғанның ағымды құрылымы *current_write_ device*, (одан шығару жіберіледі) қолданылады.

Әдеттегідей, оқығанның ағымдағы құрылымы пернетақта болып табылады, ал ағымдағы жазу құрылымы – дисплейдің пердесі. Дегенмен, басқа құрылымдарды тағайындай аласыз.

Мысалы, енгізу сыртқы жадта сақталған (мүмкін дискте) файлдан оқыла алады. Тіпті мүмкін программаны орындағанда ағымдағы енгізу мен шығару құрылымдарды қайта анықтауға болады. Оқудың және жазудың құрылымдарына қолданылғанына қарамай, Visual Prolog программасында оқу мен жазу ұқсас етіледі. Файлға оны ашу керек. Файл:

- оқу үшін;
- жазу үшін;
- қосу үшін;
- модификация үшін ашылады.

Кез келген әсер ету үшін ашық файл, операция аяқтаудан кейін жабылуы керек. Бір уақытта бірнеше файлдарды ашуға болады.

Ашық файлдардың арасында енгізу мен шығаруы белгіленеді.

Файлдарды ашу мен жабу деректердің ағындарын қайта тағайындауға қарағанда әлдеқайда көп уақытты алады.

Visual Prolog файл ашқан жатқан кезде, ол операциялық жүйенің файлдың бейнелі түрде атын нақты атымен байланыстырады және осы бейнелі түрде аты енгізу мен шығаруға бағыттылынады.

Файлдардың бейнелі түрде аттары кішкентай әріптен басталуға тиісті және file доменды сипатталғанда хабарландыруға тиісті. Мысалы: file = file1; source; auxiliary; somethingElse

Кез келген программада тек қана бір file домен рұқсат берілген.
1 кестеде көрсетілген бес кірістірілген file альтернативтерді Visual Prolog таниды.

1- кесте. File доменның кірістірілген альтернативтер

Альтернатива	**Сипаттама**
keyboard	Пернетақтада оқу (Үндемеу бойынша)
screen	Мониторға жазу
stdin	Стандарттық енгізуден оқу
stdout	Стандарттық шығуын жазу
stderr	Қателерді шығару үшін стандарттық құрылымға жазу

Файлдарды ашу және жабу

Файлдарды ашу мен жабу үшін стандарттық предикаттар туралы келесі бөлімдер сипаталады

Ескерту

Файлды ашқанда кері слэш (\), дискінің подкаталогың белгілеу үшін қолданылады. DOS-бағдарланған болжамдарда Visual Prolog ESC-символ (бағдарлаушысымен) келеді. Сондықтан программада файлға белгіленген жолды екі (\\)кері слэш әрдайым көрсету маңызды.

Мысалы:

"c:\\prolog\\include\\iodec1.con"

файлға рұқсаттың жолын ұсынады:

c:\prolog\include\iodec1.con

Предикат *openread/2*

Предикат openread оқу үшін OSFileName файл ашады, келесі формат қолданылады:

openread(SymbolicFileName, OSFileName) % (i, i)

Visual Prolog file домен жарияланған, SymbolicFileName бейнелі түрде аты бойынша ашық файлға айналады. Егер файл ашылмаса, Visual Prolog қателік туралы хабарлайды.

Предикат *openwrite/2*

Предикат openwrite жазу үшін OSFileName файл ашады, келсі формат қолдана:

openwrite(SymbolicFileName, OSFileName) % (i, i)

Егер файл болса, онда оны жояды. Басқа жағдайда Visual Prolog жаңа файлды құрады және оның тиісті тізбесінде сыйғызып салады. Егер файл жасалмаса, Visual Prolog қателік туралы хабар береді.

Предикат *openappend/2*

Предикат openappend файлдың соңына жазылса OSFileName файлды ашады.
Сонымен бірге келесі формат қолданылады:
openappend(SymbolicFileName, OSFileName) % (i, i)

Егер файл жазуға ашыла алмаса, Visual Prolog қателік туралы хабарлайды.

Предикат *openmodify/2*

Предикат openmodify жазу мен оқу үшін OSFileName ашады, Егер файл болса, ол қайта жазылмайды, openmodify келесі формат:
openmodify (SymbolicFileName, OSFileName) % (i, i)
Егер жүйе OSFileName аша алмаса, Visual Prolog қателік туралы хабарландырады. Openmodify

Предикат кез келген рұқсаты бар файлының толтырулары үшін стандарттық filepos предикатпен бірге қолданыла алады.

Предикат *filemode/2*

предикат filemode мәтіндік режимде файлды ашқан кезде предикат filemode мәтіндік немесе екілік режимде көрсетілген файлды орнатады, келесі формат қолдана:
filemode(SymbolicFileName, FileMode) % (i, i)
Егер FileMode = 0, SymbolicFileName файл екілік режимге келеді
егер FileMode = 1, онда ол мәтіндік режимге келеді.

Мәтіндік режимде жазғанда жана жолдарға "Келесі жолға өту" / "Енгізу пернесі" символдар қосылады, ал оқығанда бұл екі символ жаңа жол сияқты түсіндіріледі
Carriage return (возврат каретки) = ASCII 13
Line feed (перевод строки) = ASCII 10

Екілік режимде ешқандай өзгерістер болмайды

Екілік файлды оқу үшін тек қана readchar немесе екілік файлдарға рұқсат предикаттар қолданады.

Предикат *closefile/1*

closefile предикат белгіленген файлды жабады, ол келесі форматты қолданады:
closefile(SymbolicFileName) % (i)

Предикат *readdevice/1*

readdevice предикат readdevice current_read_device (оқудың ағымды құрылымы) қайта анықтайды немесе оған атын қайтарады.
Предикаттың форматы
readdevice(SymbolicFileName) % (i), (o)

Предикат *writedevice/1*

Егер SymbolicFileName айнымалысы анықталса және файл оқуға ашық болса, readdevice предикат оқудың ағымды құрылымын қайта анықтайды. Егер SymbolicFileName айнымалысы бос болса, онда readdevice оқудың белсенді ағымды құрылымының атын береді.

Предикат *writedevice/1*

writedevice предикат не тағайындайды немесе *current write_device* (жазудың ағымды құрылымы) ат алуға рұқсат береді. Оның форматы
writedevice(SymbolicFileName) % (i), (o)

Егер көрсетілген файл жазғанға немесе қосқанға ашылса, writedevice предикат жазу құрылымын қайта анықтайды. Егер SymbolicFileName айнымалысы бос болса, онда writedevice жазудың белсенді ағымды құрылымның атын береді.

Файлды жабу, файлға жазу және файлды ашу үлгілері

1. Келесі тізбекті жазу үшін MYDATA.FILнің файлды ашады, writedevicenің екі предикаттарының арасындағы операторлармен, содан соң барлық шығарудың MYDATA.FIL-нің файлға бұл destinationнің file доменінің сипаттама пайда болатын бейнелі түрде атына сәйкес келетіне бағыттайды, мысалы:

```
domains
file = destination
goal
openwrite(destination, "MYDATA.FIL"),
writedevice(OldOut), % Шығарудың ағымды құрылымын аламыз
writedevice(destination), % Файлға шығаруды қайта жібереміз
writedevice(OldOut), % Шығарудың ағымды құрылымын қалпына келтіреміз.
```

2. Программа ch2e09.pro (листинг 1) символдарды тіркелген клавиатурада TRYFILE.ONE файлға дисктегіні салады, read және write стандарттық предикаттарды қолданады.

Тіркелген символдар дисплейдің экранына шығарылмайды. Сіздере жақсы жаттығу болады, егер осы символдарды экранға шығару программаны жазғанда. Клавишаны басқанда файл жабылады.

Программа ch12e09.pro

```
domains
  file = myfile

predicates
  readloop - procedure ()
  run - procedure ()

clauses
  readloop:-
        readchar(X),
        X<>'#',!,
        write(X),
        readloop.
```

```
 readloop.

 run:-
        write("This program reads your input and writes it to"),nl,
        write("tryfile.one\n"),
        write("For stop press #"),nl,
        openwrite(myfile,"tryfile.one"),
        writedevice(myfile),
        readloop,
        closefile(myfile),
        writedevice(screen),
        write("Your input has been transferred to the file tryfile.one"),nl.

goal
 run.
```

Стандарттық енгізу/шығаруды қайта анықтау

file доменда stdin, stdout, stderr үш қосымша опциялар бар. Бұл файлдық ағындарының артықшылығы келесіде: сіз командалық жолында стандарттык енгізу / шығаруды қайта тағайындауға болады (кесте2)

2-кесте
Файлдық ағындар мен сипаттамалар

- stdin Стандартты енгізу файлы болып табылады. Ол тек оқуға арналған. Оны (stdin) readdevice пернетақтада stdin енгізеді.
- Stdout – жазуға арналған Стандартты шығару файлы. Оны (stdout) writedevice терминалының пернесі бойынша stdout енгізу құрылымымен тағайындайды.
- stderr бұл жазудағы қателерді шығаруға арналған стандартты файл. Бұл терминалдың экраны Writedevice (stderr) қателерді шығару үшін stderr. құрылымын тағайындайды.

Файлдармен жұмыс істеу

Бұл бөлімде файлдармен жұмыс істеу үшін басқа бірнеше предикаттарды сипаттаймыз Олар: filepos, eof, flush, existfile, deletefile, renamefile, disk және copyfile.

Бірнеше мысал келтірейік:

1. Келесі тізбектне somefile.pro файлға Text мәні жазып алынады(myfile сияқты айналып жатыр), 100 позициядан бастап файлдын бастауы бойынша.

```
Text = "A text to be written in the file",

openmodify(myfile,   "somefile.pro"),
```

writedevice(myfile),
filepos(myfile, 100, 0),
write(Text),
closeflie(myfile).

Листинг программасында бұл жасалғандарды, байттың артында байт файлдың ішіндегісін қолданып filepos тексеруге болады. Бұл программа файлдың атын сұрайды, файлдың позициялардың ішіндегісін содан кейінгісін көрсетіп, позициялардың нөмерлері пернетақтамен енгізіліп отырады.

2 листинг. ch12e09.pro программасы

```
domains
  file = input

predicates
  inspect_positions(file) - determ (i)

clauses
  inspect_positions(UserInput):-
        readdevice(UserInput),
        nl,write("Position No? "),
        readln(X),
        term_str(ulong,Posn,X),
        readdevice(input),
        filepos(input,Posn,0),
        readchar(Y),nl,
        write("Char is: ",Y),
        inspect_positions(UserInput).

goal
  write("Which file do you want to work with ?"),nl,
  readln(FileName),
  openread(input, FileName),
  readdevice(UserInput),
  inspect_positions(UserInput).
```

Eof/1 предикаты

eof перидикатын тексеруі, процесте алған оқуларға файлдың соңымен позиция келеді. Eof перидикатының түрлері:.

eof(SymbolicFileName) % (i)

Егер файл құқықтармен орындаулар уақыты тек қана жазуға ашық болса қатені eof береді. Предикат (<Ctrl> + <Z> комбинациялы пернелер) DOS файлдың аяқталуы ерекше мән туғызбайды.

Предикат пайдалы файлдармен жұмыс істеуінде предикатты анықтау үшін қолдануға пайдалы, мысалы, файлдың аяғына жете болмайтын кезде Bile нүктені сол мезгілдерге дейін қайтарады.
Предикат repfile predicates файлдармен жұмыста

```
repfile(FILE)
clauses
repfile(_).
repfile(F):-not (eof (F)), repfile(F).
```

Келесі программада біреу файлды өзгертеді, барлық әріптер бастапқы болып келеді. Листинг 3. Программа ch12e11.pro

```
omains
  file = input; output
predicates
  convert_file - procedure ()
  repfile(FILE) - nondeterm (i)
  run - determ ()
clauses
  convert_file :-
        repfile(input),
        readln(Ln),
        upper_lower(LnInUpper,Ln),          /* converts the string to uppercase */
        write(LnInUpper),nl,
        fail.
  convert_file.
  repfile(_).
  repfile(F):-
        not(eof(F)),
        repfile(F).
  run:-
        write("Which file do you want convert ?"),
        readln(InputFileName),nl,
        write("What is the name of the output file ?"),
        readln(OutputFileName),nl,
        openread(input, InputFileName),
        readdevice(input),
        openwrite(output, OutputFileName),
        writedevice(output),
        convert_file,
        closefile(input),
        closefile(output).
goal
  run.
```

Предикат *flush/1*

Передикат аталған файлға ішкі буферлер flush ішіндегісін жазып отырады.

flush(SymbolicFileName) % (i)

Ол буфер «барлық құлату жүйесін» сұрап жатыр.

Предикат *existtile/1*

Егер передикат existfile программасын сәтті орындаса, онда OSFileName файлы табылады. Предикат OSFileName тізбесі бола алады, ал сияқты, \psys\ *.cfg аты алмастырулар таңбалары бола алады: Егер файлдың аты тізбекте белгі қойылған жолда емес предикат жетіспеушілігінің existfile де болады. Бірақ, «system» (Жүйелік) қойылған атрибуттармен барлық файлдарды, файлдарды қоса existfile де табады, және "hidden" де (Бүркеме), ол тізбелерді таба алмақ. Суреттеп айтылған төмендегі тізбелерді іздестірулер предикаттардың қолдануымен істелінген болу мүмкін.

Тексеру үшін файл дискіде болған жағдайда ғана (оны ашпастан алдын) пайдалана алуға болады open(File, Name) :-

existfile(Name),

openread(File, Name).

open(_, Name) :-

write("Error: the file ", Name," is not found").

Предикат *searchfile/3*

Предикат тізім жолдарында файлды табу үшін searchfile қолданылады.

searchfile(PathList, Name, FoundName) % (i, i, o)Егер дискке түбірде autoexec.bat орналасқан:, С тең FoundName орнайды : \AUTOEXEC.BAT.

Файлдың аты алмастырулар бола алады. Бұл жағдайда алмастырулар болатын файлдан толық атымен сабақтас FoundName болады, және төменде суреттеп айтылған тізбелер іздестіру предикаттар үшін дәлелді сапада қолданылуы мүмкін болуы керек. Егер алдыңғы мысалдың файлдың атының атап қалған *.bat орнына мысалы autoexec.bat, сабақтас FoundName көрстсе болады: \ *.BAT.

Предикат *deletefile /1*

Предикат deletefile оның берілген аргументтері мен формалары мен берілген файлдарфн өшіріп тастайды. deletefile(OSFileName) % (1)

Егер файл алып тастай алмаса предикат қатені көрсетеді. OSFileName біржолдық нышандар бола алмайды.

Предикат *renamefile/1*

Предикат renamefile NewOSFileName атынан OldOSFileName атын өзгертіп жатыр. Ол өз формасын алады.

renamefile(OldOSFileName, NewOSFileName) % (i, i)

Егер NewOSFileName атымен файл болмаса предикат табысты renamefile болып қалады, және екеуі де файлдық өзгерген аттармен сақталып қалады.

Предикат *disk/1*

Предикат ағымдағы диск өзгеруі үшін disk қолданады немесе каталогта/подкаталота өз қалыпын алады :
disk(Path) % (i), (o)

Шақыруда параметрге, ағымдағы тізбеге disk қайтарылады. Бұл дискте қазіргі ағымдағы тізбесіз өзгеріссіз басқа дискке ауыстырып қосу үшін DOS-Бағдарлаған болжамдарда D қолданылады:. Бұл жерде D – құрылым таңбалаушы әріп.

Предикат *copyfile/2*

Предикатта файлдың көшірмесін алу үшін copyfile қолданылады. Ол екі параметрді қабылдайды:.copyfile(SourceName, DestinationName) % (i, i)

Бұл файлдарға жолдар дисктер және тізбелерді қоса алғанда файлдардың аттары толық немесе ішінара беріле алады. Қайта құрылмалы нышандарға тыйым салынған. Көшіріп алған атрибуттарын және құқықтарын бастапқы файл алады.

2.10. № 10-зертханлық жұмыс
Visual-ға кіріспе ішкі деректер базасы

Іішкі деректер базасы (internal fact databases) программаның facts бөліміндегі ішкі деректер қоры және predicates бөлімінде сипатталған предикаттардың қолданылуы жарияланады.

Деректер қорына жаңа айғақтарының қосымшалары үшін Visual Prolog assert, asserta, assertz-дың предикаттарын қолданады, retract және retractall-дың предикаттары қазіргі айғақтарды алып тастауы үшін қызмет көрсетеді. Consult/1 және consult/2 предикаттар файлдан айғақтарын оқиды және олардың ішкі деректер қорын, save/1 және save/2-леріне файлдағы ішкі деректер базаларын сақтайды.

Visual Prolog деректер қорына, кәдімгі предикат с тәуелді болуы фактілерді түсіндіреді. Ішкі деректер базасы предикаттарының айғақтары оңай өзгертуге болатын кестеде максимал жылдамдығының табысы кәдімгі предикаттар екілік кодтарға құрастырылады және сақталады.

Ішкі деректер базасының жариялануы (бұл database-ның сөзінің синонимы) facts маңызды сөз factstың бөлімді тануын анықтайды. Facts бөлім тиісті ішкі деректер база суреттейтін предикаттарды тізбектен тұрады. Орындаулар уақытында assertалары предикаттары арқылы және assertz деректер базасына (бірақ ереже емес) айғақтар толықтыруға болады. Немесе,

дисктегі файлдан толықтыратын айғақтары consult-тың үйреншікті предикатын шақырып алуға болады.

domainsname, address = stringage = integergender = male femalefacts(name, address, age, gender) person
predicates(name, address, age) male
(name, address, age) female
(name, age, gender) child
clauses(Name, Address, Age) male:-
(Name, Address, Age, male) person.

Мысалы, бұл (male, female, child) басқа предикаттарды қолданатындай осы сияқты person предикатын пайдалануға болады. Person программасының предикаты үшін айғақтар жұмыс уақытында толықтырып алып тастай алатын оның ерекшелігі.

Айғақтардың бөлім жариялалған предикаттарға келесі екі шектеу бар:

- деректер қорын тек айғақтармен толықтыруға рұқсат етіледі, бірақ олар ереже емес;
- базаның айғақтары еркін айнымалы бола алмайды.

Facts-тың бірнеше бөлімдерінің бар болуына рұқсат етіледі, бірақ ол үшін facts-тың әрбір бөлімінің аты анық көрсетілуі керек.

(integer) myFirstRelation
mySecondRelationfreal, string
(string) myThirdRelation
/* etc. */

Mydatabase-ның аты бар facts бөлімінің сипаттамасы mydatabase-ның аты бар айғақтардың деректер қорын құрады. Егер сіз ішкі деректер базасына ат қоймасаңыз, онда ол үндемеу бойыншаға dbasedom-ның үйреншікті атын тағайындайды. Ол егер тек қана жобаның бір бөлігі жарияламайтын модульдан тұратын программа айғақтардың жергілікті атаусыз бөлімдері бола алады

Деректер базасы предикаттарының аттары сирек кездесетін (бастапқы файл) модуль болуы керек; facts-та екі әртүрлі тараулар қолдануға болмайды. Facts және predicates-тің тарауларына предикаттардың бірдей аттарын қолдануға болмайды. Тараулар жергілікті facts-тың нақты предикаттарының аттары, олар жариялайтын предикаттар/фактілер модуль үшін жергілікті болып табылады.

Ішкі деректер базаларын қолдану

Ішкі деректер базасы

Ішкі деректер база тәуелді предикаттар туралы түсінік, сонымен қатар басқа предикаттар. Жалғыз көрнекті айырмашылық мұндай предикаттардың хабарлауы predicatestің бөлімі орынына factstың бөлімінде орналастырған тұрады:

```
domains
name = string
sex = char
facts
person(name, sex)
clauses
person("Helen", 'F').
person("Maggie", 'F').
person("Suzanne", 'F').
person("Per", 'M').
```

сіз ("Maggie \" \, \ F \) person-нің барлық әйелдерді табу үшін (Name, \ F \) немесе Maggie-нің аты бойынша әйел сіздің деректер қорыңызға бар болатын тексеру үшін person-ның мақсаты бар person шақыра аласыз.

Ішкі деректер базасының жаңартылуы

Visual Prolog айғақтары бар жұмыс үшін үйреншікті предикаттар: assert, asserts, assertz, retract, retractall, consult және save – бір немесе екі дәлелдерді иемдене алады. Міндетті емес екінші дәлел ішкі деректер базасының аты болады.

/1 және /2 белгі предикаттың осы болжамы үшін дәлелдердің керек санын предикаттың әрбір атынан кейін көрсетеді. ((i) */ және (o, i) */) /*) сондай /*лар) түсініктер предикат ол үшін параметрлердің (және) ағындарының көрсетеді.

Программаның орындауы айғақтардың енгізуі уақытында болады.

Айғақтар орындаулар уақытында предикаттардың арқылылардың айғақтарының ішкі деректер қорына қосыла алады: assert, assertалар және assertz, немесе файлдан айғақтардың жүктеуі жолымен consulttар арқылы.

Орындау бір айғақтың қосымшасы үшін үш предикат уақытында бар болады:

```
asserta(<the fact>)                          % (i)
asserta(<the fact>,  facts_sectionName)      % (i,  i)
assertz(<the fact>)                          % (i)
assertz(<the fact>,  facts_sectionName)      % (i,  i)
assert(<the fact>)                           % (i)
assert(<the fact>,  facts_sectionName)       % (i,  i)
```

Asserta предикат осы предикат, assertz үшін бар айғақтардың алдында айғақтарының деректер қорына жаңа айғаққа қыстыртады айғақтар осы предикаттың бар айғақтарынан кейін қыстыртады. Қолдануы assertтың предикаттары нәтижесін береді, assertz-дың қолдану ыңғайлы болады.

Деректер базасы предикаттарының аттары болғандықтан программаның іші сирек кездесетін немесе (жергілікті айғақтардың бөлімдері үшін) модуль, asserta және айғақтардың деректер қорына айғақ толықтыруы керек болатын assertz әрдайым белгілі. Дегенмен түрдің тексеруі

айғақтардың деректер қор тиісті мақсаттарындағымен жұмысын міндетті емес екінші дәлелді қолдануға боладуға қамтамасыз етілу үшін сол үшін.

Ғибраттанушыны бірінші предикат Suzanne туралы personның предикат суреттеп айтылған айғақ personның қазіргі мезет жад сақталған барлық айғақтарынан кейін қояды. Екінші - personның предикатының барлық бар айғақтарының алдында Michael туралы айғақ. үшінші - John туралы айғақ likesDatabaseнің айғақтарының деректер қорына likestің барлық басқа айғақтарынан кейін, төртінші Shannon туралы айғақты қояды.

```
assertz(person("Suzanne", "New Haven", 35)).
asserta(person("Michael", "New York", 26)).
assertz(likes("John",   "money"),   likesDatabase).
asserta(likes("Shannon",   "hard work"),   likesDatabase).
```

Деректер базасының бұл предикаттардың шақыруынан кейін басталған жұмыссыз келесі айғақтар сияқты:

```
% Ішкі деректер базасы - dbasedom
("Michael \" \, "New York \" \, 26) person.
%.0....   person.... басқа айғақтар.
("Suzanne \" \, "New Haven \" \, 35) person.
% Ішкі деректер базасы - likesDatabase
("Shannon \" \, "Hard work \" \) likes.
%.0....  likes.. басқа айғақтар.
("John \" \, "Money \" \) likes.
```

Ылғи бір айғақ екі рет бекитін кодқа жазыңыз. Ішкісі деректер базалары ешқандай да қайталанбаушылықтарды ескермейді, сондықтан ылғи бір айғақ айғақтардың ішкі деректер қорында көрініп қала алады. Қайталанбаушылыққа тексеруі бар asserttың болжамын жазу:

```
facts – people
person(string, string)
predicates
uassert(people)
clauses
uassert(person(Name, Address)):-
person(Name, Address),
!
; % OR
assert(person(Name, Address)).
```

Файлдан айғақтарының оқуы:

Consult предикат facts бөлімінде суреттеп айтылған айғақтарды fileName файлынан оқиды және олардың соңында тиісті деректер базасының программасын қыстырады. Consult предикат бір немесе екі дәлелді алады:

```
consult(fileName)                    %   (i)
```

consult(fileName, databaseName) % (i, i)

Егер тек қана (деректер базасының атысыз) бір дәлелмен consult шақырсақ, дегенмен assertz-дарға қарағанда, онда тек қана бөлімде (dbasedomның үндемеу бойыншасына) атсыз сипатталған айғақтар саналады.

Егер сіз (деректер базасының файл аты және аты) екі дәлелдері бар consult шақырсаңыз, онда көрсетілген деректер базасынан тек қана айғақтары тексеріледі. Егер файл бірдеңені көрсетілген базасының айғақтарынан басқа әлі болса, онда ол бұл жолға дейін жеткенде consulttің предикаты қатені қайтарады.

Consulttың предикаты айғаққа бір-бірдендері оқитынына қарау керек. Егер файл он айғақтарда болса, жетінші айғақта қандай болмасын синтаксистік қателік, consultte болады айғақтардың деректер қорына алты бірінші айғақтарын енгізіледі, кейін қателік туралы хабар береді.

Consult-тың предикаты тек қана сол save-ты құратын қалыптағы файлдары оқи алатынын атап өту керек. Файлдар келесідей болуы керек:

- екі тырнақшалардағы жолдары ішінде отыратын жоғарғы регистрдің нышандары, ерекшелікке;
- жолдар екі есе шығын ішінде отыратын кемшіліктер, ерекшелікке;
- түсініктер;
- бос жолдар;
- (symbol) идентификаторлар екі тырнақшаларсыз.

Әскерилік Аққу редакторда жасау немесе айғақтары бар файлдың өзгерісінде сақтауы керек.

Программаның орындауы айғақтардың алып тастауы уақытында

Retract предикат айғақтар бір ізге салады және олардың ішкі деректер базасынан алып тастайды. Ол келесі түрі:

retract(<the fact>) %(i)

retract(<the fact>, databaseName) % (i, i)

Retract предикат <the fact > айғақпен программаның орындалуы <the fact > еркін айнымалы уақытында ұластыра дәл келетін деректер қоры бірінші айғақты алып тастайды. Бір ізге салынған айғақтардың алып тастауын қосымша эффектпен оған рұқсат туралы ішкі деректер базасынан айғақтарының алып тастауы баламалы. Егер деректер базасының retract алып тасталатын предикат болса, retract детерминация жасамалған, жариялы детерминделген болып табылады. Retract предикат іздестіруде барлық бір ізге салынған айғақтар алып тастайды, кейін керек айғақтар көп табылмайды.

Программа facts-тың келесі бөлімдерінде болады делік:

facts

person(string, string, integer)

facts – likesDatabase

likes(string, string)

dislikes(string, string)

clauses

```
person("Fred", "Capitola", 35).
person("Fred", "Omaha", 37).
person("Michael", "Brooklyn", 26).

likes("John", "money").
Likes("Jane", "money").
likes("Chris", "chocolate").
likes("John", "broccoli").

dislikes("Fred", "broccoli").
dislikes("Michael", "beer").
```

Visual Prolog келесі мақсаттар facts мұндай бөлімдері бола беруге болады:

```
retract(person("Fred", _,_)),          %   1
retract(likes(_, "broccoli")),         %   2
retract(likes(_, "money"), likesDatabase),%   3
retract(person("Fred", _, _), likesDatabase).   %   4
```

Бірінші мақсат dbasedom-ның деректер базасынан Fred туралы person-ның бірінші айғағын алып тастайды. LikesDatabase-нің деректер базасынан екінші мақсатты ("Broccoli \" X, \) likes-пен беретін бірінші айғақ алып тасталады. Деректер базасы предикаттарының аттары болғандықтан сирек кездесетін жағдайда Visual Prolog базасынан алып тастау керектігін біледі, ал likes деректер қорында тек қана likesDatabaseнің базасы болады.

үшінші және төртінші мақсаттардың қалай сіз екінші дәлелдің түрдің тексеруі үшін қолданыла алатындығын көрсетеді. үшінші мақсатты ойдағыдай жүзеге асырылады, likesDatabase ("Money \" _, \) likesімен бірінші айғақ алып тасталады , төртінші мақсат қатені береді, өйткені likesDatabaseнің айғақтарының деректер қорына personның айғағы (және бола алмайды) жоқ. Хабар төмендегіше көрінеді:

```
506 Type error: The functor does not belong to the domain.
```

Түрдің қатесі: Функтор осы доменге жатпайды

Retract предикатынан келесі мақсатты қалай ала алатыны келтіріледі:

```
goal
retract(person(Name, Age)),
write(Name, ", ", Age), nl,
fail.
```

Ретінде екінші retractтың дәлелі қашан деректер базасының аты, тапсырма беру айғақтар алып тастайтын деректер базасының предикаты нұсқауы мүмкін. Retract осы жағдайда барлық айғақтары іздеп алып тастай деректер қор көрсетілген. Мысалға:

```
goal
retract(X, mydatabase),
write(X),
fail.
```

Бірнеше айғақтардың алып тастауы бірден

Retractall предикат <the fact > туралы дәл келетін барлық айғақтарды деректер базасынан алып тастайды. Retractall предикатының келесі қалыбы болады:

retractall(<the fact>)
retractall(<the fact>, databaseName)

Тап қалған әсерге retractall әсер сол сияқты

retractall(X):- retract(X), fail.
retractall(_).
оны бірақ едәуір тез.

Retractall предикатты әрдайым ойдағыдай сақтайтыны белгілі. Осылайша, assert және retract предикаттары жағдайында, түрдің тексеру үшін екінші дәлелді қолдануға болады. Егер retractall-дың шақыруы астын сызу символы қолданылса және retract предикатының жағдайы болса, онда facts-тың көрсетілген бөлімінен барлық айғақтарды алып тастауға болады.

Келесі мақсат person-ның айғақтары бар деректер базасынан еркектер туралы барлық айғақтарды алып тастайды
retractall(_, mydatabase).

Келесі мақсат mydatabase базасынан барлық айғақтарды алып тастайды
retractall(person(_, _, _, male)).

Determ негізгі сөзімен айтқанда жариялалған айғақтар

Determ маңызды сөз деректер базасы осы қорытып айтқанда жариялалған деректер базасының предикаты үшін айғақ аспайтын бір бола aeterm Ключевоемен қорытып айтқанда жариялалған айғақтар.

Determ маңызды сөз деректер базасы осы Ключевоеден қорытып айтқанда жариялалған деректер базасының предикаты үшін айғақ аспайтын бір го бола алатынын анықтайды.

Детерминделген компиляторға айғақты тану тиімді кодтан астам шығаруға мүмкінлатынын анықтайды. Егер программа кіріспе деректер қорына екінші мұндай айғақ орнатуға тырысады қатені шығарады. Демек, ерекше сақтықпен программашыға детерминделген айғақтар қолдануы керек.

Determ жариялайтын айғақтың алып тастауында retract/1 және retract/2дің детерминация жасамалған предикаттарының шақыруы детерм болады. Деректер базасының уақыты кез келген уақытта counterда бір айғақтан аспайды, сондықтан:

```
facts
determ counter(integer CounterValue)
goal
…
retract(counter(CurrentCount)),  % Пролог не установит точку отката
Count= CurrentCount +1,
assert(counter(Count)),
```

вместо
facts
counter(integer CounterValue)
predicates
determ retract_d(dbasedom)
clauses
retract_d(X): - retract(X), !. % детерминированный предикат
goal
retract_d(counter(CurrentCount)), % Пролог не установит точку отката
Count= CurrentCount + 1,
asserta(counter(Count)),

Single негізігі сөзімен айтқанда жарияланған айғақтар

Single маңызды сөз деректер базасы single Ключевоемен қорытып айтқанда жариялалған деректер базасының предикаты үшін бір-ақ айғақта әрдайым болатынын анықтайды.

Айғақтар (бір рет) single сондықтан программа мақсат шақыратында белгілі болуы керек; демек, олар программаның бастапқы кодындағы clauses-тың тараулардасына аты-жөнін көрсетуі керек. Мысалы:

```
facts - properties
single numberWindows_s(integer)
clauses
numberWindows_s(0).
```

Бір рет айғақтар алып тастала алмайды. Егер бір рет компилятор айғақты алып тастаса қате шығарылады.

Егер ол еркін дәлелдермен шақырса бір рет айғақтың біреуі өйткені әрдайым бар болады, бір рет айғақтың шақыруы ешқашан бітпейді.

Мысалы, келесі шақыру:

numberWindows_s (Num) ,

егер Num-еркін айнымалы болса жүні жығылумен ешқашан бітпейді. Демек, procedureның детерминизмінің түрімен жариялалған предикаттардағы бір рет айғақтарды ыңғайлы қолданылады.

Assert, asserta, assertz және consulttің singlenің айғақ қолданылған предикаттары retract және assertтің предикаттарының булары сол сияқты жұмыс істейді. (consult) assert предикаттар атап айтқанда айғақтың қазіргі данасы көрсетілгенге жаңа өзгертеді.

Компиляторға айғақтың декларациясының алдында single сөздер қолдану Негізгі сөздер бір рет айғаққа рұқсат және оның түрлендіруі үшін оптимизациялалған кодты жасауға мүмкіндік береді. Мысалы, компилятордың assert-тың бір рет айғақ қолданылған предикаттары үшін retract және assert-тің детерминделген (retract және assert предикаттары жұп ретінде қолдануда (детерминация жасамалған) айғақпен) кәдімгі айғақ қолданылған предикаттары жұпқа қарағанда тиімдірек жұмыс істейтін код шығарады.

Бос емес кейбір домендерді бір рет айғақтайды

```
global domains
font = binary
facts - properties
single my_font(font)
clauses
my_font($[00] )
```

Басқа маңызды ерекше жағдай reftың үйреншікті домен болатын бір рет айғақтардың инициализациясы болып табылады. Ref домен сыртқы деректер қорларындағы сілтеме сандары үшін домен Visual Prolog болып табылады, бірақ сонымен бірге ол Visual Prolog берілетін бума жариялалған көпшілігінде алдын ала анықталған домендерді қолданылады. Мысалы, VPI window-ші негізгі домен осылай жариялаған:

```
domains
window = ref
```

Ref домені мәндерінің инициализациясы үшін ()) ерекше алдыңғы нышаны бар таңбасыз сандар қолданылады. Мысалы:

```
facts
single mywin(WINDOW)
clauses
    mywin(~0).
```

Ішкі деректер базасының қолдануының мысалдары.

1. Листинг151) ch08e01.pro программа – ішкі деректер базасы көмегімен сарапшылық жүйені жазатын тұрып қал. Мысал бұндағы деректер базасының қолданулары маңызды артықшылығы программа (және олар алып тастау) өнер-білім ЄдЄ толықтыра алатын болып табылады.

Ch08e01.pro Листинг151 программа.

```
domains
  thing = string
  conds = cond*
  cond  = string

facts
  is_a(thing,thing,conds)
  type_of(thing,thing,conds)
  false(cond)

predicates
  run(thing) - nondeterm (i)
  ask(conds) - nondeterm (i)
  update - procedure ()
```

```
clauses
  run(Item):-
        is_a(X,Item,List),
        ask(List),
        type_of(ANS,X,List2),
        ask(List2),
        write("The ", Item," you need is (a/an) ", Ans),nl.

  run(_):-
        write("This program does not have enough "),
        write("data to draw any conclusions."),
        nl.

  ask([]).
  ask([H|T]):-
        not(false(H)),
        write("Does this thing help you to "),
        write(H," (enter y/n)"),
        readchar(Ans), nl, Ans='y',
        ask(T).

  ask([H|_]):-
        assertz(false(H)), fail.

  is_a("language","tool",["communicate"]).
  is_a("hammer","tool",["build a house","fix a fender","crack a nut"]).
  is_a("sewing_machine","tool",["make clothing","repair sails"]).
  is_a("plow","tool",["prepare fields","farm"]).

  type_of("english","language",["communicate with people"]).
  type_of("prolog","language",["communicate with a computer"]).

  update:-
        retractall(type_of("prolog","language",["communicate with a computer"])),
        asserta(type_of("PDC Prolog","language",
                              ["communicate with a personal computer"])),
        asserta(type_of("prolog","language",
                              ["communicate with a mainframe computer"])).

goal
  run("tool").
```

Келесі айғақтар бола алар ма еді asserta предикаты арқылы енгізілген немесе assertz, немесе файлдан consult предикаты арқылы саналған. Олар clauses бөлімінде орналастырған.

```
is_a("language","tool",["communicate"]).
```

```
  is_a("hammer","tool",["build a house","fix a fender","crack a nut"]).
  is_a("sewing_machine","tool",["make clothing","repair sails"]).
  is_a("plow","tool",["prepare fields","farm"]).

  type_of("english","language",["communicate with people"]).
  type_of("prolog","language",["communicate with a computer"]).
```

Мақсаттар ретінде:

```
goal
run (tool) .
```

Келесі мақсатты енді енгізу:

```
update,   run(tool).
```

Update предикат программаның бастапқы кодына қосылған, айғақ алып тастайды

```
type_of(prolog,   language,   ["communicate with a computer"]).
```

из внутренней базы фактов knowledgeBase и добавляет два новых факта в программу:

```
type_of(prolog,   language,
["communicate with a mainframe computer"])
type_of("Visual Prolog",   language,
["communicate with a personal computer"])
```

Оның аттары бар save/2 мәтіндік файл және деректер базасы предикаты шақыруы арқылы knowledgebase-тың айғақтарының мәліметтерінің мәтіндік файлындағы барлық базасыны сақтауға болады. Мысалы, шақырудан кейін

```
save("mydata.dba", knowledgeBase)
```

mydata.dba файл Visual Prolog, және әрбір айғақты ұқсас кәдімгі программаның clausestың бөліміне жеке жолда жазып алады. Бұл файлдан жадтағы айғақтарды consult предикаты арқылы санауға болады:

```
consult("mydata.dba",   knowledgeBase).
```

2. (factstың тарауларда суреттеп айтылған айғақтармен) деректер базасы предикаттар олар термдер болып табылады.

Айғақтардың базасын Visual Prolog ішкі домен, factstың бөлімінен тиісті айғақтарына шығарады. Мысалы:

```
предикаттар ол үшін домен

      (name, telno) person
      (cno, cname) city
```

Мұндай хабарлаулар, Visual Prolog-ші компилятор алынып шығарады dbal:ның домені

```
domains(name, telno) dbal = person; citytcno, cname
```

Қорытынды

- Visual Prolog ішкі деректер базасы facts программаларының айғақтарынан тұрады.

• Facts бөлімдерге (қай тиісті ішкі домендерді құрады) ат қоюға болады. Factstың атаусыз бөлімі үшін домен үнсіздік бойыншаға bdasedom-ның домені болады. Программада factstың бірнеше бөлімдері қатыса алады, бірақ сонымен бірге олардың әрқайсылары сирек кездесетін ат алуы мүмкін. Фактілер базасының предикаты facts бөлімшесін сипаттауға болады.

• Asserta, assertz және consulttің предикаттары көмегімен программа айғақтардың ішкі деректер қорына айғақтары жұмыс уақытында толықтыруға болады. Retracttары предикаттары арқылы және retractall программа бұл айғақтар жұмыс уақытында алып тастауға болады.

• Ішкі деректер базасыдан файлға save айғақтары предикат айғақтардың мұндай файлын редактор арқылы құрып немесе өңдеуге болады.

• Термдермен жұмыс істеу кезінде facts бөлімдерінің аттары шығарылған ішкі домендерде қолданылады.

Тапсырма:

1. Тиісті мәліметтердің анықтауы үшін ережені жазу. Әрбір ереже тек қана қойылған сұраққа керексіз мәліметтерді қайта санамай жауап беруі керек. Кітапханадағы кітаптардың тізімі. Әрбіреуін көрсету: реттік нөмір, автор, атау, бағаны, шығаруды жыл.

Ережелер арқылы анықтау:

- шығаруды бір жылдың кітаптарының атаулары.
- авторлар және 2003 жылдан кейін шығарылған кітаптардың атаулары.
- 2000 жылға дейін шығарылған кітаптардың бағасын.
- 1999 жыл шығарылған кітаптардың саны.
- осы автордың кітаптарының атаулары

ҚОРЫТЫНДЫ

Логика элементтерін ақпараттық технологиялар курсына енгізу тиісті қажетті іс болып табылады және ол аталған оқу құралының басты жетістігі болып есептеледі. Өйткені қазіргі заманғы білім берудің негізгі міндеттерінің бірі ретінде студенттердің логикалық дұрыс ойлауын дамыту аталады. Бүгінде адам санасының мүмкіндіктерін кеңейтетін танымның көптеген әртүрлі әдістері бар: үлгілеу және математикалық әдістер, соның ішінде ықтималдықтар теориясының әдістері, физикалық және биологиялық эксперименттер, ЭЕМ-да ақпарат өңдеу, т.б. Бірақ барлық осы әдістерді тиімді пайдалану үшін адамның ойлауы логикалық дұрыс болуы керек, сондықтан да ғылым ғана логикаға дұрыс ойлау заңдарын тануға үйретеді. Әрине, адам логиканың дәл ережелері мен заңдарын білмей-ақ, тек оларды жоғары деңгейде қолдана отырып дұрыс ойлай алады. Алайда логикаға ие адам анағұрлым дәл ойлайды, оның аргументациясы сенімдірек екенін де ұмытпау керек.

Логикалық ойлау туа бітетін қасиет емес, сондықтан оны әртүрлі әдістермен дамытуға болады. Логиканы жүйелі меңгеру – осы бағыттың анағұрлым тиімді жолдарының бірі.

Логикалық программаның идеялық тамыры математикалық логикада, математиктердің логикалық формуланы және формальді анықтау әдісін қолдануы, сипаттау бойынша автоматты түрде нәтиже алуға және есепті формальді сипаттау әдісін ашуға септігін тигізеді.

Пролог тілі программа құруға үйретуде кеңінен қолданылып, ойлау қабілетін жетілдіреді және құрылымдалмаған программа жазуға мүмкіндік береді.

Оқу құралында Visual Prolog логикалық программалау тілінің теориялық материалдары және зертханалық жұмыстары даярланған.

Пайдаланылған әдебиеттер тізімі:

1. Камардинов О. Жасанды интеллект, сараптаушы жүйелер, Пролог: оқу құралы. – Шымкент 2003
2. Каймин В.А. Информатика: учебник. – М.: ИНФРА-М, 2003.
3. Калягин В.О. Интеллектуальная собственность. – М.: НОРМА, 2000.
4. Eduardo Costa «Visual Prolog 7.1 for Tyros». Перевод с английского. – Перевод: И. Алексеев, Е.А. Ефимова, 2008, М. Сафронов, 2007. Редактор перевода: Е.А. Ефимова, 2008. Оформление: М.Сафронов, И. Алексеев. – 2007. – С. 174.
5. Хакимова Т. Ақпараттық технологиялар пәнінде жасанды интеллекті оқыту// Информатика негіздері Республикалық ғылыми-әдістемелік журнал. – №2 ,201.
6. Хакимова Т. Ақпараттық технологияларды оқытуда логикалық PROLOG тілін пайдалану // МИФ Республикалық оқу әдістемелік журнал. – №4, 27-29б. – Алматы, 2011.
7. Хакимова Т. Жасанды интеллекті іс жүзінде пайдалану // Информатика негіздері Республикалық ғылыми-әдістемелік журнал. – №1. – Алматы, 2012. – 9-13.