

Министерство образования Республики Беларусь

Учреждение образования  
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет информационных технологий и управления

Кафедра информационных технологий автоматизированных систем

Отчёт  
по практической работе №3  
«Развитые типы данных в программах на Прологе. Базы данных в  
программах на Прологе»  
по дисциплине «Экспертные Системы»

Вариант 2

Выполнил:  
студент гр. 820601  
Шведов А. Р.

Проверила:  
Т. В. Тиханович

Минск 2022

# 1 ЦЕЛЬ РАБОТЫ

Целью работы является элементарное введение в базы данных в системе *VISUAL PROLOG(VIP)*. Необходимо ознакомиться и разобраться с операциями создания, открытия, поиска и изменения информации в базе данных и самостоятельно написать учебную программу для закрепления навыков программирования.

.

## 2 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

### 2.1 Работа с базами данных в языке Пролог

Данные хранятся в базе данных в форме записей. Запись состоит из полей, представляющих различные типы данных. В Прологе необходимо объявить структурированный тип для записей базы данных – функтор. Пусть в базе содержатся сведения о футбольных командах; полями таких записей будут название команды, страна и международный рейтинг. Пример объявления данных записей представлен ниже:

```
DOMAINS
name=symbol
country=symbol
rate=integer
teams=teams(name, country, rate)
```

Раздел *DOMAINS* в Прологе используется для объявления нестандартных или пользовательских типов. Выше объявлены следующие типы: *name*, *country*, *rate*, *teams*. Наиболее интересен последний тип, который называется функтором. Примером индивидуальной записи такого типа может служить следующая запись:

```
teams(милан, италия, 75).
```

В Прологе есть два типа баз данных: внешние и внутренние. Внутренние целиком грузятся в оперативную память, а потому имеют скорее учебное назначение. Все команды для работы с внешними базами данных делятся на четыре группы:

Первая группа: общие команды для работы с базами данных. Сюда относятся команды создания, открытия, закрытия, удаления, сжатия, копирования и др.

Вторая группа: команды для работы с цепочками записей. Под цепочкой (*chain*) понимается последовательность записей, имеющих один и тот же тип.

Третья группа: команды для работы с индивидуальными записями.

Четвертая группа: команды для работы с *B*-деревьями. *B*-деревья предназначены для быстрого поиска данных по ключам. В качестве ключей могут использоваться, например, названия команд.

Создание базы данных выполняется командой *db\_create*(селектор, имя\_файла, место). Селектор является объектом типа *db\_selector* и объявляется в разделе *GLOBAL DOMAINS*. Этот раздел должен быть единственным в приложении. Однако система сама его создает в файле с

расширением *PRE*, который она подключает к программе с помощью команды *#include*. Селектор используется в качестве программного имени базы данных. Имя файла – это имя файла на диске. Место – это один из следующих стандартных спецификаторов:

- *in\_file*: база данных создается в файле и сохраняется в файле;
- *in\_memory*: база данных создается в памяти ЭВМ;
- *in\_ems*: для создания базы данных используется расширенная память.

Открытие базы данных выполняется командой *db\_open*(селектор, имя\_файла, место).

Заккрытие базы данных выполняется командой *db\_close*(селектор)

Удаление базы данных выполняется командой *db\_delete*(имя\_файла, место). Можно только закрытую базу. Команда создания базы данных одновременно делает ее открытой.

Для открытия поврежденной базы данных используется команда *db\_openinvalid*(селектор, имя\_файла, место).

Повреждение базы может возникнуть при сбое питания и открытой базе, некорректном завершении работы.

Команда выбора следующей по порядку цепочки: *db\_chains*(селектор, *Chain*) Эта команда присваивает переменной *Chain* имя очередной цепочки. Если цепочек больше нет, то возникает возврат в ближайшую точку ветвления.

Следующая команда возвращает ссылку на первую запись в цепочке: *chain\_first*(селектор, имя\_цепочки, ссылка).

Следующий предикат отыскивает и возвращает ссылку на указанную запись: *chain\_terms*(селектор, имя\_цепочки, тип\_цепочки, запись, ссылка).

Следующая команда добавляет запись в начало цепочки: *chain\_inserta*(селектор, имя\_цепочки, тип\_записи, запись, ссылка). Последний аргумент не должен быть задан и получает значение. Пример: *chain\_inserta(mydba, chain1, dbdom, city("Minsk", "Belarus"), Ref)*.

Следующая команда удаляет всю цепочку: *chain\_delete*(селектор, имя\_цепочки).

Команда создания дерева имеет такой формат: *bt\_create*(селектор, имя\_дерева, селектор\_дерева, длина\_ключа, степень\_ветвления\_вершин). Здесь селектор\_дерева не должен быть задан и получает значение в результате выполнения команды. Остальные параметры должны быть заданы. Длина\_ключа имеет тип *UNSIGNED* и определяет максимальное число символов в ключе; параметр степень\_ветвления\_вершин имеет тип *UNSIGNED*, и рекомендуется устанавливать его равным 3 или 4.

Команда *bt\_close*(селектор, селектор\_дерева) закрывает дерево.

Следующая команда удаляет В-дерево: *bt\_delete*(селектор, имя\_дерева).

Для открытия В-дерева используется команда *bt\_open*(селектор, имя\_дерева, селектор\_дерева). Эта команда использует в качестве входных аргументов первые два. Селектор дерева возвращается в качестве результата.

Команда закрытия дерева имеет такой вид: *bt\_close*(селектор, селектор\_дерева). Основными операциями на дереве являются операции с ключами. Для вставки ключа в дерево используется команда: *key\_insert*(селектор, селектор\_дерева, ключ, ссылка). Все аргументы должны быть заданы. Это значит, что запись уже должна содержаться в базе данных и на нее существует ссылка. Ссылку, в частности, можно получить при вставке записи в базу данных командой *chain\_inserta*.

Удаление ключа из дерева выполняет команда *key\_delete*(селектор, селектор\_дерева, ключ, ссылка). Все аргументы должны быть заданы.

Для получения ссылки по значению ключа используется команда: *key\_search*(селектор, селектор\_дерева, ключ, ссылка). Первые три аргумента должны быть заданы, последний возвращается системой. Если в дереве нет указанного ключа, то возникает состояние неудачи и выполняется возврат к предыдущей точке ветвления.

## 3 ПРАКТИЧЕСКАЯ ЧАСТЬ

### 3.1 Ход работы

Согласно заданию, создадим базу данных и интерфейс для работы с ней по теме «Товары».

В разделе *global domains* объявим типы *id*, *name*, *price*, *creatorCompany*, соответствующие типам *integer*, *string*, *real* и *string* соответственно. Также был объявлен функтор *goods=goods(id, name, price, creatorCompany)*, описывающий тип базы данных товаров. В файле *lab3.pre* был объявлен селектор базы данных *db\_selector = goodsDB*.

В разделе *predicates* объявим предикаты *start*, *repeat*, *selectOption(integer)*, *createDB*, *openDB*, *closeDB*, *deleteDB*, *addGoods*, *searchInDB*, *searchInDB(name, ref)*, *removeGoods*.

Целевым предикатом является предикат-правило ***start***. Он вызывает предикат ***repeat*** для создания точки, на которую позже сможет вернуться выполнение программы после вызова оператора *fail*, стандартный предикат *write()* для вывода на экран сообщений о доступных пользователю опциях (меню), стандартный оператор *readInt()* для считывания ввода выбранной опции пользователем в переменную *Option*, а так же предика *selectOption* для передачи о обработки выбранной опции.

Предикат ***selectOption(integer)*** вызывает предикаты *createDB*, *openDB*, *closeDB*, *deleteDB*, *addGoods*, *searchInDB*, или *removeGoods* в зависимости от переданного значения. Для всех опций, кроме последней (выход), с помощью отсечения и конъюкции с оператором *fail* реализуется повтор вывода меню.

Предикат ***createdDB*** проверяет файл *lab3\_1.bin* на наличие оператором *existfile* и выводит сообщение о существовании данного файла в случае возвращения положительного значения, либо создаёт данный файл, базу данных и *B*-дерево в случае возвращения отрицательного значения. Создание базы данных осуществляется вызовом предиката *db\_create*, создание *B*-дерева осуществляется вызовом предиката *bt\_create*. После создания базы данных и *B*-дерева осуществляется вывод на экран сообщения об успешном создании базы данных, а также закрытие *B*-дерева с помощью предиката *bt\_close* и базы данных с помощью предиката *db\_close*.

Предикат ***openDB*** осуществляет открытие базы данных с помощью предиката *db\_open()*, а так же выводит сообщение об удаче.

Предикат ***closeDB*** осуществляет закрытие базы данных с помощью предиката *db\_close()*, а так же выводит сообщение об удаче либо неудаче.

Предикат ***deleteDB*** осуществляет проверку на существование файла *lab3\_1.bin* с помощью предиката *existfile()* и осуществляет удаления базы данных с помощью предиката *db\_delete()* с выводом сообщения об удалении базы при возвращении положительного значения, либо выводит на экран сообщения об отсутствии базы данных в случае возвращения отрицательного значения.

Предикат ***addGoods*** осуществляет вывод на экран сообщения о необходимости ввести идентификационный номер, название, цену, а также компанию-производителя товара, далее осуществляет считывание введенных данных в переменные *ID*, *Name*, *Price* и *CreatorCompany* с помощью стандартных предикатов *readint*, *readln* и *readreal*. Затем осуществляется вставка товара с введенной информацией в базу данных с помощью предиката *chain\_inserta*, открытие *B*-дерева с помощью предиката *bt\_open*, вставка в *B*-дерева ссылки на функтор по ключу *Name* с помощью предиката *key\_insert*. Вставка ссылки по ключу имени в *B*-дерева необходима для последующей реализации поиска. После этого *B*-дерево закрывается с помощью предиката *bt\_close()*. Далее выводится сообщение об успешном добавлении товара в базу данных.

Предикат ***searchInDB(name, ref)*** принимает параметры имени для поиска по *B*-дереву и переменной для занесения ссылки на найденный результат. *B*-дерево открывается с помощью предиката *bt\_open()*, затем осуществляется поиск ссылки на функтор по дереву по ключу переданного параметра *Name* с помощью предиката *key\_search*. Результат поиска заносится в переменную *TermRef*. Ссылка из переменной *TermRef* заносится в полученную переменную *Ref*. Далее *B*-дерево закрывается с помощью предиката *bt\_close*.

Предикат ***searchInDB*** реализует вывод сообщения о запросе названия товара для поиска, считывание введенного значения в переменную *Name*, передачу данного значения, а также переменной *Ref* в предикат *searchInDB*, получение функтора из ссылки, занесенной в переменную *Ref* с помощью предиката *ref\_term*, а также вывод найденной информации о товаре на экран.

Предикат ***removeGoods*** реализует открытие *B*-дерева с помощью предиката *bt\_open*, вывода сообщения о запросе наименования товара для удаления, считывание введенного значения в переменную *Name* с помощью оператора *readln()*, поиск в базе данных товара по введенном наименованию с помощью предиката *searchInDB(name, ref)* с занесением ссылки в переменную *Ref*, удаления ключа *Name* из *B*-дерева с помощью предиката *key\_delete()*, закрытие *B*-дерева с помощью предиката *bt\_close*, удаление функтора из базы

данных с помощью предиката *term\_delete*, а так же вывод сообщения об успешном удалении товара из базы данных.

Пример создания базы данных представлен на рисунке 1.

```
1 - Create DB
2 - Open DB
3 - Close DB
4 - Delete DB
5 - Add new good to DB
6 - Search in DB
7 - Remove goods from DB
8 - Exit

Select an option: 1

DB created.
```

Рисунок 1 – Создание базы данных

Пример открытия базы данных представлен на рисунке 2.

```
1 - Create DB
2 - Open DB
3 - Close DB
4 - Delete DB
5 - Add new good to DB
6 - Search in DB
7 - Remove goods from DB
8 - Exit

Select an option: 2

DB opened.
```

Рисунок 2 – Открытие базы данных

Пример открытия базы данных представлен на рисунке 3.



```
1 - Create DB
2 - Open DB
3 - Close DB
4 - Delete DB
5 - Add new good to DB
6 - Search in DB
7 - Remove goods from DB
8 - Exit

Select an option: 5

Enter id: 1

Enter name: iPhone XS

Enter Price: 999

Enter creator company: Apple

Good added.
```

Рисунок 3 – Добавление товара в базу данных

Пример поиска в базе данных представлен на рисунке 4.

```
1 - Create DB
2 - Open DB
3 - Close DB
4 - Delete DB
5 - Add new good to DB
6 - Search in DB
7 - Remove goods from DB
8 - Exit

Select an option: 6

Enter goods name to search: iPhone XS

Found goods: (1, iPhone XS, 999, Apple)
```

Рисунок 4 – Поиск товара в базе данных

Пример удаления товара из базы данных представлен на рисунке 5.

```
1 - Create DB
2 - Open DB
3 - Close DB
4 - Delete DB
5 - Add new good to DB
6 - Search in DB
7 - Remove goods from DB
8 - Exit

Select an option: 7

Enter goods name to remove from DB: iPhone XS

Goods with name 'iPhone XS' removed from DB
```

Рисунок 5 – Удаление товара из базы данных

Пример закрытия базы данных представлен на рисунке 6.

```
1 - Create DB
2 - Open DB
3 - Close DB
4 - Delete DB
5 - Add new good to DB
6 - Search in DB
7 - Remove goods from DB
8 - Exit

Select an option: 3

DB closed.
```

Рисунок 6 – Закрытие базы данных

Пример удаления базы данных представлен на рисунке 7.

```
1 - Create DB
2 - Open DB
3 - Close DB
4 - Delete DB
5 - Add new good to DB
6 - Search in DB
7 - Remove goods from DB
8 - Exit

Select an option: 4

DB deleted.
```

Рисунок 7 – Удаление базы данных

Ниже представлен код данной программы:

*global domains*

*id=integer*

*name=string*

*price=real*

*creatorCompany=string*

*goods=goods(id, name, price, creatorCompany)*

*% db\_selector = goodsDB*

*predicates*

*nondeterm start*

*nondeterm repeat*

*nondeterm selectOption(integer)*

*nondeterm createDB*

*nondeterm openDB*

*nondeterm closeDB*

*nondeterm deleteDB*

*nondeterm addGoods*

*nondeterm searchInDB*  
*nondeterm searchInDB(name, ref)*  
*nondeterm removeGoods*

*goal*

*start.*

*clauses*

*start:- repeat,*  
    *write ("\n1 - Create DB"), nl,*  
    *write ("2 - Open DB"), nl,*  
    *write ("3 - Close DB"), nl,*  
    *write ("4 - Delete DB"), nl,*  
    *write ("5 - Add new good to DB"), nl,*  
    *write ("6 - Search in DB"), nl,*  
    *write ("7 - Remove goods from DB"), nl,*  
    *write ("8 - Exit"), nl,*  
    *write ("\nSelect an option: "), readint(Option), nl,*  
    *selectOption(Option).*

*selectOption(1):- createDB, !,fail.*  
*selectOption(2):- openDB, !,fail.*  
*selectOption(3):- closeDB, !,fail.*  
*selectOption(4):- deleteDB, !,fail.*  
*selectOption(5):- addGoods, !, fail.*  
*selectOption(6):- searchInDB, !, fail.*  
*selectOption(7):- removeGoods, !, fail.*  
*selectOption(8):- closeDB.*

*createDB:-*  
    *existfile("lab3\_1.bin"),*  
    *write("DB already exists!"), nl.*

*createDB:-*  
    *db\_create(goodsDB, "lab3\_1.bin",in\_file),*

```

    bt_create(goodsDB, "treegoods", Bsel, 15,3),
    write("DB created."), nl,
    bt_close(goodsDB, Bsel),
    db_close(goodsDB).

```

*openDB:-*

```

    existfile("lab3_1.bin"),
    db_open(goodsDB, "lab3_1.bin", in_file),
    write("DB opened."), nl.

```

*closeDB:-*

```

    db_close(goodsDB),
    write("DB closed."), nl.

```

*closeDB:-*

```

    write("DB is not opened!"), nl.

```

*deleteDB:-*

```

    existfile("lab3_1.bin"),
    db_delete("lab3_1.bin", in_file),
    write("DB deleted."), nl.

```

*deleteDB:-*

```

    write("DB doesn't exist!"), nl.

```

*addGoods:-*

```

    write("Enter id: "), readint(ID), nl,
    write("Enter name: "), readln(Name), nl,
    write("Enter Price: "), readreal(Price), nl,
    write("Enter creator company: "), readln(CreatorCompany), nl,
    chain_inserta(goodsDB,"goods",goods,goods(ID,    Name,    Price,
CreatorCompany), Ref),
    bt_open(goodsDB, "treegoods", Bsel),
    key_insert(goodsDB, Bsel, Name ,Ref),
    bt_close(goodsDB, Bsel),
    write("Good added."), nl.

```

*addGoods:-*

*write("You should open DB first!").*

*searchInDB(Name, Ref):-*

*bt\_open(goodsDB, "treegoods", BSel),  
key\_search(goodsDB, BSel, Name, TermRef),  
Ref = TermRef,  
bt\_close(goodsDB, BSel).*

*searchInDB:-*

*write("\nEnter goods name to search: "),  
readln(Name), nl,  
searchInDB(Name, Ref),  
ref\_term(goodsDB, goods, Ref, Term),  
Term=goods(ID, \_, Price, CreatorCompany),  
write("Found goods: (", ID, ", ", Name, ", ", Price, ", ",  
CreatorCompany, ")"), nl.*

*removeGoods:-*

*bt\_open(goodsDB, "treegoods", BSel),  
write("\nEnter goods name to remove from DB: "),  
readln(Name), nl,  
searchInDB(Name, Ref),  
key\_delete(goodsDB, BSel, Name, Ref),  
bt\_close(goodsDB, BSel),  
term\_delete(goodsDB, "goods", Ref),  
write("Goods with name '", Name, "' removed from DB"), nl.*

*repeat.*

*repeat:- repeat.*

## ВЫВОД

В ходе выполнения данной лабораторной работы я изучил механизмы работы с базами данных в системе *VISUAL PROLOG(VIP)*, а также ознакомился с операциями создания, открытия, поиска и изменения информации в базе данных и самостоятельно создал учебную программу для закрепления навыков программирования.