**The laboratory work 2**

Creating a project allows you to test an example as an autonomous executable program. After the project is run, an exe file is created for execution, the work of which is completed after the first solution that satisfies the solution of the problem. Running the program in this mode does not automatically output variable values, so you need to use the standard write output predicate.

**An example:**

The following relations and facts are created

Parent (“Ivan”, “Katya”)

Parent (“Ann”, “Oleg”)

Parent (“Oleg”, “Dima”)

Parent (“Igor”, “Olga”)

Parent (“Oleg”, “Victor”)

Parent (“Igor”, “Ivan”)

Man (“Dima”)

Man (“Ivan”)

Man (“Igor”)

Man (“Oleg”)

Man (“Victor”)

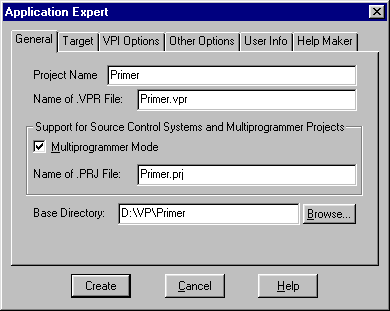
Woman (“Katya”)

Woman (“Olga”)

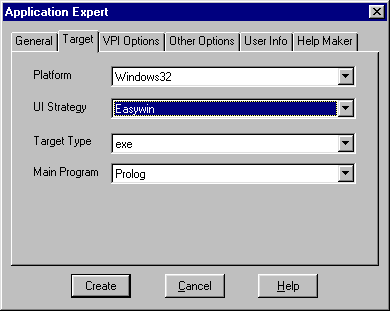
Woman (“Ann”)

Create a new predicate **grandfather (X, Y)** and determine who is the grandfather of Katya. Create a project and test the sample as an autonomous executable program.

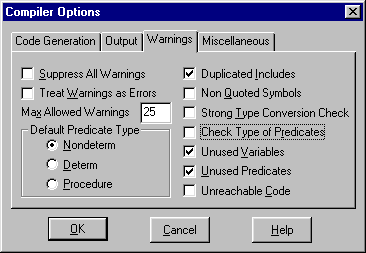
**The solution**  
1. Start the Visual Prolog environment and create a new project (Project | New Project), the Application Expert window is activated.  
2. Define the project name (Primer) and the base directory where the project will be saved (for example, D: \ VP \ Primer)



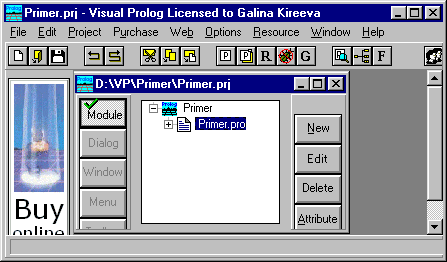
On the Target tab, set the parameters and click the Create button to create the project (Figure 5):



Open the Compiler Options window (Options | Project | Compiler Options), open the Warnings tab and set the compiler options for the created project (Fig. 6):



Click OK.  
4. In the project window, select the file Primer.pro and open it for editing (double click or Edit button)



The file with the .pro extension contains sections PREDICATES, GOAL, CLAUSES. Add the necessary definitions so that the program will turn out:

DOMAINS  
name = string  
PREDICATES  
parent (name, first name)  
woman (name)  
man (name)  
grandfather (name, first name)  
CLAUSES  
parent ("Ivan", "Katya").  
parent ("Ann", "Oleg").  
parent ("Oleg", "Dima").  
parent ("Igor", "Olga").  
parent ("Oleg", "Victor").  
parent ("Igor", "Ivan").  
man ("Dima").  
man ("Ivan").  
man ("Igor").  
man ("Oleg").  
man ("Victor").  
woman ("Katya").  
woman ("Olga").  
woman ("Anna").  
grandfather (X, Z): - parent (X, Y), parent (Y, Z), man (X).  
GOAL  
grandfather (X, "Katya"), write (X).

**Assignments**  
Modify the source code of the example as follows:  
1) add a new rule to the grandmother and determine who is the grandmother;  
2) add a new rule to the grandson and determine who is Anna's grandson;  
3) add a new rule to the brother and determine who Dima's brother is;  
4) add a new sister rule and determine who is Ivan's sister.

**Backtracking**

Search with a return (backtracking) is one of the main ways of finding solutions to the task in PROLOG. While performing a search, PROLOG can face the need to choose between alternative ways. Then he puts the marker at the junction point (the rollback point) and selects the first sub-goal. If it is not satisfied, then PROLOG returns to the rollback point and proceeds to the next subgoal.  
The Visual Prolog environment allows you to use the debugger to step through the program. The debugger works with the compiled code. In the source code, you can set breakpoints and execute the program in steps. In the step-by-step execution mode, you can view the values ​​of variables and the content of approved facts.

**Example**  
There is a database containing the facts of the type of **Rest** (name, city), **Kazakhstan** (city), **Russia** (city), **Baltic** (city). Make a rule that allows you to determine who rested in Russia.  
  Track the search for a solution to a problem using the Visual Prolog debugger and build a target search tree with a return.  
**The solution**:

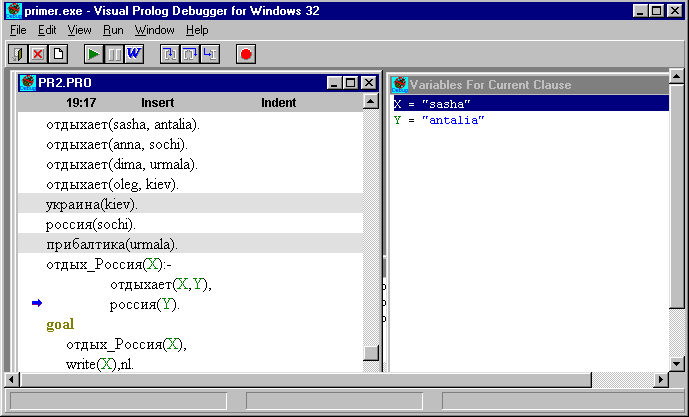
Track the search for a solution to a problem using the Visual Prolog debugger and build a target search tree with a return.  
**The solution**:  
1. Create a new project (Project | New Project) and type the text of the program:  
DOMAINS  
name, city = string  
PREDICATES  
Rest (name, city).  
Kazakhstan (city).  
Russia (city).  
Baltic (city).  
Rest\_Russia (name).  
CLAUSES  
Rest (“Sasha”, “Antalia”).  
Rest (“Ann”, “Sochi”).  
Rest (“Dima”, “Urmala”).  
Rest (“Oleg”, “Astana”).  
Kazakhstan (“Astana”).  
Russia (“Sochi”).  
Baltic (“Urmala”).

Rest\_Russia (X): - Rest (X, Y), Russia (Y).  
GOAL  
      Rest\_Russia (X),  
      write (X), nl.

3. Save the project (Project | Save Project)

4. Run it for execution (Project | Run, or the <F9> key, or the <R> button). The result of the program:  
Ann

5. Track the search for this solution with a debugger (Debugger). For this:  
a) run the debugger (Project | Debug);  
b. In the debugger window, select View | Local Variables (for viewing current values of variables);  
c) press the <F7> (or Run | Trace Into) key to step through the program, the current values of the variables are displayed in the Variables For Current Clause window



**Assignments**  
1. The database contains the following facts:  
fondOf ("Kolya", “guitar”).  
fondOf ("Olya", “violin”).  
fondOf ("Dima", “swimming”).  
fondOf ("Tanya", “tennis”).  
sport (“swimming”).  
sport (“tennis”).  
music\_inst (“violin”).  
music\_inst (“guitar”).  
a) make up the athlete's rule and determine who is keen on sports;  
b) trace the search for a solution using a debugger;  
c) construct a target search tree with a return.  
  
2. The database contains the following facts:  
fondOf ("Dima", “swimming”).  
fondOf ("Tanya", “tennis”).  
fondOf ("Kolya", “guitar”).  
fondOf ("Olya", “violin”).  
sport (“swimming”).  
sport (“tennis”).  
music\_inst (“violin”).  
music\_inst (“guitar”).  
a) make up a musician rule and determine who is fond of music;  
b) trace the search for a solution using a debugger;  
c) construct a target search tree with a return.