ПРОЕКТ "РЕЧНИК НА ДВА ЕСТЕСТВЕНИ ЕЗИКА"

Въведение. Нека разработим програма-речник за изучаване на чужд език. Програмата ще предлага следните възможности:

1) Да се въвеждат двойки думи - едната на основния език, а другата на изучавания език. Думите се записват в текстов файл.

2) Да се извежда на екрана дума на един от двата езика и да се сравнява с въведена от потребителя дума на другия език.

3) Да се проверява правилността на въвежданата дума и да се отброяват правилните отговори.

4) При погрешен отговор да се изписва правилният отговор.

5) Да дава оценка при желание от потребителя на неговата успеваемост по време на теста.

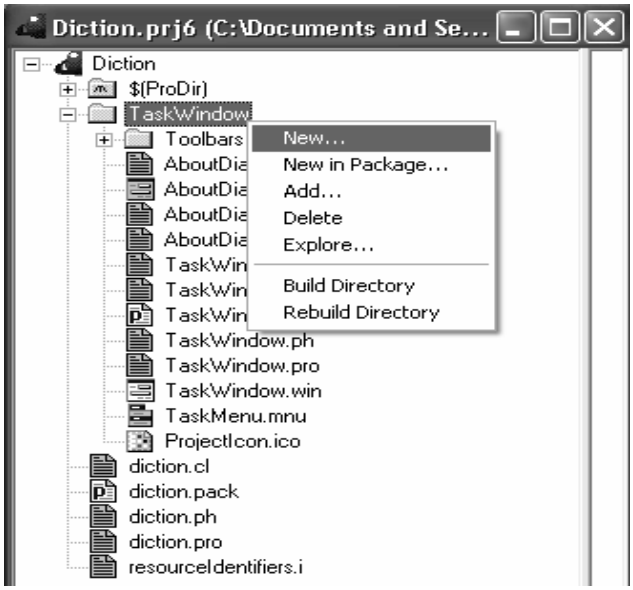
6) При правилен отговор за повече от половината думи програмата дава много добра оценка. В противен случай тя подканя потребителя да продължи да се упражнява.

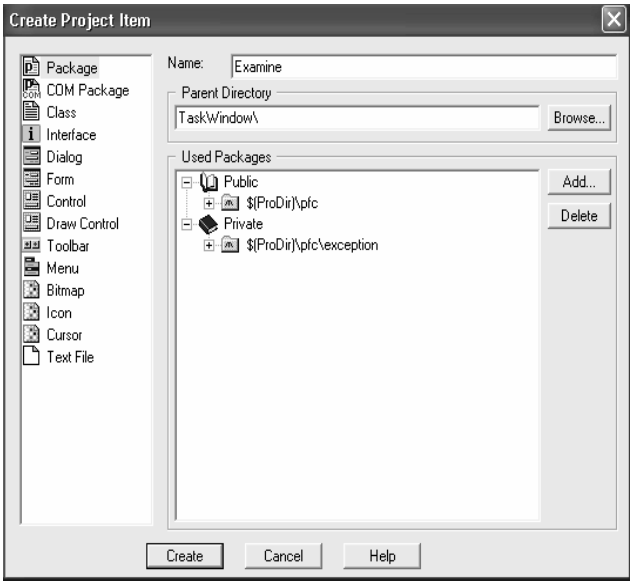
Постановка. За реализацията на програмата ще използваме обектно ориентираните възможности на Visual Prolog 7.0 и неговия графичен потребителски интерфейс (Object-oriented GUI [pfc/gui]; Exe).

Започнете нов проект на Visual Prolog 7.0. Нека името на проекта е Diction.

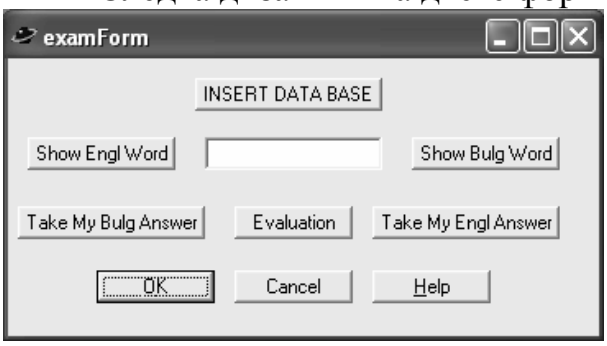
Създайте нов пакет Examine и форма в него ExamForm.

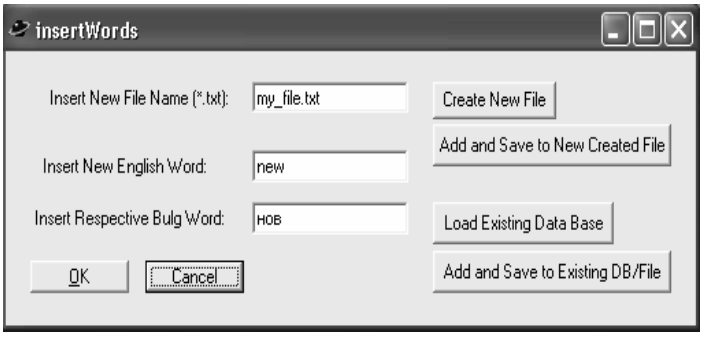
Създайте още един нов пакет InsertWords и форма в него InsertWordsForm за въвеждане на съответстващи думи на двата езика в нов текстов файл.



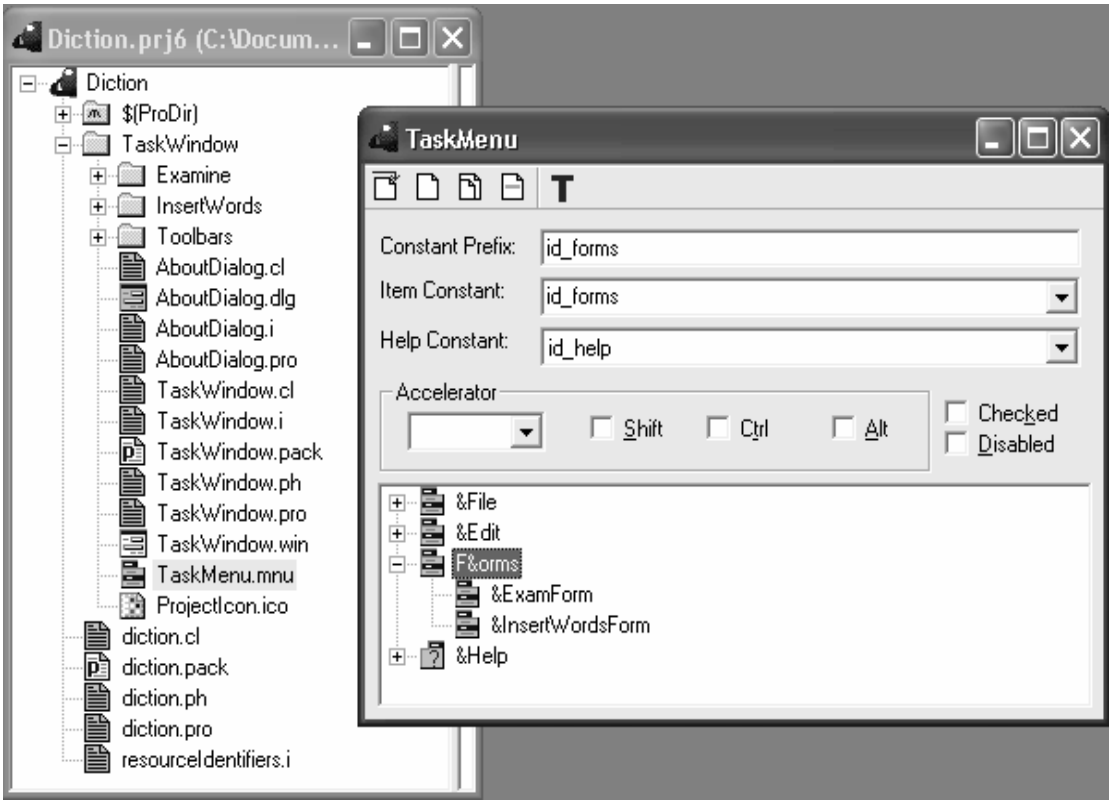


Следва дизайнът на двете форми.





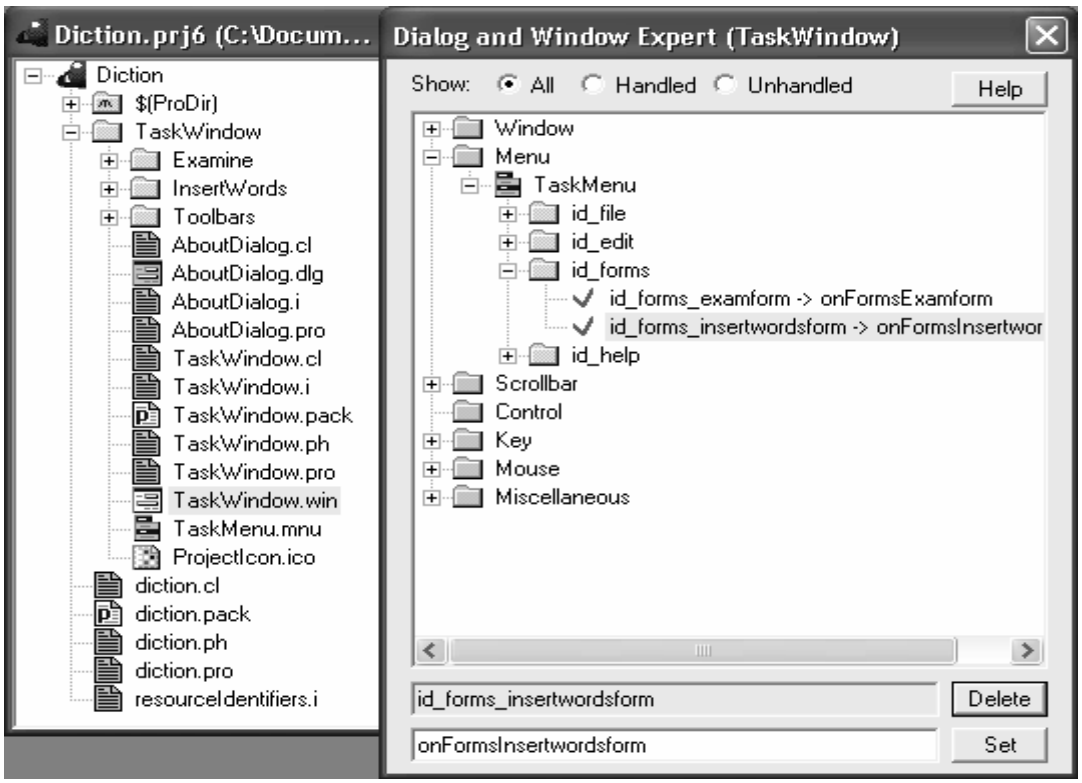
За да могат да се извикват двете форми, ще направим ново меню с име Forms и две подменюта на това меню с имена съответно ExamForm и InsertWordsForm.



Сега да свържем тези подменюта със съответния код за визуализиране на формите.

```
predicates
   onFormsExamform : window::menuItemListener.
clauses
   onFormsExamform(_Source, _MenuTag):-
     EF=examForm::new(This),  EF:show().
```

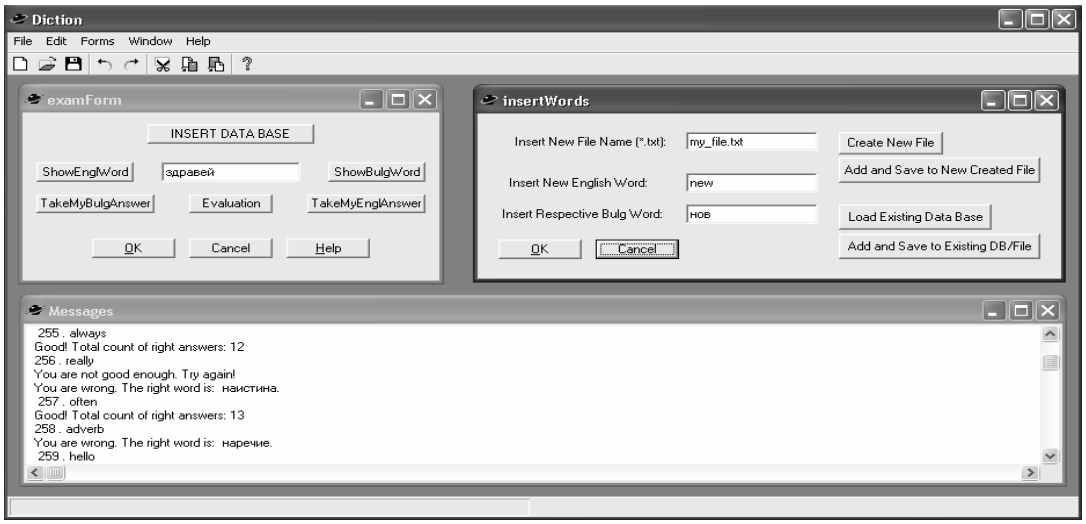
```
predicates
   onFormsInsertwordsform : window::menuItemListener.
clauses
   onFormsInsertwordsform(_Source, _MenuTag):-
     IW=insertWordsForm::new(This),
     IW:show().
```



Ако искате формата за учене на думи да се показва по подразбиране при стартиране на проекта, необходимо е да допълните клаузата на предиката onShow на TaskWindow.pro по следния начин:

```
predicates
   onShow : window::showListener.
clauses
   onShow(_, _CreationData):-
      _MessageForm = messageForm::display(This),
      EF = examForm::new(This),
      EF:show().
```

Компилирайте програмата дотук. Пробвайте работата на двете команди от менюто Forms. На следващата картинка виждате визуализирани двете форми.



Можете да видите как изглежда дървото на проекта. Разполагаме с текстов файл (dict1_db.txt), който съдържа 310 думи на български и английски език. Думите са номерирани. Предикатът ebw(English-Bulgarian Word) е триместен. Ето част от фактите в този файл:

```
clauses
ebw(1,"ship","кораб").
ebw(2,"sheep","овца").
ebw(3,"far","далеч").
ebw(4,"girl","момиче").
```

Ще използваме този файл, за да проверяваме работата на създаваната програма. По-късно ще създадем и други текстови файлове. Поставете този файл в папка Exe, която се намира в папката на вашия проект.

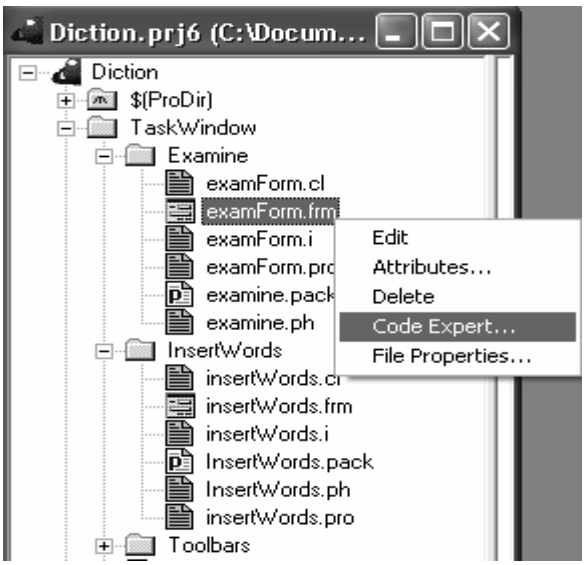
Типът на данните в базата данни се описва във файла examForm.pro след редовете:

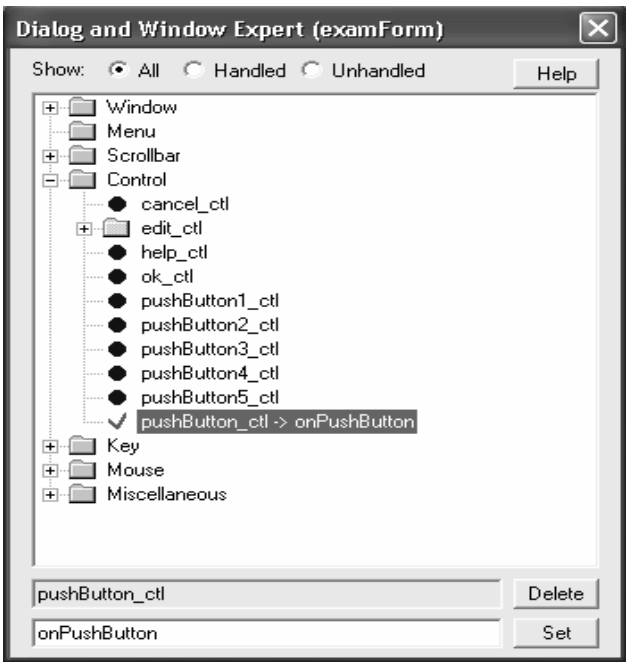
```
clauses
   classInfo(className, classVersion).
```

по следния начин:

```
domains
      list = string*.
class facts -  dict1DB
      ebw :       (integer Number, string Engl_Word, string Bulg_Word).
```

Сега да напишем клаузите за зареждане на тази база данни при натискането на бутона INSERT DATABASE от ExamForm, така че формата да работи с фактите от този файл.





Допълваме клаузите на този команден бутон във файла examForm.pro.

```
predicates
   onPushButton : button::clickResponder.
clauses
   onPushButton(_Source) = button::defaultAction() :-
          FileName = vpiCommonDialogs::getFileName(
                        "*.Txt", ["Txt files (*.txt)","*.txt","All files", "*.*"],
                         "Load dict1_db database",
                         [], ".", _),
                         ! ,
                         reconsult(FileName),
          stdIO::writef("Database % loaded\n", FileName).
   onPushButton(_Source) = button::defaultAction().
```

Тъй като се извиква предикатът reconsult, той трябва да се декларира и дефинира с код, който се поставя в началото на файла, където описахме дефиницията на базата данни:

```
class predicates
      reconsult : (string FileName).
clauses
      reconsult(FileName) :-
         retractFactDB( dict1DB),
         file::consult(FileName, dict1DB).
```

Компилирайте и пробвайте програмата дотук. Трябва да можете да визуализирате двете форми и да заредите базата данни от бутона INSERT DATA BASE.

Сега да добавим и предикатите, с които програмата ще предлага дума на английски език, ще взема от текстовото поле написаната от потребителя дума на български, ще проверява дали е правилна, ще извежда съответните съобщения, ще променя стойностите на броячите и ще оценява знанията на потребителя. Да не забравим да дефинираме двата брояча: counter, който показва номера на текущата дума, и counterOKA – показва броя на правилно написаните от потребителя думи. Класовете факти и предикатите на формата се добавят отново във файла examForm.pro.
Ето предикатите и техните клаузи:

```
class facts
        counter : integer := 1.
        counterOKA : integer :=0.
class predicates
    test_engl_bulg : () nondeterm.
clauses
     test_engl_bulg():-
            ebw(counter,Engl,Bulg),
            stdIO::writef(" % . % \n",counter,Engl),
            !.
    test_engl_bulg():-
            stdIO::writef("\n mark:"),
            evaluation(),
            counter:=1,
            counterOKA:=0,
            stdIO::writef("We start again. Your counters are set to zero \n"),
            !.
class predicates
    test_bulg_engl : () nondeterm.
clauses
     test_bulg_engl ():-
            ebw(counter,Engl,Bulg),
            stdIO::writef(" % . % \n",counter,Bulg),
            !.
    test_bulg_engl():-
            stdIO::writef("\n mark:"),
            evaluation(),
            counter:=1,
            counterOKA:=0,
            stdIO::writef("We start again. Your counters are set to zero \n"),
            !.
```

```
class predicates
   isOK : (string X, string X1) nondeterm.
clauses
   isOK(X,X):-
         counter:=counter+1,
         counterOKA := counterOKA+1,
         stdIO::writef(" Good! Total count of right answers: % \n", counterOKA),
         !.
   isOK(X,X1):-
         counter:=counter+1,
         stdIO::writef(" You are wrong. The right word is:  %. \n ",X),
         !.
 class predicates
   evaluation : () nondeterm.
clauses
   evaluation():-
       counter - counterOKA = 0,
       stdIO::writef("Excellent! You are perfect! "),
       !.
   evaluation():-
       counter - counterOKA > counterOKA ,
       stdIO::writef(" You are not good enough. Try again!  \n"),
       !.
   evaluation():-
       stdIO::writef(" OK! You are very good! \n"),
       !.
```

Предикатите test_engl_bulg : () nondeterm. и test_bulg_engl : () nondeterm. имат две клаузи. Първата клауза взема от базата данни дума с номер съдържанието на брояча counter и я извежда на екрана съответно на български (test_bulg_engl) или на английски език (test_engl_bulg). Втората клауза се изпълнява само ако броячът показва номер, по-голям от броя на думите в текстовия файл, т.е. когато всички думи са били визуализирани на екрана. Втората клауза извиква предиката за оценяване на знанията на потребителя и установява броячите в начално състояние. Броячът на думи се установява в единица, за да започне показването на думите отначало, а броячът на правилните отговори се нулира, за да започне оценяването на знанията отначало.

Потребителят може да получава думи на български или английски език в зависимост от това кой от двата бутона ще щракне. Докато потребителят не въведе своя отговор (макар и погрешен), програмата ще чака. Ако искате, можете да промените това.

Предикатът isOK : (string X, string X1) nondeterm. се извиква от бутоните за получаване на отговора на потребителя. Този предикат има два аргумента. Ако системата задава на потребителя думи на английски, а той пише

съответните думи на български, този предикат получава като аргумент въведения в текстовото поле отговор на потребителя и думата на български от текстовия файл. В случай, че системата задава дума на български, а потребителят пише съответната дума на английски, този предикат получава като аргумент въведената в текстовото поле от потребителя дума на английски и думата на английски взета от текстовия файл. Първата клауза на този предикат се изпълнява, ако двата аргумента са еднакви, т.е. когато думата, въведена в текстовото поле, е същата като тази, взета от текстовия файл – речник. Тогава броячът на думите се увеличава с единица, за да се вземе следващата дума от базата данни, а броячът на правилни отговори се увеличава с единица, като извежда поощрително съобщение на потребителя. Тази клауза завършва с !(cut), затова втората клауза няма да се изпълни. Текстовото поле се изчиства, за да е готово за ново въвеждане на отговор.

Втората клауза се изпълнява, ако първата не се е изпълнила, т.е. когато отговорът на потребителя не съответства на думата от речника. Тогава се увеличава с единица само броячът на думите и се извежда правилното значение на думата. Текстовото поле отново се изчиства.

Предикатът evaluation : () nondeterm може да се извика при натискане на бутона Evaluation или при условие, че всички думи от речника са били извеждани. Този предикат има три клаузи. Първата се изпълнявава, ако всички отговори на потребителя са правилни. Тогава се изписва съобщението “Excellent! You are perfect!”. Ако първата клауза не се изпълни, ще се активира втората клауза, която проверява дали погрешните отговори на потребителя са повече от правилните отговори. Ако е така, се извежда съобщението: “You are not good enough. Try again! \n”. Ако и втората клауза не се изпълни, правилните отговори на потребителя са повече от грешките и се изписва съобщението: “OK! You are very good! \n”.

Предикатът за оценяване може да бъде извикан по всяко време.

Добавете тези предикати във файла examForm.pro след предиката reconsult.

Остава само да свържете бутоните с код, който да извиква тези предикати и да пробвате програмата. От дървото на проека щракнете с десния дутон на мишката върху файла examForm.frm\Code Expert… \ Controls \ и добавете предикати към всеки бутон на формата (сините точки до тях показват, че те не са свързани с код).

Ето допълнителния код за всеки бутон от нашата форма:

Внимание! Винаги, когато свързвате бутон с код внимавайте какво е името на бутона във вашия проект и какво е името на бутона, зададено тук. Използвайте имената, които вие сте поставили, вместо тези, които са дадени тук!

```
predicates
   onPushButton1 : button::clickResponder.
clauses
   onPushButton1(_Source) = button::defaultAction() :-
         test_engl_bulg(), !.
   onPushButton1(_Source) = button::defaultAction().

predicates
   onPushButton2 : button::clickResponder.
clauses
   onPushButton2(_Source) = button::defaultAction() :-
         ebw(counter,Engl,Bulg),
         Answer = edit_ctl:getText(),
         isOK(Bulg,Answer), !.
   onPushButton2(_Source) = button::defaultAction().

predicates
   onPushButton3 : button::clickResponder.
clauses
   onPushButton3(_Source) = button::defaultAction() :-
         test_bulg_engl(), !.
    onPushButton3(_Source) = button::defaultAction().

predicates
   onPushButton4 : button::clickResponder.
clauses
   onPushButton4(_Source) = button::defaultAction() :-
         ebw(counter,Engl,Bulg),
         Answer = edit_ctl:getText(),
         isOK(Engl,Answer),
         edit_ctl:setText(""), !.
   onPushButton4(_Source) = button::defaultAction().

predicates
   onPushButton5 : button::clickResponder.
clauses
   onPushButton5(_Source) = button::defaultAction() :-
         evaluation(), !.
   onPushButton5(_Source) = button::defaultAction().
```

Дойде време да разгледаме и формата за въвеждане на нови думи в текстов файл, чието име си изберете сами. За целта е необходимо да напишете кода за зареждане на файл с база данни на бутона Add word to the data base. Той ще взема въведените думи, например на английски и на български, ще увеличава брояча на думите с единица всеки път, когато се добави дума в

текстовия файл, и ще добавя думи във файла, който сме заредили в паметта с другия бутон.

Първо ще трябва да заредите текстов файл, в който ще се допълват новите думи на български и английски. Новата дума ще бъде добавена най-накрая на файла с вградения предикат assertz(FactFromDataBase), който има за аргумент името на факт от базата данни. Този предикат може да се използва, за да се допълва с факти файл, в който са записвани данни. Няма да променяме фактите в базата данни. Работим с един тип факти ebw(counter, English, Bulgarian). Следователно предикатът ще изглежда по следния начин:

assertz(ebw(counter, English, Bulgarian)).

Този предикат добавя динамично новите факти. За да ги запомним във файла и да ги ползваме при следващо включване на компютъра, ще използваме предиката file::save(FileName, DataBaseName) с аргументи името на файла и името на базата данни.

Във файлът InsertWordsForm.pro след редовете:

```
constants
   className = "TaskWindow/InsertWords/insertWords".
   classVersion = "".
clauses
   classInfo(className, classVersion).
```

добавете следния код:

```
domains
      list = string*.
class facts -  dict1DB
      ebw :       (integer Number, string Engl_Word, string Bulg_Word).

class facts
      counter : integer := 0.

class predicates
      reconsult : (string FileName).
clauses
      reconsult(FileName) :-
         retractFactDB( dict1DB),
         file::consult(FileName, dict1DB).
```

```
class predicates
      mysave : (string FileName).
clauses
      mysave(FileName) :-
         file::save(FileName, dict1DB).

class predicates
   old_counter : ().
clauses
   old_counter() :-
      ebw(_,_,_),
      counter:=counter+1,
      fail.
  old_counter().
```

Предикатът old_counter преброява думите, записани до момента в базата данни.

Свържете бутона за зареждане на съществуващ файл с база данни със следния код:

```
predicates
   onPushButton1 : button::clickResponder.
clauses
   onPushButton1(_Source) = button::defaultAction() :-
          FileName = vpiCommonDialogs::getFileName(
                        "*.Txt", ["Txt files (*.txt)","*.txt","All files", "*.*"],
                         "Load dict1_db database",
                         [], ".", _),
                         ! ,
                        reconsult(FileName),
                        edit_ctl:setText(FileName),
                        counter:=0,
                        old_counter(),
          stdIO::writef("Database % loaded\n", FileName).
   onPushButton1(_Source) = button::defaultAction().
```

Свържете бутона за добавяне на нова дума към съществуваща база данни със следния код:

Внимание: Следващите бутони вземат думите и името на файла от текстовите полета, които вие поставихте във формата. Внимателно проверете какви са имената им във вашия проект и ако има разлика с имената им тук, то използвайте вашите имена.

```
predicates
   onPushButton : button::clickResponder.
clauses
   onPushButton(_Source) = button::defaultAction() :-
      English = edit1_ctl:getText(),
      Bulgarian = edit2_ctl:getText(),
      FileName=edit_ctl:getText(),
      reconsult(FileName),
      counter:=0,
      old_counter(),
      counter:=counter+1,
      assertz(ebw(counter, English, Bulgarian)),
      mysave(FileName),
      stdIO::writef("Words are saved in % file. \n", FileName),
      !.
   onPushButton(_Source) = button::defaultAction().
```

Тук предикатът old_counter() преброява думите, въведени до момента, за да може новата дума да получи следващия пореден номер.

Свържете бутона за създаване на нов текстов файл за друга база данни с думи със следния код:

```
predicates
   onPushButton2 : button::clickResponder.
clauses
   onPushButton2(_Source) = button::defaultAction() :-
      counter:=0,
      retractFactDB( dict1DB),
      FileName=edit_ctl:getText(),
      mysave(FileName),
      stdIO::writef("File % created. DataBase is retracted. \n", FileName),
      !.
   onPushButton2(_Source) = button::defaultAction().
```

Свържете бутона за добавяне на думи в новосъздадения файл със следния код:

```
predicates
   onPushButton3 : button::clickResponder.
clauses
   onPushButton3(_Source) = button::defaultAction() :-
      English = edit1_ctl:getText(),
      Bulgarian = edit2_ctl:getText(),
```

```
    FileName=edit_ctl:getText(),
    reconsult(FileName),
    counter:=0,
    old_counter(),
    counter:=counter+1,
    assertz(ebw(counter, English, Bulgarian)),
    mysave(FileName),
    stdIO::writef("Words are saved in % file. \n", FileName),
    !.
  onPushButton3(_Source) = button::defaultAction().
```

Задание

1. Помислете как да се усъвършенства предикатът за оценяване на знанията и реализирайте вашата идея.

2. Като имате предвид, че една дума може да има няколко значения и повече от един варианти на изписване (например в засимост от род и число), потърсете по-подходящ вариант за базата данни и за предикатите за визуализиране на думи и получаване на съответния отговор.

Забележка. Подходящо е да се работи със списъци от съответстващи си думи.

3. Модифицирайте програмата така, че при извеждането на думите от базата данни да се работи с променлива стъпка. Например в началото през пет, след това през четири, през три, през две и накрая последователно.

4. Реализирайте вариант за извеждане на думи под управлението на псевдослучайна последователност от числа.

5. Реализирайте функция на програмата-речник, която да показва значението на въведена от потребителя дума.

Забележка. Може да реализирате програмата по два начина - като работите със съответстващи си думи на двата езика или като работите със списъци от съответстващи си думи.