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).

Закрытие 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 комманада 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)закрываетдерево.

Следующая команда удаляет *B*-дерево: *bt_delete*(селектор, имя_дерева).

Для открытия B-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 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 файде *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 оператора *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 данный и *B*-дерево в случае возвращения отрицательного значения. Создание базы данный 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 возвращения 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 веденной 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 преданного 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 введённого значения в переменную *Name*, передачу данного значения, а также переменной *Ref* в предикат *searchInDB*, получение функтора из ссылки, занесённой 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 введённого значение в переменную *Name* с помощью оператора *readln*(), поиск в базе данных товара по введённом наименованию с помощью предиката *searchInDB*(*name,* r*ef*) 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.

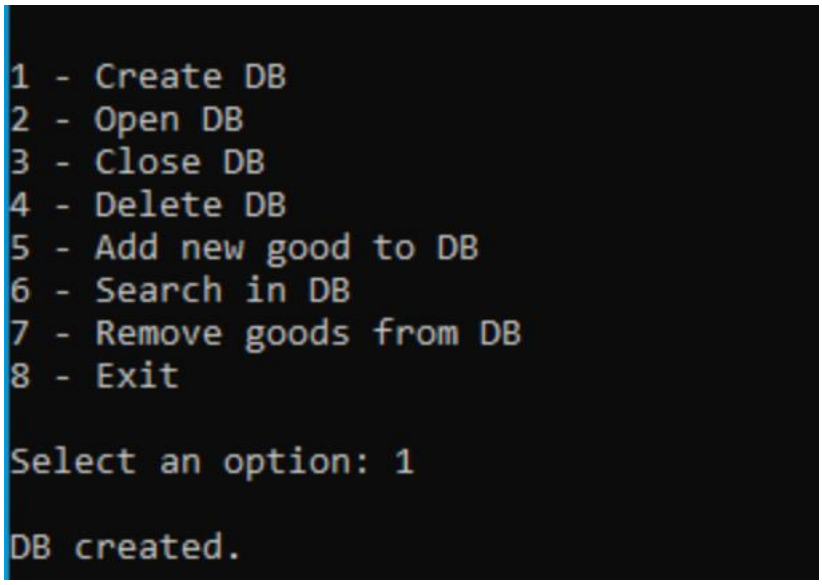

Рисунок 1 – Создание базы данных

Пример открытия базы данных представлен на рисунке 2.

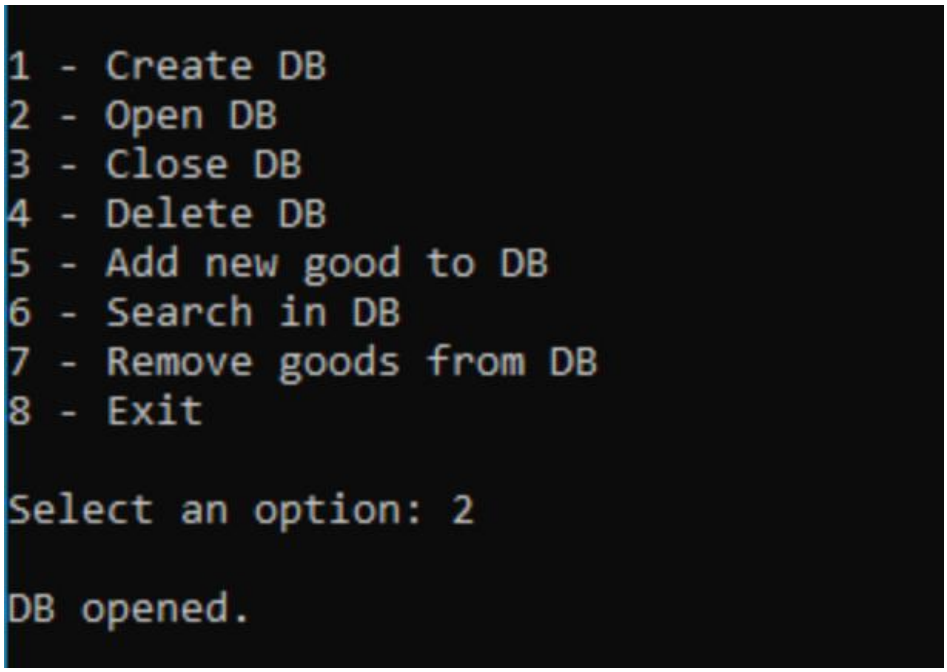

Рисунок 2 – Открытие базы данных

Пример открытия базы данных представлен на рисунке 3.

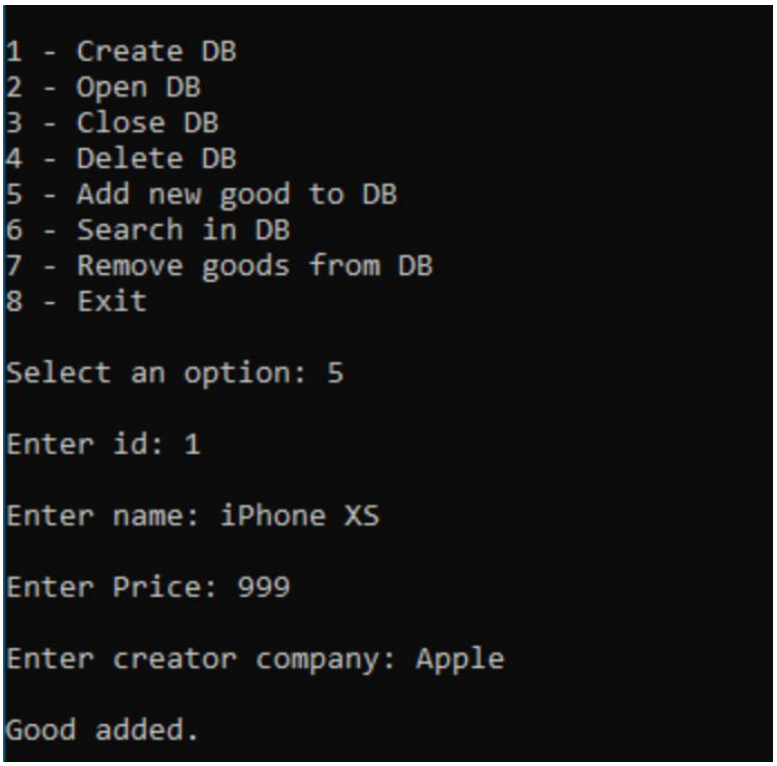

Рисунок 3 – Добавление товара в базу данных

Пример поиска в базе данных представлен на рисунке 4.

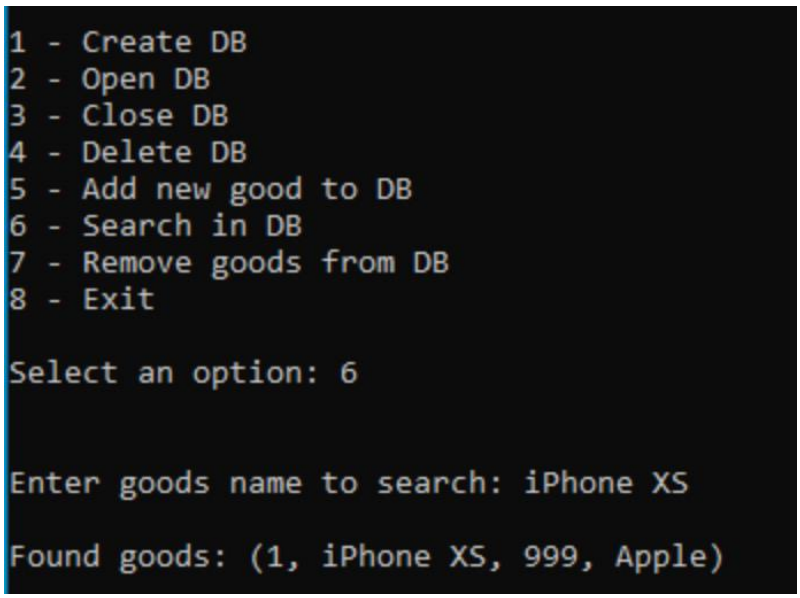

Рисунок 4 – Поиск товара в базе данных

Пример удаления товара из базы данных представлен на рисунке 5.

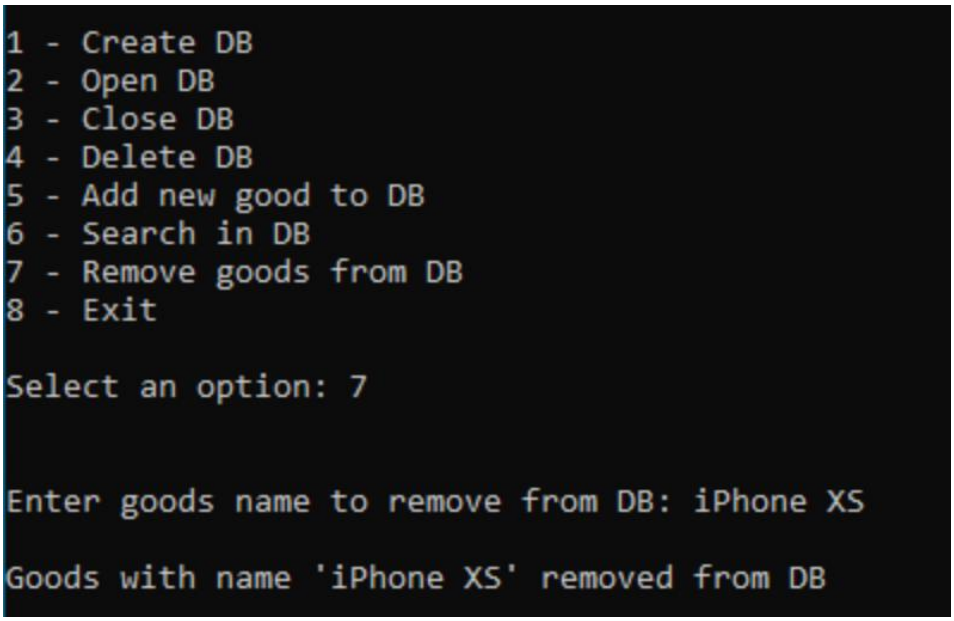

Рисунок 5 – Удаление товара из базы данных

Пример закрытия базы данных представлен на рисунке 6.

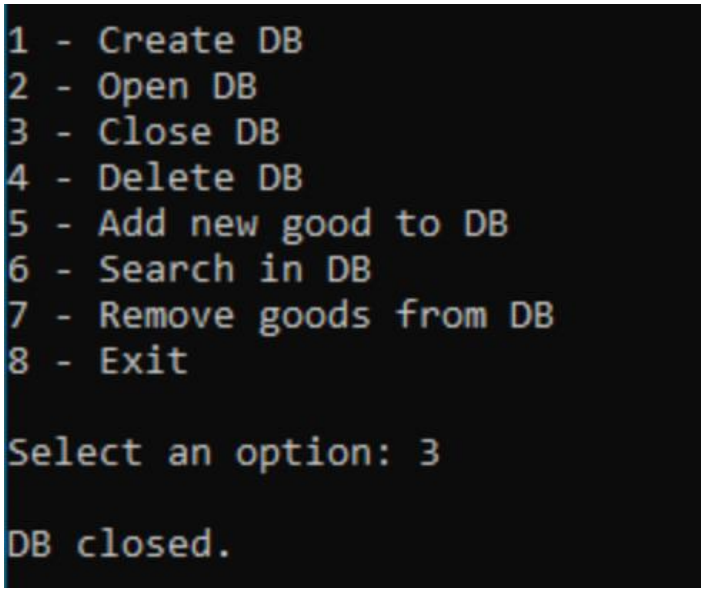

Рисунок 6 – Закрытие базы данных

Пример удаления базы данных представлен на рисунке 7.

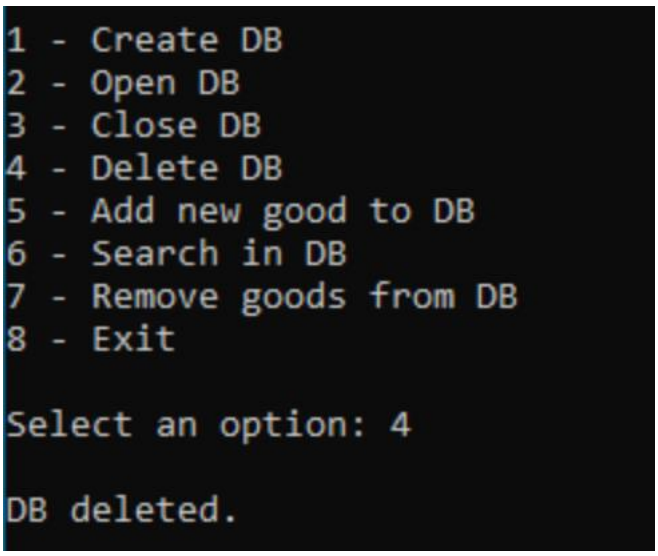

Рисунок 7 – Удаление базы данных

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.

.