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 ИВ-221
Гордов Роман Сергеевич

Работу проверила:

Сороковых Дарья Анатольевна

Новосибирск 2024 г.

Цель работы:

Создать программы на языке swi-prolog, которые решают следующие задачи:

1. Осуществление циклического сдвига в списке на n элементов влево.
2. Перестановка слов в каждой строке текстового файла в обратном порядке и сохранение результата в новый файл.
3. Создание базы данных товаров и нахождение товаров с минимальной стоимостью.

Задача 1: Циклический сдвиг списка влево на n элементов

Необходимо выполнить циклический сдвиг элементов заданного списка влево на n позиций. Например, при сдвиге списка `[1, 2, 3, 4, 5]` на 2 позиции результат должен быть `[3, 4, 5, 1, 2]`.

```
shift_left(List, N, Result) :-
    length(List, Len),
    Shift is N mod Len,
    append(Left, Right, List),
    length(Left, Shift),
    append(Right, Left, Result).
```

Стоит пояснить в prolog мы не вычисляем значения переменных, как в процедурных языках, а задаём логические отношения между переменными, а значит когда мы говорим `append(Left, Right, List)` и добавляем `length(Left, Shift)`, prolog гарантирует, что:

Первые Shift элементов попадут в `Left`.

Остальная часть списка останется в `Right`.

```
?- shift_left([1,2,3,4,5], 2, Result).
Result = [3, 4, 5, 1, 2] .
```

Задача 2: Перестановка слов в строках файла в обратном порядке

Требуется считать строки из текстового файла, изменить порядок слов в каждой строке на обратный и сохранить результат в новый файл.

```
reverse_words(Line, ReversedLine) :-
    split_string(Line, " ", "", Words),
    reverse(Words, ReversedWords),
    atomic_list_concat(ReversedWords, " ", ReversedLine).
```

Разбиваем строку на список слов Words, используя пробел как разделитель. Переворачиваем список и обратно склеиваем соединяя их пробелами.

```
reverse_file(InputFile, OutputFile) :-
    open(InputFile, read, InStream),
    open(OutputFile, write, OutStream),
    process_lines(InStream, OutStream),
    close(InStream),
    close(OutStream).
```

Открываем рабочие файлы. Записываем из входного потока в выходной. Закрываем файлы.

```
process_lines(InStream, OutStream) :-
    read_line_to_string(InStream, Line),
    (   Line = end_of_file ->  true  ;   reverse_words(Line, ReversedLine),
        writeln(OutStream, ReversedLine),
        process_lines(InStream, OutStream)
    ).
```

Считываем строку из файла. Проверяем не достигнут ли конец файла. Переворачиваем слова в строке и записываем ее в выходной файл.

```
?- reverse_file('rgr.pl', 'output.txt').
true.
```

sibsutis > flp2024 > swi-prolog > output.txt

```
1   :- Result) N, shift_left(List,
2   Len), length(List,
3   Len, mod N is Shift
4   List), Right, append(Left,
5   Shift), length(Left,
6   Result). Left, append(Right,
7
8   :- ReversedLine) reverse_words(Line,
9   Words), "", ", " split_string(Line,
10  ReversedWords), reverse(Words,
11  ReversedLine). ", " atomic_list_concat(ReversedWords,
12
```

Задача 3: База данных товаров

Необходимо создать базу данных товаров, включающую наименование, фасовку и стоимость, а затем найти товары с минимальной стоимостью.

```
:- dynamic product/2.


load_db(FileName) :-
(exists_file(FileName) ->
consult(FileName),
write('Database successfully loaded.'), nl
;
write('Database file does not exist, new file created.'), nl
).

Функция для загрузки файла.
save_db(FileName) :-
tell(FileName),
listing(product/2),
told.

Функция для сохранения файла.
view_db :-
(product(_, _) ->
listing(product/2)
;
write('Database is empty.'), nl
).

Просмотр базы данных.
add_products :-
write('Enter product to add ( product(Name, Price) ). Enter "stop." to break: '),
nl,
read(Product),
(Product \= stop ->
assertz(Product),
add_products
).

Функция для создания динамических предикатов.
delete_products :-
write('Enter product to delete ( product(Name, Price) ). Enter "stop." to break: '),
nl,

read(Product),
(Product \= stop ->
(retract(Product) ->
write('Product deleted'),
```

```
nl
;
write('Product not found: '),
nl
),
delete_products
).
```

Функция для удаления предиката.

```
min_price :-
findall(Price, product(_, Price), Prices),
(Prices \= [] ->
min_list(Prices, MinPrice),
findall(Name, (product(Name, Price), Price == MinPrice), Result),
write(Result), nl
;
write('Database is empty.'), nl
).
```

Поиск минимальных цен.

```
main_menu(FileName) :-
repeat,

write('Commands:'), nl,
write('1. Print data'), nl,
write('2. Add item'), nl,
write('3. Delete item'), nl,
write('4. Find less price'), nl,
write('5. Exit'), nl,

read(Choice),
nl,

handle_choice(Choice, FileName),
Choice = 5.

handle_choice(1, _) :- view_db, !.
handle_choice(2, _) :- add_products, !.
handle_choice(3, _) :- delete_products, !.
handle_choice(4, _) :- min_price, !.
handle_choice(5, FileName) :- save_db(FileName), !.
handle_choice(_, _) :- write('Invalid command.'), nl, fail.
```

Меню.

```
main :-
write('Enter Database filename: '),
read(FileName),
load_db(FileName),
main_menu(FileName).
Main.
```

Вывод:

В результате выполнения работы были реализованы три программы на языке swi-prolog, успешно решающие поставленные задачи:

1. Реализован циклический сдвиг элементов списка.
2. Реализована обработка текстового файла для перестановки слов в строках в обратном порядке.
3. Создана база данных товаров с возможностью поиска товаров с минимальной стоимостью.