|  |  |  |
| --- | --- | --- |
| Министерство науки и высшего образования  Российской Федерации | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования | | |
| «Новосибирский государственный технический университет» | | |
|  | | |
| Теоретической и прикладной математики | | |
|  | | |
| Лабораторная работа № 6 | | |
| по дисциплине «Логическое программирование» | | |
| **Предикаты ввода-вывода, предикаты обработки строк** | | |
|  | | |
|  | | |
|  | Факультет: | ПМИ |
| Группа: | ПМИ-02 |
| Вариант: | 7 |
| Студент: | Сидоров Даниил, |
|  | Дюков Богдан |
| Преподаватель: | Авдеенко Т. В., |
|  | Целебровская М. Ю.  . |
|
|  |  |
|  | | |
| Новосибирск | | |
| 2022 | | |

1. **Цель работы**

Изучить особенность использования предикатов данных типов и получить навык использо­вания этих предикатов при проектировании программ на Прологе.

1. **Эксперимент**

**Предикаты ввода вывода:**

Предикаты ввода с клавиатуры или с файла:

readln - для чтения строк символов целиком.

*readint, readreal, readchar -* для чтения целых, вещественных и символьных значений соответственно.

*readterm-* для чтения составных объектов.

Предикат вывода:

*write (X1, X2, …, Xn)* – бесформатный вывод.

*writef (Format\_string, X1, X2, …, Xn)* – форматированный вывод.

Предикаты определения устройства ввода и вывода:

*readdevice(symbolic\_name\_of\_file)* - переопределение текущего устройства ввода.

*writedevice(symbolic\_name\_of\_file)* – переопределение текущего устройства вывода.

Открытие/закрытие файла для записи/чтения:

*openread (symbolic\_name\_of\_file, dos\_file\_name)* – открывает файл для записи.

*openappend (symbolic\_name\_of\_file, dos\_file\_name)* – открывает файл для записи в конец файла.

*openmodify (symbolic\_name\_of\_file, dos\_file\_name)* – открывает файл для записи и чтения.

*closefile(symbolic\_name\_of\_file)* – закрывает указанный файл.

Вспомогательные предикаты:

*eof(symbolic\_name\_of\_file)* – проверка конца файла.

*existfile(dos\_file\_name)* – проверяет наличие файла в каталоге на текущем дисководе.

**Предикаты работы со строками:**

***frontchar (String1, Char, String2)****,* где *String1 = Char+String2*;

frontchar можно использовать для расщепления строки в последовательность символов или для создания строки из последовательности символов, а также для проверки символов в строке.

***fronttoken (String1, Token, Rest)****,* где *String1* – исходная строка, *Token* – первая лексема строки, *Rest* – остаток строки без первой лексемы, *String1= Token+Rest*.

В случае потока (i,o,o) fronttoken находит первую лексему String1, связывает ее с Token, а остаток String1 связывает с Rest.

Варианты потока (i,i,o),(i,o,i), а также (i,i,i) служат для проверки; если связанные аргументы соответствуют частям String1 (первой лексеме, всему, что находится после первой лексемы, или же им обоим), то fronttoken завершается успешно, в противном случае - неудачно.

В случае, если использован поток (o,i,i), предикат создает объединение Token и Rest, связывая String1 с результатом.

***frontstr (Length, InpString, StartString, RestString)*** - назначает первые Length символов строки InpString в строку StartString, остальные - в строку RestString.

***concat (String1, String, String3)****, String3 = String1 +String2;*

*-* конкатенация

*-* проверка того, что третья строка является конкатенацией первых двух

- нахождение разности двух строк, например неизвестен первый аргумент

***str\_len (String, integer\_length)****, String* - строка, *integer\_length* - длина строки.

Определяет длину строки или проверяет, является ли заданная строка заданной длины.

1. **Задание**

7. Даны два символьных файла *F1* и *F2*, Файл *F1* содержит произ­вольный текст. Слова в тексте разделены пробелами и знаками препи­нания. Файл *F2* содержит не более сорока слов, которые разделены запятыми. Эти слова образуют пары; каждое первое слово считается заменяемым, каждое второе заменяющим. Найти в файле *F1* все заменяемые слова и заменить их на соответствующие заменяющие. Резуль­тат поместить в файл *G*.

**Листинг программы**

domains

file = input; output; changeInput

str = string

list = str\*

predicates

nondeterm substitution(string,string,string)

nondeterm modifiedStr(list, list)

nondeterm readChangeWords(file,list)

nondeterm makeChangeList(list,string)

nondeterm strToList(str,list)

nondeterm repeat

nondeterm substitutionStr(list,list,list)

pairSearch(list,str, str)

nondeterm writeStr(list, file)

nondeterm search(str, list)

nondeterm reverse(list,list)

nondeterm append(list,list,list)

nondeterm add(str,list,list)

clauses

repeat.

repeat:-repeat.

substitution(Input,Output,Change) :- makeChangeList(ListChange,Change),openread(input, Input),openwrite(output,Output), writedevice(output),readdevice(input),

repeat,readln(Str),strToList(Str, List), modifiedStr(List, ListChange),eof(input),closefile(input), closefile(output).

modifiedStr(List,ListChange):-substitutionStr(List, ListChange, []).

substitutionStr([],\_,Result):-writeStr(Result,output).

substitutionStr([Head|Tail],L2,Result):-search(Head,L2),!,pairSearch(L2,Head,Replace),add(Replace,Result,D),substitutionStr(Tail,L2,D);add(Head,Result,K),substitutionStr(Tail,L2,K).

makeChangeList(List,Change):-openread(changeInput, Change),readdevice(changeInput),readChangeWords(changeInput,List),closefile(changeInput).

readChangeWords(File,Result) :-not(eof(File)),!,readln(Str),strToList(Str,Result).

strToList("",[]):-!.

strToList(Str,[Head|Tail]) :- fronttoken(Str,Head,Str1),!, strToList(Str1,Tail).

writeStr([Head|Tail],Output):-write(Head),write(" "),writeStr(Tail,Output).

writeStr([],\_):-nl.

pairSearch([H,H1,\_|Tail],X,Y):-H<>X,!,pairSearch(Tail,X,Y);Y=H1.

search(X,[X,\_,\_|\_]).

search(X,[\_,\_,\_|Tail]):-search(X,Tail).

reverse([],[]).

reverse([Head|Tail],Result):-reverse(Tail,Reverse),append(Reverse,[Head],Result).

append([],L,L).

append([Head|L1],L2,[Head|L3]):-append(L1,L2,L3).

add(Str,L1,Result):-reverse(L1,Reverse),reverse([Str|Reverse],Result).

1. **Результаты выполнения**

|  |  |  |  |
| --- | --- | --- | --- |
| **Исходный текст** | **Слова заменители** | **Полученный текст** | **Комментарий** |
| Hello world  It's very good weather | Hello Goodbye, good bad, | Goodbye world  It ' s very bad weather | Все верно |
| Hello world  It's very good weather | - | - | При не заданных словах-заменителях правило не выполнится |
| Hello world  It's very good weather  My bad car is red | Hello Goodbye, good bad, | Goodbye world  It ' s very bad weather  My bad car is red | В тексте встречается слово-заменитель. Все верно. |
| - | Hello Goodbye, good bad, | - | Без исходного текста программа не выполнится |

1. **Описание задач, решаемых данной программой**

1)Разделение строки на слова и добавление каждого слова в список.

strToList("",[]):-!.

strToList(Str,[Head|Tail]) :- fronttoken(Str,Head,Str1),!, strToList(Str1,Tail).

2)Если элемент не равен Х, идем дальше по списку, иначе берем слово-заменитель.

pairSearch([H,H1,\_|Tail],X,Y):-H<>X,!,pairSearch(Tail,X,Y);Y=H1.

3)Запись списка в файл.

writeStr([Head|Tail],Output):-write(Head),write(" "),writeStr(Tail,Output).

writeStr([],\_):-nl.

4)Пока файл не пуст, считываем слова и добавляем их в список.

readChangeWords(File,Result) :-not(eof(File)),!,readln(Str),strToList(Str,Result).

5)Создание списка из пар слов.

makeChangeList(List,Change):-openread(changeInput, Change),readdevice(changeInput),readChangeWords(changeInput,List),closefile(changeInput).

6)Ищем есть ли для текущего слова слово-заменитель. Если находим этот заменитель, добавляем слово заменить в список, иначе вставляем изначальное слово в список.

modifiedStr(List,ListChange):-substitutionStr(List, ListChange, []).

substitutionStr([],\_,Result):-writeStr(Result,output).

substitutionStr([Head|Tail],L2,Result):-search(Head,L2),!,pairSearch(L2,Head,Replace),add(Replace,Result,D),substitutionStr(Tail,L2,D);add(Head,Result,K),substitutionStr(Tail,L2,K).

6)Чтение входных данных, преобразование строки в список, изменение исходного строки и запись измененной строки в файл. И так с каждой строкой.

substitution(Input,Output,Change) :- makeChangeList(ListChange,Change),openread(input, Input),openwrite(output,Output), writedevice(output),readdevice(input),repeat,readln(Str),strToList(Str, List), modifiedStr(List, ListChange),eof(input),closefile(input), closefile(output)