**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

отчет

**по лабораторной работе №1**

**по дисциплине «Алгоритмы и структуры данных»**

Тема: Рекурсия

|  |  |  |
| --- | --- | --- |
| Студент гр. 7381 |  | Павлов А.П. |
| Преподаватель |  | Фирсов М.А. |

Санкт-Петербург

2018

**Цель работы.**

Ознакомиться с основными методами использования рекурсии и написать программу с использованием рекурсии.

**Задание.**

Вариант 15.

Построить синтаксический анализатор для понятия скобки.

скобки::=А | A ( ряд\_скобок )

ряд\_скобок::= скобки | скобки ; ряд\_скобок

**Основные теоретические положения.**

Рекурсия — определение, описание, изображение какого-либо объекта или процесса внутри самого этого объекта или процесса, то есть ситуация, когда объект является частью самого себя.

**Ход работы:**

Программа посимвольно считывает входные данные и проверяет их на принадлежность к данной грамматике. Соответствующие сообщения об ошибке или корректности считанной строки выводятся на консоль. В ходе выполнения программы на консоль отображается глубина рекурсии.

*Функция* ***int brackets****(*int index, int \* checkCallRows***)***

Аргументы:

index – индекс отступа, который увеличивается, чтобы при демонстрации следующего входа в функцию отступ был больше.

checkCallRows – переменная, необходимая для того, чтобы была ли вызвана функция ***rowsBrackets*** раннее.

Возвращаемое значение:

0 – если обрабатываемый символ не принадлежит грамматике.

1 – если обрабатываемый символ принадлежит грамматике.

Описание:

Если текущий символ – 'A', то считывается следующий символ и проверяется по следующим условиям:

1. Если текущий символ – '(', то вызывается функция ***rowsBrackets***. Если последняя функция возвращает 1, то считывается и следующий символ и проверяется. В случае ')' функция возвращает 1, иначе вызывается функция errorMessage с параметром 3, и функция возвращает 0.
2. Если текущий символ является символом перевода строки('\n') или концом файла(EOF), то символ возвращается в поток ввода, и функция возвращает 1.
3. Если текущий символ является или ';', или ')', то совершается проверка значения переменной checkCallRows. Если последняя не равна 0, то в поток возвращается текущий символ и возвращается 1. В ином случае выводится сообщение об ошибке и возвращается 0.

В иных случаях выводится сообщение об ошибке с параметром 1 и возвращается 0.

*Функция* ***int rowsBrackets(****int index, int \* checkCallRows, char a****)***

*Рекурсивная функция.*

Аргументы:

index – индекс отступа, который увеличивается, чтобы при демонстрации следующего входа в функцию отступ был больше.

a – текущий символ на проверку.

checkCallRows – переменная, необходимая для того, чтобы узнать была ли функция **rows*Brackets*** вызвана раннее.

Возвращаемое значение:

0 – если обрабатываемый символ не принадлежит грамматике.

1 – если обрабатываемый символ принадлежит грамматике.

Описание:

Вызывается функция ***brackets***. Если ***brackets*** возвращает 1, то считывается следующий символ проверятся, является ли он ';'. Если да, то вызывается функция ***rowBrackets***, если последняя возвращают 1, то и первая функция возвращает 1. В противном случае считанный символ возвращается в поток, и функция возвращает 1.

В иных случаях возвращается 0.

*Функция* ***void errorMessage****(enum err\_num numberError, int index)*

Аргументы:

numberError – код ошибки.

tabs\_count – индекс отступа, который увеличивается, чтобы при демонстрации следующего входа в функцию отступ был больше.

Возвращаемое значение:

Функция ничего не возвращает.

Описание:

Вызывается функция печати отступов ***print\_tabs***. В зависимости от значения numberError выводится соответствующая информация об ошибке.

*Функция* ***void print\_tabs****(index)*

Аргументы:

index – индекс отступа, который увеличивается, чтобы при демонстрации следующего входа в функцию отступ был больше.

Возвращаемое значение:

Функция ничего не возвращает.

*Функция* ***int main()***

Описание:

В главной функции программы происходит вызов функции ***brackets*** и проверка, удовлетворены ли все ее условия, то есть, является ли выражение *скобками* или нет. В первом случае выводится «It is brackets», в противном «It is not brackets».

**Тестирование программы:**

Файлы с тестовыми данными, находящиеся в директории **Tests**. Они служат для проверки функционала и работоспособности написанной программы.

|  |  |  |
| --- | --- | --- |
| № теста | Входные данные | Результат работы программы |
| 1 | A(A;A) | It is brackets |
| 2 | A(A(A);A(A)) | It is brackets |
| 3 | A; | It is not brackets  ERROR: Missing '(' |
| 4 | A(A; | It is not brackets  ERROR: Unknown symbol |
| 5 | A(A;A;A(A)) | It is brackets |
| 6 | A(A(();())) | It is not brackets  ERROR: Unknown symbol |

**Выводы.**

В ходе выполнения лабораторной работы получены знания о синтаксическом анализаторе и грамматиках, а также закреплены знания по теме «рекурсия».

Исходный код:

Приложение А: код файла Lab1.c

#include <stdio.h>

int rowsBrackets(int index, int \* checkCallRows, char a);

void print\_tabs(int index){//for print tabs

for(int i = 0; i < index; i++)

printf("\t");

}

enum err\_num{err1 = 1, err2, err3, err4};

void errorMessage(enum err\_num numberError, int index){

print\_tabs(index-1);

switch(numberError){

case 1:

printf("ERROR: Unknown symbol\n");

break;

case 2:

printf("ERROR: Missing '('\n");

break;

case 3:

printf("ERROR: Missing ')'\n");

break;

default:

printf("ERROR: Unkown command\n");

}

}

int brackets(int index, int \* checkCallRows){

int result = 0;

char a = getchar();

print\_tabs(index);

printf("CALL\_BRACKETS with symbol '%c'\n", a);

index++;

if (a == 'A')

{

a = getchar();

if (a == '(')

{

if(rowsBrackets(index, checkCallRows, a))

{

a = getchar();

if(a == ')')

result = 1;

else {

errorMessage(3, index);

result = 0;

}

}

}

else if((a == EOF)||(a == '\n'))//for single A

{

ungetc(a, stdin);

result = 1;

}

else if(a == ';' || a == ')')

{

if((\*checkCallRows)!=0)

{

ungetc(a, stdin);

result = 1;

}

else {

errorMessage(2, index);

result = 0;

}

}

else{

errorMessage(1, index);

result = 0;

}

}

else{

errorMessage(1, index);

result = 0;

}

print\_tabs(index-1);

printf("END\_BRACKETS\n");

return result;

}

int rowsBrackets(int index, int \* checkCallRows, char a){

print\_tabs(tabs\_count);

printf("CALL\_ROW\_BRACKETS with symbol '%c'\n", a);

int result = 0;

(\*checkCallRows) = 1;

index++;

if (brackets(index, checkCallRows))

{

a = getchar();

if (a == ';')

{

if(rowsBrackets(index, checkCallRows, a))

result = 1;

}

else{

ungetc(a, stdin);

result = 1;

}

}

print\_tabs(index-1);

printf("END\_ROW\_BRACKETS\n");

return result;

}

int main(){

int checkCallRows = 0;//to check the function call

int index = 0;//index for tabs

char ch = getchar();

if(ch == '\n' || ch == EOF){

printf("Input empty\n");

return 0;

}

else

ungetc(ch, stdin);

if(brackets(index, &checkCallRows))

printf("It is brackets\n");

else

printf("It isn't brackets\n");

return 0;

}

Приложение Б: код файла compile.sh

#!/bin/bash

gcc ./Source/Lab1.c -o Lab1

echo 'Test 1:'

cat ./Tests/Test1.txt

echo -e '\n'

./Lab1 < ./Tests/Test1.txt

echo -e ''

echo 'Test 2:'

cat ./Tests/Test2.txt

echo -e '\n'

./Lab1 < ./Tests/Test2.txt

echo -e ''

echo 'Test 3:'

cat ./Tests/Test3.txt

echo -e '\n'

./Lab1 < ./Tests/Test3.txt

echo -e ''

echo 'Test 4:'

cat ./Tests/Test4.txt

echo -e '\n'

./Lab1 < ./Tests/Test4.txt

echo -e ''

echo 'Test 5:'

cat ./Tests/Test5.txt

echo -e '\n'

./Lab1 < ./Tests/Test5.txt

echo -e ''

echo 'Test 6:'

cat ./Tests/Test6.txt

echo -e '\n'

./Lab1 < ./Tests/Test6.txt