МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Алгоритмы и структуры данных»

Тема: Рекурсия

Студент гр. 9382 _____ Дерюгин Д.А.

Преподаватель _____ Фирсов М.А.

Санкт-Петербург

2020

Цель работы.

Ознакомится с понятием рекурсии, написать синтаксический анализатор, проверяющий корректность строчки, научится писать рекурсивные функции на языке C++.

Задание.

Вариант 6.

Построить синтаксический анализатор для понятия *простое выражение*. *простое_выражение::=простой_идентификатор | (простое_выражение знак_операции простое_выражение)*

простой_идентификатор::= буква

*знак_операции:: = – | + | **

Ход работы.

Общее описание функций.

В функции main объявляем переменную fileName типа string, в которую записываем имя файла, с которого считать проверяемую строку.

Открываем файл и передаём его содержимое в функцию isSimpleExpresson. В данной функции объявляем переменные:

letter - символ, который проверяется в данным момент;

flag - переменная будет принимать значение true если проверяемый символ подходит под определение простого выражение, в противном случае переменная принимает значение false;

depth - целочисленная переменная, которая будет хранить в себе глубину рекурсии.

После объявление переменных в функции проверяется корректность первого символа строчки. Если этот символ «(« или же латинская буква, тогда вызывается рекурсивная функция isSimple, которая будет описана ниже: если же это другой символ, тогда вызывается функция printError с кодом ошибки 2.

После отработки функции isSimple, в переменную flag записывается результат. Если flag принимает значение «ложь» или же файл не до конца считан, тогда вызывается printError с кодом ошибки 4.

В конце работы функция isSimpleExpression возвращает булеву переменную.

Функция isSimple. Данная рекурсивная функция принимает на вход сам файл, символ, с которого начинать считывать и глубину рекурсии.

Вначале проверяется: является ли символ латинской буквой. Если это так, то функция завершает свою работу и возвращает true. Если это не так, то проверяется, является ли переданный символ «(». После считывается следующий символ строки и вызывается эта же функция. Так проверяется первая часть простого выражения. После этого, если первый символ корректен, вызывается функция isOperation, которая проверяет корректность знака. Если знак верный, вызывается второй раз функция isSimpleExpression. При данном вызове проверяется корректность третьей части простого выражения. В конце, если все три части простого выражения корректны, проверяется символ «)». В результате, функция isSimpleExpression возвращает булеву переменную.

Функция printError. Служит для вывода сообщения об ошибки. Поддерживает 10 различных ошибок, которые могут произойти в результате выполнения программы. Принимает код ошибки, ничего не возвращает.

Описание алгоритма.

Сначала проверяется является ли простое выражение буквой. Если это ложь, тогда проверяется, первый символ простого выражения(если это не символ открывающийся скобки, то программа выдаёт ошибку и заканчивает работу). После проверяется первая часть простого выражения(функция вызывает сама себя, но с другими значениями и проверяет уже внутреннее выражение. И так до конца вложенности). Затем проверяет корректность знака операции. В конце проверяется третья часть простого выражения(также как и первая часть, вызывая саму себя, но с другими значениями).

Выходные данные.

В конце программы функция main выводит результат работы:

«Its not a Simple Expression» если строка не является простым выражением;

«Its a Simple Expression» если строка является простым выражением.

Тестирование.

Входная строчка	Результат программы	Комментарий
(((f-a)+(a+(a-d)))-((f*g)-g))	Its a Simple Expression	Строчка является простым выражением
-dkslafj	Its not a Simple Expression	Действительно, первый символ должен быть или латинская буква, или «(«.
	Its not a Simple Expression	Это пустая строчка.
(a-a)e	Its not a Simple Expression	После закрывающий скобки есть лишний символ.
(aaa)	Its not a Simple Expression	Это не выражение, потому что знак операции не может быть буквой.
(a-a)I	Its not a Simple Expression	Вторая часть простого выражения не является простым выражением.
(0-a)	Its not a Simple Expression	Первая часть простого выражения не является простым выражением.
(a-a	Its not a Simple Expression	Нет закрывающей скобки в конце выражения.
(((d-t)-(f+(f+(d-g))))-((e+j)-a))	Its a Simple Expression	Действительно, данная строчка является простым выражением.

Выводы.

В данной лабораторной работы мы научились применять рекурсивные функции на языке C++. Сделали программу, которая проверяет, является ли входная строка простым выражением.

ПРИЛОЖЕНИЕ А.
ИСХОДНЫЙ КОД ПРОГРАММЫ.

```
#include <iostream>
#include <string.h>
#include <fstream>
#include <cctype>
using namespace std;

/*Функция вывода ошибок*/
void printError(int numberOfError){
   if (numberOfError == 1) cout<<"\nCannot open file\n";
   if (numberOfError == 2) cout<<"\nIncorrect first symbol\n";
   if (numberOfError == 3) cout<<"\nempty string\n";
   if (numberOfError == 4) cout<<"\nExtra symbols\n";
   if (numberOfError == 5) cout<<"\nIncorrect operation or operation is empty\n";
   if (numberOfError == 6) cout<<"\nIncorrect first simple expression\n";
   if (numberOfError == 7) cout<<"\nIncorrect second simple expression\n";
```

```
    if (numberOfError == 8) cout<<"\nNo symbol ')'\n";
    if (numberOfError == 9) cout<<"\nEmpty simple expression\n";
    if (numberOfError == 10) cout<<"\nNo '(' symbol\n";
}

/*Функция проверки знака операции*/
bool isOperation(ifstream &file, char letter){
    return letter == '+' || letter == '-' || letter == '*';
}

/*Функция проверки простого выражения*/
bool isSimple(ifstream &file, char letter, int depth){
    bool flag;
    /*Является ли простое выражание латинской буквой*/
    if(isalpha(letter)){
        return true;
    }
    else if (letter == '('){
        /*Проверка первой части простого выражения*/
       if (file>>letter){
            if (letter == '(')  for (int i = 0; i < depth+1; i++) cout<<"\t";
            else  for (int i = 0; i < depth; i++) cout<<"\t";
            cout<<letter<<"\n";
            flag = isSimple(file, letter, depth+1);
            /*Проверка второй части(корректность знака оперции) простого
выражения*/
            if (flag){
                if (file>>letter){
                    for (int i = 0; i < depth; i++) cout<<"\t";
                    cout<<letter<<"\n";
```

```
        flag = isOperation(file, letter);
        /*Проверка третьей части простого вырожения*/
        if (flag){
            if (file>>letter){
                if (letter == '(') {
                    for (int i = 0; i < depth+1; i++) cout<<"\t";
                    cout<<letter<<"\n";
                    flag = isSimple(file, letter, depth+1);
                }
                else {
                    for (int i = 0; i < depth; i++) cout<<"\t";
                    cout<<letter<<"\n";

                    flag = isSimple(file, letter, depth);
                }

            }
        }
        else {
            printError(5);
            return false;
        }
    }
}
else {
    printError(6);
    return false;
}
/*Проверка на наличие закрывающей скобки*/
if (flag) {
```

```
                if (file>>letter){
                    for (int i = 0; i < depth; i++) cout<<"\t";
                    cout<<letter<<"\n";
                    return (letter == ')');
                }
                else{
                    printError(8);
                    return false;
                }
            }
            else{
                printError(7);
                return false;
            }
        }
        else{
            printError(9);
            return false;
        }
    }
    else{
        printError(10);
        return false;
    }
}
```

/*Функция проверки первого сивмола, а также проверки на налчие лишних символов*/

```
bool isSimpleExpression(ifstream &file){
    char letter;
```

```
    bool flag = false;
    int depth = 0;

    if (file>>letter) {
        cout<<letter<<"\n";
        if (letter == '(' || isalpha(letter)) flag = isSimple(file, letter, depth);
        else printError(2);
        file>>letter;
        if (flag && !file.eof()) printError(4); // Вызов функции ошибок, если есть лишние символы или ошибки в самой строке
        flag = (flag && file.eof());
    }
    else printError(3);
    return flag;
}


int main(){
    string fileName = "input.txt";

    ifstream fin;
    fin.open(fileName);
    if (!fin.is_open()){
        printError(1);
        return 0;
    }

    if(isSimpleExpression(fin)) cout<<"\nIts a Simple Expression";
    else cout<<"\nIts not a Simple Expression";
```

```
    fin.close();
    return 0;
}
```