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: Программирование рекурсивных алгоритмов

Студент гр. 9303 _____ Микулик Д.П.

Преподаватель _____ Филатов Ар.Ю.

Санкт-Петербург

2020

Цель работы.

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

Задание.

Вариант 14

Построить синтаксический анализатор для понятия «скобки».

Скобки : = A | (B скобки скобки)

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 работы программы-анализатора, были реализованы следующие функции: bool Bracket(string& line, int& cur, int& len, int depth), void Error(), void PrettyPrint(int& len).

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 Bracket, также будет выводиться информация о том, является ли строка «скобками», или же нет.

bool Bracket(string& line, int& cur, int& len, int depth) – главная функция программы, принимает на вход 4 аргумента: текущую строку, текущий индекс элемента строки, длину строки, глубину вызова рекурсии, так как данная функция вызывается рекурсивно, определяя, является ли строка выражением типа «скобки» или нет. Возвращает true в случае, если строка – это «скобки», иначе false. В функции последовательно проверяется: является ли элемент символом «A» или «(», в случае, если встретился символ «(» проверяется наличие следующего символа, при наличии его равенство символу «B», далее дважды вызывается рекурсия, для проверки понятий «скобки», после же проверяется, замыкается ли выражение символом «)». При несоблюдении хотя бы одного из этих условий, функция вернет false. Также, при несовпадении после окончания работы функции значений текущего индекса элемента и длины, уменьшенной на единицу, выражение не будет считаться скобками.

Void PrettyPrint(int& len) – печатает отступ из символов «!», количество символов равно глубине рекурсии.

void Error() – выводит сообщение об ошибке.

Исходный код программы представлен в приложении А. Результаты тестирования включены в приложение Б

Выводы.

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

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <string>

using namespace std;

bool Brackets(string &line, int& cur, int &len, int depth);
void PrettyPrint(int &len);
void Error();

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 << "Parenthesis analyser: " << endl;
        string line;
        while(getline(input, line)){
            int len = line.length();
            int cur = 0;
            bool b = Brackets(line, cur, len, 1);
            if (b && (cur == (len - 1))){
                cout << "This is parenthesis: ";
                cout << line << endl;
            }
            else{
                cout << "This is not parenthesis: ";
                cout << line << endl;
            }
        }
    }
    return 0;
}

bool Brackets(string &line, int& cur, int &len, int depth){
    //cout << depth << " ";
    PrettyPrint(depth);
    cout << "Start" << endl;
    bool res = false;
    if(cur < len){
        if (line[cur] == 'A'){
                PrettyPrint(depth);
                cout << "End" << endl;
                return true;
```

```
}
else if (line[cur] == '('){
    cur++;
    if((cur) < len){
        if (line[cur] == 'B'){
            cur++;
            res = Brackets(line, cur, len, depth+1);
            if (res){
                cur++;
                res = Brackets(line, cur, len, depth+1);
            }
            else{
                Error();
                PrettyPrint(depth);
                cout << "End" << endl;
                return false;
            }
            if (res){
                cur++;
                if (cur < len){
                    PrettyPrint(depth);
                    cout << "End" << endl;
                    return (line[cur] == ')');
                }
                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{
            cout << "End" << endl;
            PrettyPrint(depth);
            Error();
            return false;
        }

    }

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


    void Error(){
         cout << "String contains extra- or incorrect characters!" <<
endl;
           cout << "Correct format of string is: brackets := A |
B(brackets brackets)" << endl;
    }
```

ПРИЛОЖЕНИЕ Б
ТЕСТИРОВАНИЕ

Файл со входными данными: input.txt

Таблица Б.1 - Примеры тестовых случаев

№ п/п	Входные данные	Выходные данные	Комментарии
1.	A	!Start !End This is parenthesis: A	Программа работает нормально.
2.	(BAA)	!Start !!Start !!End !!Start !!End !End This is parenthesis: (BAA)	Программа работает нормально.
3.	(B(BAA)A)	!Start !!Start !!!Start !!!End !!!Start !!!End !!End !!Start !!End !End This is parenthesis: (B(BAA)A)	Программа работает нормально.
4.	(B(BAA)B())	!Start !!Start !!!Start !!!End !!!Start	Программа работает нормально. Ошибка здесь в выражении B()

		!!!End !!End !!Start String contains extra- or incorrect characters! Correct format of string is: brackets := A | B(brackets brackets) !!End String contains extra- or incorrect characters! Correct format of string is: brackets := A | B(brackets brackets) !End This is not parenthesis: (B(BAA)B())	
5.	(B(B(BA(BAA))A)(BA(BAA)))	!Start !!Start !!!Start !!!!Start !!!!End !!!!Start !!!!!Start !!!!!End !!!!!Start !!!!!End !!!!End !!!End !!!Start !!!End !!End !!Start	Программа работает нормально.

		!!!Start !!!End !!!Start !!!!Start !!!!End !!!!Start !!!!End !!!End !!End !End This is parenthesis: (B(B(BA(BAA))A) (BA(BAA)))	
6.	A(B(B(BA(BAA))A) (BA(BAA)))	!Start !End This is not parenthesis: A(B(B(BA(BAA))A) (BA(BAA)))	Программа работает нормально. Ошибка: программа не может начинаться с А в данном случае.
7.	(B(B(BA(BAA))A) (BA(BAA))	!Start !!Start !!!Start !!!!Start !!!!End !!!!Start !!!!!Start !!!!!End !!!!!Start !!!!!End !!!!End !!!End !!!Start !!!End !!End	Программа работает нормально. Ошибка: отсутствие одной из замыкающих скобок.

		!!Start !!!Start !!!End !!!Start !!!!Start !!!!End !!!!Start !!!!End !!!End !!End String contains extra- or incorrect characters! Correct format of string is: brackets := A | B(brackets brackets) !End This is not parenthesis: (B(B(BA(BAA))A) (BA(BAA))	
8.	(B(B(BA(BAA))A) (BA(BAA))))	!Start !!Start !!!Start !!!!Start !!!!End !!!!Start !!!!!Start !!!!!End !!!!!Start !!!!!End !!!!End !!!End !!!Start !!!End	Программа работает нормально. Ошибка: дополнительные замыкающие скобки.

		!!End !!Start !!!Start !!!End !!!Start !!!!Start !!!!End !!!!Start !!!!End !!!End !!End !End This is not parenthesis: (B(B(BA(BAA))A) (BA(BAA))))	
9.	(A(B(BA(BAA))A) (BA(BAA)))	!Start String contains extra- or incorrect characters! Correct format of string is: brackets := A | B(brackets brackets) !End This is not parenthesis: (A(B(BA(BAA))A) (BA(BAA)))	Программа работает нормально. Ошибка: недопустимый символ A после открывающей скобки.
10. (тест на некорректных данных)	asd(B(B(BA(BAA))A) (BA(BAA)))	!Start String contains extra- or incorrect characters! Correct format of string is: brackets := A | B(brackets brackets) !End This is not parenthesis:	Программа работает нормально.

		asd(B(B(BA(BAA))A)(BA(BAA)))	
11. (тест на некорректных данных)	asdasdasd	!Start String contains extra- or incorrect characters! Correct format of string is: brackets := A | B(brackets brackets) !End This is not parenthesis: asdasdasd	Программа работает нормально.
12. (тест, где имя входного файла некорректно)	dfdsfdsff	Enter the name of an input file: dfdsfdsff You haven't entered correct input file.	Программа работает нормально.