Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и информатики»

(СибГУТИ)

Кафедра прикладной математики и кибернетики

РАСЧЕТНО-ГРАФИЧЕСКОЕ ЗАДАНИЕ

по дисциплине «Функциональное и логическое программирование»

Вариант 7

Выполнил:

студент группы ИС-242

Любицкий М. Е.

ФИО студента

Работу проверил: Сороковых Д. А.

ФИО преподавателя

Новосибирск 2024 г.

**Олавление**

[**Задание** 3](#_Toc184029338)

[**Выполнение** 4](#_Toc184029339)

[**Листинг программы** 7](#_Toc184029340)

**Задание**

1.7. Осуществите циклический сдвиг в списке на n элементов влево.

2.7. В каждой строке текстового файла переставьте слова в обратном порядке. Сформируйте новый файл.

3.7. Создайте базу данных о товарах: наименование товара, фасовка, стоимость. Найдите все товары с минимальной стоимостью.

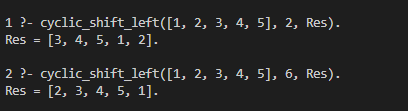
**Выполнение**

**Задача 1: Циклический сдвиг списка на N элементов влево**

**Ход выполнения:**

1. Реализован предикат cyclic\_shift\_left/3, принимающий:
   * Входной список,
   * Количество элементов для сдвига,
   * Результирующий список после сдвига.
2. Предикат построен следующим образом:
   * Сначала вычисляется остаток от деления N на длину списка (ModN), чтобы избежать избыточных операций.
   * Затем список разбивается на две части: первые ModN элементов и оставшуюся часть.
   * Результирующий список формируется путём конкатенации оставшейся части и первых ModN элементов.

**Тестирование:**

****

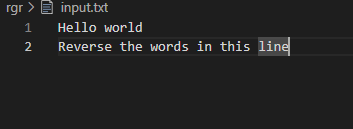
**Задача 2: Перестановка слов в строках текстового файла**

**Ход выполнения:**

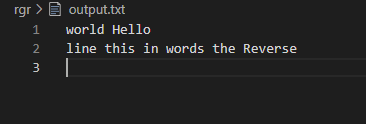
1. Реализован предикат reverse\_words\_in\_file/2, который:
   * Считывает строки из исходного файла.
   * Для каждой строки:
     + Разбивает её на слова,
     + Переворачивает порядок слов,
     + Собирает строку обратно.
   * Записывает результаты в новый файл.
2. Использованы стандартные предикаты для работы с файлами (open/3, read\_line\_to\_string/2, write/2) и обработки строк.

**Тестирование:**

Входной файл:



Выходной файл:



Запуск программы



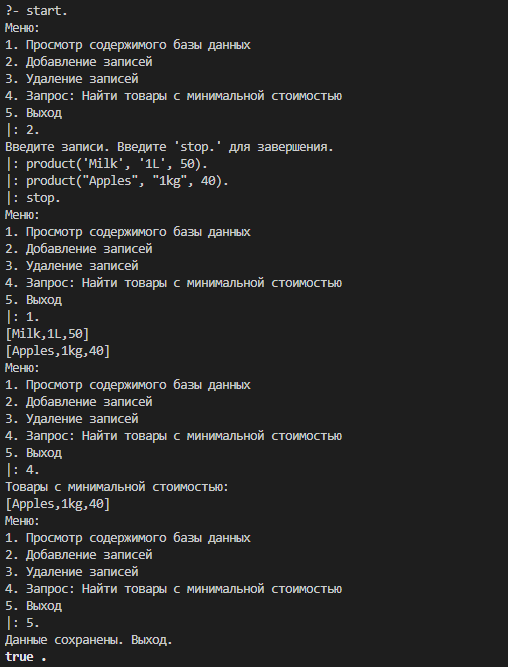
**Задача 3: Работа с базой данных товаров**

1. Загружать базу данных из файла.
2. Обеспечивать работу с меню, включая просмотр, добавление, удаление записей и выполнение запроса.
3. Сохранять базу данных обратно в файл.

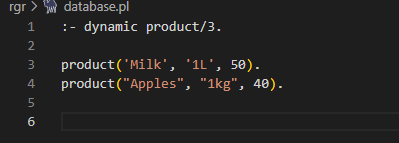
**Ход выполнения:**

1. **Структура данных:**
   * Записи базы данных представлены динамическими фактами product(Name, Packaging, Price).
2. **Функционал:**
   * **Просмотр базы данных:** Предикат view\_database перебирает все записи и выводит их в удобном формате.
   * **Добавление записей:** Предикат add\_records позволяет вводить новые факты, используя рекурсивное чтение до команды stop..
   * **Удаление записей:** Предикат delete\_records удаляет указанные записи аналогично добавлению.
   * **Запрос:** Предикат find\_min\_price находит товары с минимальной стоимостью.
   * **Сохранение:** Все изменения записываются в файл через save\_database.
3. **Меню:**
   * Реализовано через предикат menu, предоставляющий пользователю интерфейс выбора действий.

**Тестирование:**

****

Содержимое базы данных:



**Листинг программы**

**Задача 1:**

#### cyclic\_shift\_left(List, N, Result) :-

#### length(List, Len),

#### Shift is N mod Len,

#### shift\_left(List, Shift, Result).

#### shift\_left(List, 0, List) :- !.

#### shift\_left([H|T], N, Result) :-

#### append(T, [H], Temp),

#### N1 is N - 1,

#### shift\_left(Temp, N1, Result).

**Задача 2:**

#### reverse\_words\_in\_file(InputFile, OutputFile) :-

#### open(InputFile, read, InStream),

#### open(OutputFile, write, OutStream),

#### process\_lines(InStream, OutStream),

#### close(InStream),

#### close(OutStream).

#### process\_lines(InStream, OutStream) :-

#### read\_line\_to\_string(InStream, Line),

#### (   Line \= end\_of\_file

#### ->  split\_string(Line, " ", "", Words),

#### reverse(Words, ReversedWords),

#### atomic\_list\_concat(ReversedWords, " ", ReversedLine),

#### writeln(OutStream, ReversedLine),

#### process\_lines(InStream, OutStream)

#### ;   true

#### ).

**Задача 3:**

#### :- dynamic product/3.

#### load\_database(File) :-

#### exists\_file(File), !,

#### consult(File).

#### load\_database(\_).

#### save\_database(File) :-

#### tell(File),

#### listing(product/3),

#### told.

#### view\_database :-

#### forall(product(Name, Packaging, Price),

#### writeln([Name, Packaging, Price])).

#### add\_records :-

#### writeln("Введите записи. Введите 'stop.' для завершения."),

#### read\_record.

#### read\_record :-

#### read(Product),

#### (   Product == stop

#### ->  true

#### ;   assertz(Product),

#### read\_record

#### ).

#### delete\_records :-

#### writeln("Введите записи для удаления. Введите 'stop.' для завершения."),

#### read\_deletion.

#### read\_deletion :-

#### read(Product),

#### (   Product == stop

#### ->  true

#### ;   retract(Product),

#### read\_deletion

#### ).

#### find\_min\_price :-

#### findall(Price, product(\_, \_, Price), Prices),

#### min\_list(Prices, MinPrice),

#### findall([Name, Packaging, MinPrice], product(Name, Packaging, MinPrice), Results),

#### writeln("Товары с минимальной стоимостью:"),

#### forall(member(Result, Results), writeln(Result)).

#### menu :-

#### writeln("Меню:"),

#### writeln("1. Просмотр содержимого базы данных"),

#### writeln("2. Добавление записей"),

#### writeln("3. Удаление записей"),

#### writeln("4. Запрос: Найти товары с минимальной стоимостью"),

#### writeln("5. Выход"),

#### read(Choice),

#### handle\_choice(Choice).

#### handle\_choice(1) :- view\_database, menu.

#### handle\_choice(2) :- add\_records, menu.

#### handle\_choice(3) :- delete\_records, menu.

#### handle\_choice(4) :- find\_min\_price, menu.

#### handle\_choice(5) :- save\_database('database.pl'), writeln("Данные сохранены. Выход."), !.

#### handle\_choice(\_) :- writeln("Неверный выбор, попробуйте снова."), menu.

#### start :-

#### load\_database('database.pl'),

#### menu.