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

ОТЧЕТ

по лабораторной работе №1

по дисциплине «АиСД»

Тема: Программирование рекурсивных алгоритмов

Студент гр. 9303 _____ Емельянов С.А.

Преподаватель _____ Филатов А.Ю.

Санкт-Петербург

2020

Цель работы.

Создание рекурсивного алгоритма для анализа понятия «скобки», дальнейшее тестирование программы.

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:: = – | + | *

Основные теоретические положения.

Рекурсивным называется объект, содержащий сам себя или определенный с помощью самого себя. Мощность рекурсии связана с тем, что она позволяет определить бесконечное множество объектов с помощью конечного высказывания. Точно так же бесконечные вычисления можно описать с помощью конечной рекурсивной программы. Рекурсивные алгоритмы лучше всего использовать, когда решаемая задача, вычисляемая функция или обрабатываемая структура данных определены с помощью рекурсии. Если процедура (функция) P содержит явное обращение к самой себе, она называется прямо рекурсивной. Если P содержит обращение к процедуре (функции) Q, которая содержит (прямо или косвенно) обращение к P, то P называется косвенно рекурсивной. Многие известные функции могут быть определены рекурсивно. Например факториал, который присутствует практически во всех учебниках по программированию, а также наибольший общий делитель, числа Фибоначчи, степенная функция и др.

Выполнение работы.

Для решения поставленной задачи была разработана программа в среде Visual Studio Code на языке C++.

`int main()` – в данной функции считывается введенное с консоли пользователем имя файла, в случае, если файл с таким именем существует, то будет происходить последовательное считывание строк из файла, к каждой из строк будет применяться функция `SimpleExpression`, также будет выводиться информация о том, является ли строка «скобками», или же нет.

Bool `SimpleExpression(string &line, int& cur, int depth)` – функция п принимает на вход 3 аргумента: текущую строку, текущий индекс элемента строки, глубину вызова рекурсии. Возвращает `true` в случае, если строка – это «простое выражение», иначе `false`.

`void PrettyPrint(int& len)` – печатает отступ из символов «~», количество символов равно глубине рекурсии.

`void Error()` – печатает сообщение об ошибке.

Исходный код программы представлен в приложении А. Результаты тестирования приведены в приложении А.

Выводы.

Была реализована программа, включающая в себя рекурсивную функцию `SimpleExpression` для синтаксического анализа выражения «простое выражение». Также было проведено тестирование программы. Были удовлетворены следующие требования: информация на вход подается из файла, выводятся сообщения о начале и конце вызова функции `SimpleExpression` с отступами, соответствующими глубине рекурсии, результаты работы программы и все сообщения выводятся в консоль.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

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

void PrettyPrint(int &len){
    for (int i = 0; i < len; i++){
        cout << "~";
    }
}

void Error(){
        cout << "String contains extra - or incorrect characters!\n";
}

bool sign ( char c){
    return (c == '+' )|| (c == '*' )|| (c == '-');
}

bool SimpleExpression(string &line, int& cur, int depth){
    PrettyPrint(depth);
    cout << "Start" << endl;
     if (isalpha(line[cur])||line[cur] == '('|| line[cur] == ')'||(sign(line[cur])&&cur > 0)){
        if (cur>0){
                    if((sign(line[cur])&&line[cur-1]== ')')||(sign(line[cur])&&isalpha(line[cur-1]))){
                PrettyPrint(depth);
                cout << "End" << endl;
                return true;
```

```
            }
        }
        if(isalpha(line[cur])){
            if (cur>0){
                if(isalpha(line[cur-1])&&isalpha(line[cur]))
{
                    PrettyPrint(depth);
                    cout << "End" << endl;
                    return false;
                }
            }
            PrettyPrint(depth);
            cout << "End" << endl;
            return true;
        }
        else if(line[cur] == '('){
            cur++;
            if(SimpleExpression(line, cur, depth+1)){
                cur++;
                if(SimpleExpression(line, cur, depth+1)){
                    cur++;
                    if(SimpleExpression(line, cur, depth+1))
{
                        cur++;
                        if (line[cur] == ')'){
                            PrettyPrint(depth);
                            cout << "End" << endl;
                            return true;
                        }
                        else{
                            Error();
                            PrettyPrint(depth);
                            cout << "End" << endl;
                            return false;
```

```
                }
            }
            else{
                Error();
                PrettyPrint(depth);
                cout << "End" << endl;
                return false;
            }

        }
        else{
            Error();
            PrettyPrint(depth);
            cout << "End" << endl;
            return false;
        }


    }
    else{
        Error();
        PrettyPrint(depth);
        cout << "End" << endl;
        return false;
    }

}
else{
    Error();
    PrettyPrint(depth);
    cout << "End" << endl;
    return false;
}
```

```
        }
        else{
            Error();
            PrettyPrint(depth);
            cout << "End" << endl;
            return false;
        }

    }

    int main(){
        string file_name;
        cout << "Enter the name of an input file: " << endl;
        getline(cin, file_name);
        ifstream input(file_name);
        if (!input){
            cout << "You haven't entered correct input file." <<
endl;
        }
        else{
            cout << "Simple expression analyser: " << endl;
            string line;
            while(getline(input, line)){
                int len = line.length();
                int cur = 0;
                bool flag = SimpleExpression(line, cur, 1);
                if (flag && (cur == (len - 1))){
                    cout << "This is simple expression: ";
                    cout << line << endl;
                    cout<<"-----------------------------------
-----------------------------------------------"<<endl;
                }
                else if(line[cur] == ')'||cur == 0){
                    Error();
```

```
                    cout << "This is not simple expression: ";
                    cout << line << endl;
                    cout<<"простое_выражение::=простой_идентифик
атор|(простое_выражение знак_операции простое_выражение)\n";
                    cout<<"простой_идентификатор::= буква\n";
                    cout<<"знак_операции:: = – | + | *\n";
                    cout<<"-----------------------------------
-----------------------------------------------"<<endl;
                }
                else{
                    cout << "This is not simple expression: ";
                    cout << line << endl;
                    cout<<"простое_выражение::=простой_идентифик
атор|(простое_выражение знак_операции простое_выражение)\n";
                    cout<<"простой_идентификатор::= буква\n";
                    cout<<"знак_операции:: = – | + | *\n";
                    cout<<"-----------------------------------
-----------------------------------------------"<<endl;
                }
            }
        }
        return 0;
    }
```

ПРИЛОЖЕНИЕ Б

ТЕСТИРОВАНИЕ

Таблица Б.1 - Примеры тестовых случаев

№ п/п	Входные данные	Выходные данные	Комментарии
1.	A	~Start ~End This is simple expression: A	Программа работает исправно.
2.	)))))	~Start String contains extra - or incorrect characters! ~End String contains extra - or incorrect characters! This is not simple expression:))))) простое_выражение::=простой_идентификатор| (простое_выражение знак_операции простое_выражение) простой_идентификатор::= буква знак_операции:: = – | + | *	Программа работает исправно.
3.	+((((a+b)+(a+b))+a)+(a+b))	~Start String contains extra - or incorrect characters! ~End String contains extra - or incorrect characters! This is not simple expression: +((((a+b)+(a+b))+a)+(a+b)) простое_выражение::=простой_идентификатор| (простое_выражение знак_операции простое_выражение) простой_идентификатор::= буква знак_операции:: = – | + | *	Программа работает исправно.
4.	(ddfsd+dsgdsg+gdsgdsg)	~Start ~~Start ~~End ~~Start ~~End String contains extra - or incorrect characters! ~End This is not simple expression: (ddfsd+dsgdsg+gdsgdsg)	Программа работает исправно.

		простое_выражение::=простой_идентификатор| (простое_выражение знак_операции простое_выражение) простой_идентификатор::= буква знак_операции:: = – | + | *	
5.	(AAA)	~Start ~~Start ~~End ~~Start ~~End String contains extra - or incorrect characters! ~End This is not simple expression: (AAA) простое_выражение::=простой_идентификатор| (простое_выражение знак_операции простое_выражение) простой_идентификатор::= буква знак_операции:: = – | + | *	Программа работает исправно.
6.	(+++)	~Start ~~Start String contains extra - or incorrect characters! ~~End String contains extra - or incorrect characters! ~End This is not simple expression: (+++) простое_выражение::=простой_идентификатор| (простое_выражение знак_операции простое_выражение) простой_идентификатор::= буква знак_операции:: = – | + | *	Программа работает исправно.
7.	((((a+b)+(a+b))+a)+(a+b))	~Start ~~Start ~~~Start ~~~~Start ~~~~~Start ~~~~~End ~~~~~Start ~~~~~End	Программа работает исправно.

		~~Start ~~~~~End ~~~~End ~~~~Start ~~~~End ~~~~Start ~~~~~Start ~~~~~End ~~~~~Start ~~~~~End ~~~~~Start ~~~~~End ~~~~End ~~~End ~~~Start ~~~End ~~~Start ~~~End ~~End ~~Start ~~End ~~Start ~~~Start ~~~End ~~~Start ~~~End ~~~Start ~~~End ~~End ~End This is simple expression: ((((a+b)+(a+b))+a)+(a+b))	