10. ПРОГРАМА НА VISUAL PROLOG

Въведение. Visual Prolog изисква величините, с които ще се работи, предварително да се описват. Това създава известни неудобства, но за сметка на това средата преобразува (компилира) програмата в изпълним от компютъра код, което прави изпълнението й много по-бързо. Друго удобство на подобен подход е възможността да се откриват своевременно многобройните грешки, с които е свързано писането на програми, особено от начинаещи програмисти.

В унисон с модерните среди за програмиране и средата Visual Prolog генерира автоматично по-голямата част от необходимите текстове на изходната програма и оставя за програмиста сравнително малко места за попълване.

Изходната програма на Visual Prolog включва четири основни секции:

- в секция **clauses** се поставят фактите и правилата;
- в секция **predicates** се декларират предикатите по следния начин:

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

- в секция **domains** се описват типовете на всички величини, с които трябва да работи програмата;
- в секция **goal** се поставя целта за програмата или списък от подцели.

Наред с тези задължителни секции в програмата често се използват и следните секции:

- секция **facts**, в която се включват онези факти, които бихте искали да могат да се променят динамично по време на изпълнение на програмата;
- секция **constants**, в която на отделни редове се подреждат необходимите символни константи по следния начин:

<уникално име> = <макродефиниция>

Когато е необходимо да се декларират данни, предикати или факти, които могат да се използват на всяко място от програмата, е необходимо в самото начало на програмата да се включат отделни секции за **global domains**, **global predicates** и **global facts**.

Постановка. В следващия текст стъпка по стъпка ще се доразвива проект на Visual Prolog 7.0. Трябва да се разполага с проекта **family2** на **Visual Prolog 7.0** и текстовия файл **fa.txt.** Началният проект е взаимстван от сайта www.visual-prolog.com.

1. Копирайте проекта в друга папка, за да се запази оригиналът. Компилирайте проекта. При компилирането се добавят редица пакети и файлове, необходими за работата на приложението.

2. Поставете файла **fa.txt** в папка **exe** на проекта, която се е създала автоматично при компилирането на проекта. Във файла fa.txt се съдържат факти за име и пол на няколко души и за името на техни родители.

3. Пробвайте програмата, като от менюто **file** заредете файла **fa.txt** и след това направете справка за бащата (father), дядото (grandFather) и прародителите (ancestor) на хората, записани в базата данни.

4. Допълнете файла **fa.txt** с факти от родословно дърво, например по следния начин:

```
person("Иван",male()).
person("Петър",male()).
person("Георги",male()).
person("Тодор",male()).
person("Петър2",male()).
person("Ивайло",male()).
person("Асен",male()).
person("Асен2",male()).
parent("Иван","Асен").
parent("Петър","Асен").
parent("Георги","Иван").
parent("Тодор","Иван").
parent("Петър2","Иван").
```

Двата предиката не бива да се смесват. Най-напред допълнете фактите за единия предикат, например за person, а след това - за parent.

Тези домейни (типове данни) са декларирани във файла TaskWindow.pro на програмата по следния начин:

```
    gender = female(); male().
class facts – familyDB
    person : (string Name, gender Gender)
    parent : (string Person, string Parent).
```

Обърнете внимание, че ще работим с база данни familyDB. Предикатите father, grandFather и ancestor са декларирани и дефинирани в същия файл TaskWindow.pro. Разгледайте файла и намерете тези дефиниции.

Пробвайте програмата дотук.

Добавяне на нова роднинска връзка.

1. Добавете предикат **brother** и неговите клаузи в **TaskWindow.pro** файла, където са и дефинициите на останалите предикати (father, grandfather и ancestor).

```
class predicates
    brother : (string Person1, string Person2) nondeterm anyflow.
clauses
    brother(Person1, Person2):-
    parent(Person1, Roditel),
    parent(Person2, Roditel),
    not(Person1 = Person2),
    person(Person1, male()).
```

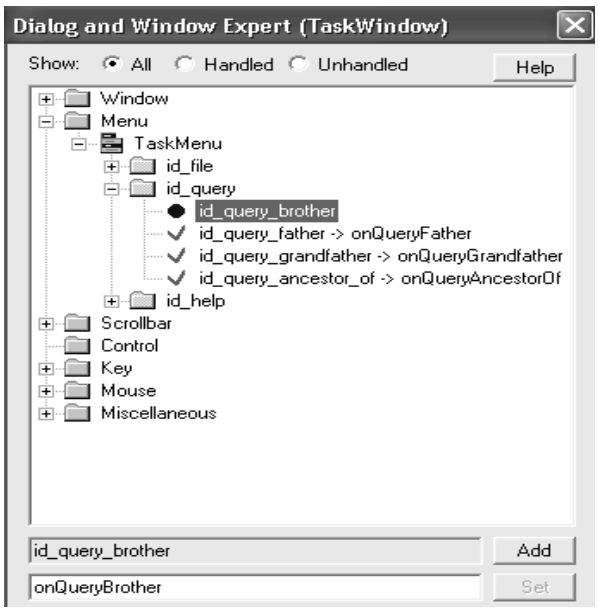
2. Добавете нова позиция (подменю) **Brother** в менюто **Query**. Използвайте **TaskMenu.mnu**. Позиционирайте се върху **Query** и от лентата с бутони изберете **new subitem**. Напишете името на подменюто в новопоявилата се икона. Останалите имена ще се изпишат автоматично, когато завършите

писането на името (не ги променяйте). Когато се опитате да затворите прозореца програмата, ще ви попита дали искате да запишете промените. Отговорете с **yes**.

3. Свържете новото подменю с код, който да се изпълни при избиране на менюто от потребителя. За свързването на подменюто с кода използвайте: TaskWindow.win\ (десен бутон на мишката)
\CodeExpert\Menu\TaskMenu\id_query\.
Щракнете два пъти върху синята точка на новото подменю (синята точка е индикация, че подменюто не е свързано с код). В прозореца, който ще се отвори, допълнете следния код

```
predicates
        onQueryBrother : vpiDomains::menuItemHandler.
clauses
        onQueryBrother(_MenuTag) = handled(0) :-
        stdIO::write("\n brother test \n"),
        brother(X,Y),
        stdIO::writef("% is a brother of % \n", Y,X),
        fail.
        onQueryBrother(_MenuTag) = handled(0).
```

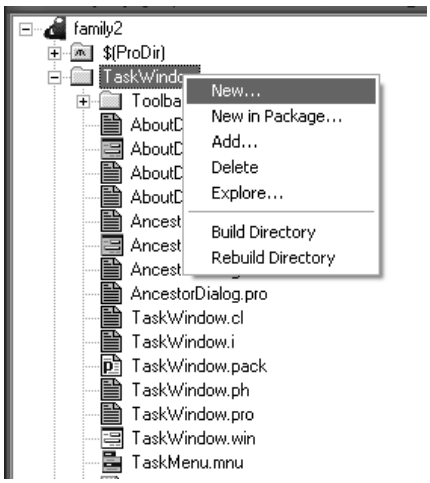
Този код е в същия файл TaskWindow.pro и се свързва с новото подменю. Обърнете внимание, че системата автоматично създава предикат за новото подменю, като оставя празни клаузите, които вие трябва да допълните!



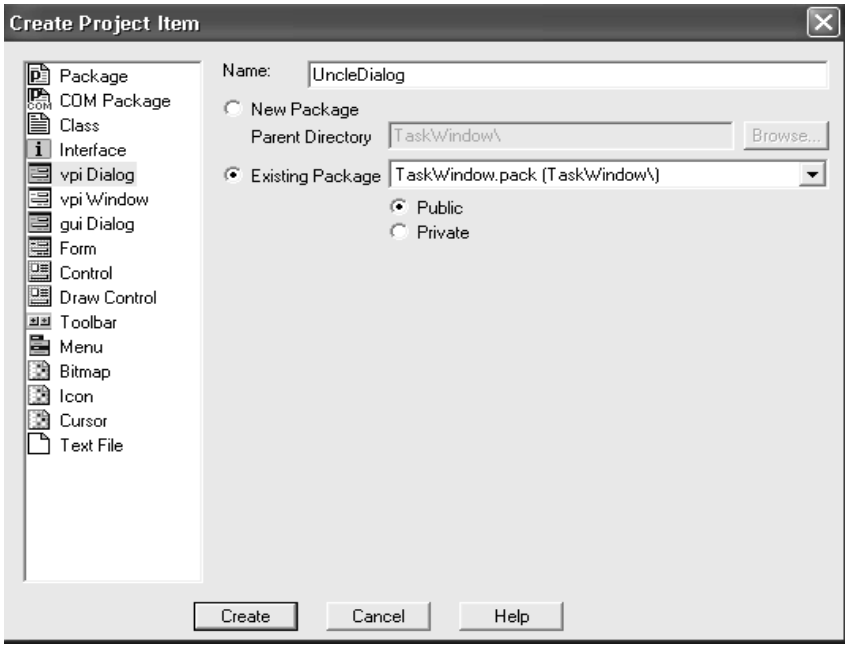
С тези си действия създадохме възможност да се извиква дефиницията на предиката brother(X,Y) и да се разпечатват всички братя от базата данни.

Създаване на диалог**.**

1. Избирате TaskWindow\десен бутон\New...\ vpiDialog

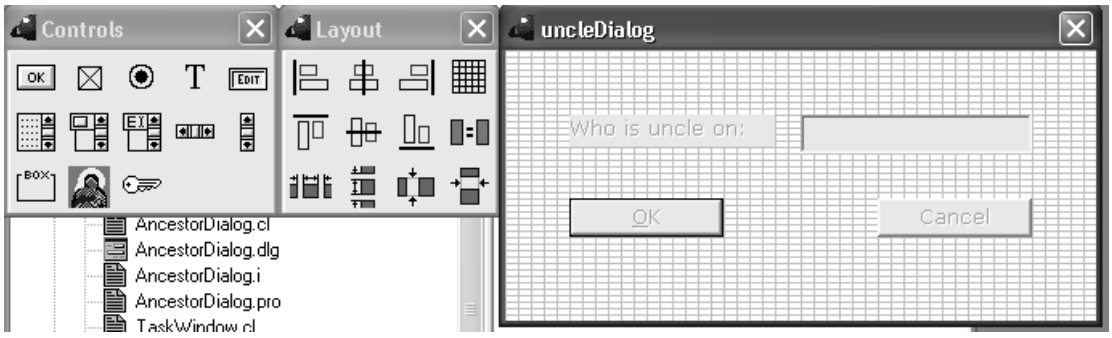


В появилия се прозорец попълвате името на диалога UncleDialog и останалите позиции, както е показано на следващия екран



2. Оформянето на диалоговия прозорец е в следните стъпки:

- Премахвате бутона help.
- Добавяте Textbox - text: Кой е чичо на: (Who is uncle on:).
- Добавяте EditControl , като оставяте празно полето за въвеждане на текст.



- Затваряте и записвате, при което автоматично се създава файлът uncleDialog.dlg.
- Завършвате с компилиране на програмата, с което се създават файловете на новия диалог: uncleDialog.cl, uncleDialog.i и uncleDialog.pro.

3. Дефинирате предиката Uncle във файла TaskWindow.pro, където са дефинициите на разгледаните досега предикати.

```
class predicates
   uncle : (string Person, string Uncle) nondeterm anyflow.
clauses
   uncle(Person, Uncle) :-
      father(Person, Father),
      brother(Uncle, Father),
      person (Uncle , male()).
```

4. Свързвате диалога с меню, за да може да се вика. Последователността е вече известна.

- TaskMenu.mnu.
- Добавяте подменю Uncle в менюто Query.
- Затваряте и съхранявате.

Свързвате менюто с код, който да се изпълнява при извикването му.

TaskWindow.win\десен бутон\Code Expert…\Menu… (отново Пролог ви предлага дефиниция на предиката onQueryUncle с празни клаузи, които да допълните!). Допълнен кодът има следния вид:

```
predicates
   onQueryUncle : vpiDomains::menuItemHandler.
clauses
   onQueryUncle(_MenuTag) = handled(0) :-
      X = uncleDialog::getName(thisWin),
      stdIO::write("\n uncle test \n"),
      uncle(X,Y),
      stdIO::writef("% is a uncle of % \n", Y,X),
      fail.
   onQueryUncle(_MenuTag) = handled(0).
```

5. В uncleDialog.cl се добавя кодът

```
class uncleDialog : uncleDialog
   open core

 predicates
      getName : (vpiDomains::windowHandle Parent) -> string Name determ.
```

6. В uncleDialog.pro се добавят клаузите за getName след тези редове:

```
clauses
   classInfo(className, classVersion).
```

се добавя следният код:

```
 domains
        optionalString = none(); one(string Value).
   class facts
        name : optionalString := none().
  clauses
      getName(Person) = Name :-
              name := none(),
             Dlg = uncleDialog::new(),
           Dlg:show(Person),
          one(Name) = name.
```

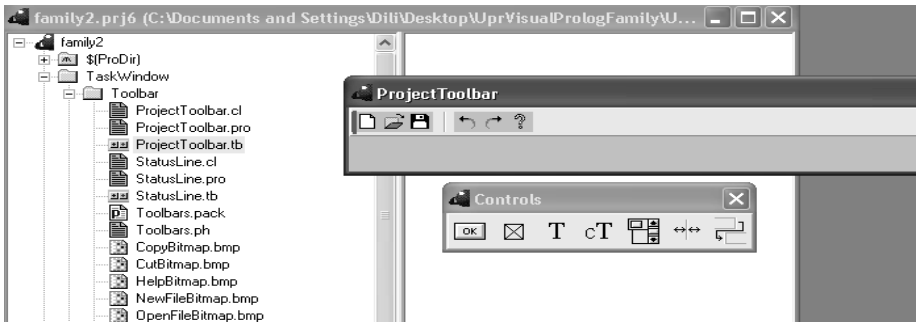
преди тези редове:

```
facts
   thisWin : vpiDomains::windowHandle := erroneous.
```

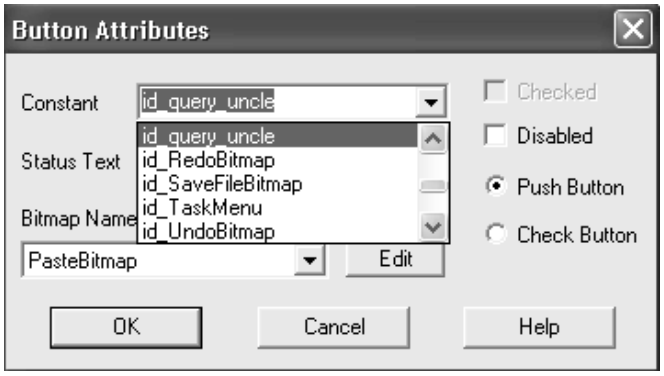
7. Добавят се и клаузи за бутон ОК пак в същия файл (намерете генерирания код и го допълнете):

```
predicates
   onControlOK : vpiDomains::controlHandler.
clauses
   onControlOK(_Ctrl, _CtrlType, _CtrlWin, _CtrlInfo) = handled(0) :-
         EditCtrl = vpi::winGetCtlHandle(thisWin, idc_uncledialog_1),
         Name = vpi::winGetText(EditCtrl),
         name := one(Name),
         vpi::winDestroy(thisWin).
```

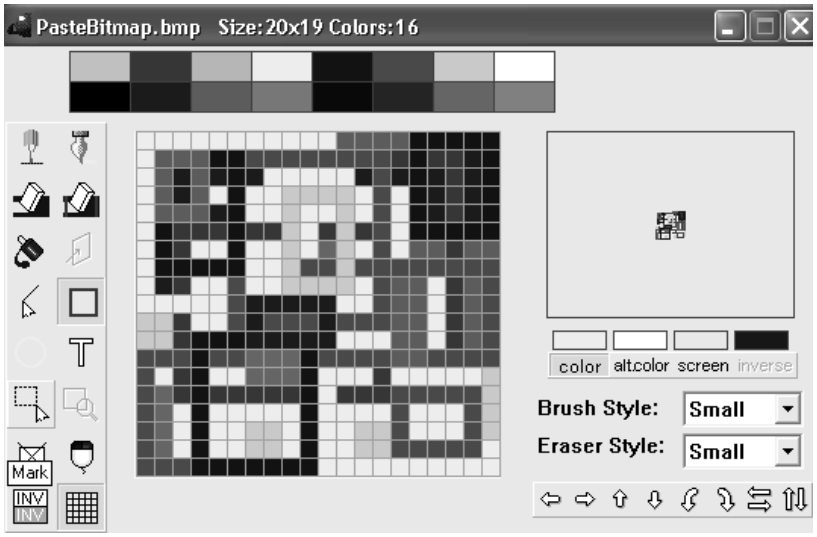
8. Добавяне на бутон в лентата с бутони меню Toolbar\ файл ProjectToolbar.tb



- Добавяте бутон Push Button от лентата с контроли, като щракнете върху нея.
- Виждате атрибутите на бутона
- Променяте рисунката на бутона чрез button **Edit** Bitmap.



- Свързвате бутона със съответното меню, като от горното падащо меню (Constant) изберете името на командата от менюто, която искате да стартирате.
- Ако щракнете на бутона Edit, който се намира в горния прозорец Button Attributes, ще можете да промените изрисуваната картинка върху добавения бутон.



Пробвайте програмата в завършения й вид.

Задание

1. Обогатете базата данни с още роднини.

2. Добавете нови роднински връзки със съответните диалози.

3. Нека си зададем въпроса, какви са взаимоотношенията между тези роднини? Доколко са съвместими характерите им и доколко се разбират по между си? Астрологията дава забавен отговор на тези въпроси на базата на построяването на звездна карта на всеки човек – кръг, в който се отбелязват разположението на планетите в мига на неговото раждане.

Аспектите са благоприятни, когато планетите се намират в един и същи зодиакален знак, образуват съвпад, когато отстоят на два знака (60°), образуват секстил, когато отстоят на четири знака и образуват тригон (120°). Аспектите не ни въздействат добре, когато планетите отстоят на три знака и образуват квадрат (90°) или на шест знака и образуват опозиция (180°).

За да се разтълкува рожденатa карта, необходимо е да се определят и разтълкуват местонахожденията на планетите в деня, часа и мястото на раждане на човека. Съставете програмна система, която прави партньорски хороскоп, като ползвате данни за местоположението на небесните тела за голям период от време и тълкуването им от астролог. На компакт диска, който придружава това ръководство разполагате с текстов файл с факти за местоположението на слънцето, луната и планетите за период от 100 години и с някои тълкувания на астроложката Линда Гуман, публикувани в нейната книга “Знаци на сродните души” през 1999г.

Файлът с данни се казва horoscope_DB1.txt. Разгледайте тяхното представяне и дефинирайте предикати, които да ги използват. На компакт диска можете да намерите описание на необходимите действия (стъпка по стъпка) за изпълнение на задачата, а също и примерен вариант на програмна система, която ви помага да построите звездната си карта и да определите степента си на съвместимост с даден човек.

Декларацията на някои предикати е следната:

```
domains
    list = string*.
class facts -  horoscopeDB

    aries :      (list BirthDay).  %определяне датата за всяка зодия.
    taurus :    (list BirthDay).
    gemini :    (list BirthDay).
      ….
    moon :   (string MonthYear, list FromToDateHour). %определяне
местоположението на
    mercury :   (string Year, list FromToMonthDate).     %луната и планетите в
годината
    venus :     (string Year, list FromToMonthDate).     %месеца и датата на
раждане.
    mars :       (string Year, list FromToMonthDate).
    aspect :  (string ZodiacSign1ZodiacSign2, string C_S_T_S_O). % определяне
дали
     % слънцето, луната и планетите на двамата партньори се намират в един
и същи
     % зодиакален знак, или образуват секстил, тригон или квадрат, или са в
опозиция.

    sun_sun :      (list C_S_T_S_O, string Data).    % След определянето на
аспектите, в които
```

```
sun_moon :  (list C_S_T_S_O, string Data).      % се намират планетите,
слънцето и луната
sun_mercury :    (list C_S_T_S_O, string Data). % на двамата партньори в
базата данни се
sun_venus :     (list C_S_T_S_O, string Data).    %  търси тълкуването
  …
moon_moon :  (list C_S_T_S_O, string Data).
moon_mercury :   (list C_S_T_S_O, string Data).
moon_venus :    (list C_S_T_S_O, string Data).
  …
mercury_mercury : (list C_S_T_S_O, string Data).
mercury_venus :     (list C_S_T_S_O, string Data).
  …
venus_venus :   (list C_S_T_S_O, string Data).
venus_mars :    (list C_S_T_S_O, string Data).
  …
```