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. 8304 _____ Мухин А. М.

Преподаватель _____ Фирсов М. А.

Санкт-Петербург

2019

Цель работы.

Ознакомиться с основными понятиями и приёмами рекурсивного программирования, получить навыки программирования рекурсивных процедур и функций на языке программирования C++.

Задание.

Построить синтаксический анализатор для понятия скобки:
15й вариант.
скобки::=A | A (ряд_скобок)
ряд_скобок::= скобки | скобки ; ряд_скобок

Выполнение работы.

Тело программы состоит из 4х функций: main(), bracket(), rowBracket() и checkEL(). Работа начинается с функции int main() из которой вызывается функция bracket(), возвращаемое значения которой присваивается переменной result.

В версии со считыванием данных из файла вначале выделяется место для хранения очереднойй строки последовательности, которую по условию надо вывести на экран, затем идём возврат указателя на поток. Тут есть разница, если это последнее слово, значит у него нет символа переноса строки, тогда надо возвращать указатель, на один символ больше.

В функции bool bracket() мы определем на какую последовательность символов указывает поток. Если это символ ‘A’ и после него ничего нет, он нам подходит и мы возвращаем true. Если это ‘(’, то следующая за ней комбинация символов должна подходить под выражение, описанное в условие как ‘ряд_скобок’. Если это так, то проверяется наличие ‘)’ и в случае успеха возвращается истина. Если символа ‘)’ не обнаружено, мы выводим сообщение об ошибке и вернём значение ложь. Если после символа ‘A’ нет ни конца последовательности, ни ‘(’, мы возвращаем истину, так как вариант обычной A нам подходит.

Функция bool rowBrackets() возвращает истину, если последовательность, которую обработает, bracket() равна истине. Или ещё тогда, когда после этого, поток указывает на ‘;’ и следующая последовательность равна rowBrackets().

В конце концов, переменной result присвоено булево значение. Оно передаётся функции void checkEL() в качестве аргумента, для того, чтобы проверить закончилась ли строка и вывести соответствующую информацию.

Отдельно следует рассмотреть случай, когда в конце файла стоит символ переноса строки. Так как этот символ не идёт в счёт длины строки, однако это строка есть, мы не можем перенести указатель на поток, как мы это делали в других случах. Вместо этого вызывается функция checkEL() с заведомо ложным параметром и так как это последняя строка идёт выход из цикла while.

В конце функции main() мы отчищаем всё занятую нами память, а также закрываем файлы ввода и вывода.

char s - переменная, необходимая для хранения только что извлеченного из потока символа.

ifstream in - переменная, которая открывает файл для считывания последовательности на проверку.

ofstream out - переменная, которая открывает файл для записи служебной информации и результата.

char* k - переменная, необходимая для хранения строки последовательности.

bool first_error - переменная, необходимая для того, чтобы выводить первую определенную ошибку.

bool result - переменная, получающая ответ на вопрос, является ли наша последовательность скобками.

Разработанный программный код см. в приложении А.

Тестирование.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные
1.	A(A;A(A))	Данные файла: A(A;A(A)) True
2.	A(A(A);A)	Данные файла: A(A(A);A) True
3.	A(A;A;A;A(A;A;A(A;A;A(A;A;A))))	Данные файла: A(A;A;A;A(A;A;A(A;A;A(A;A;A)))) True
4.	A(A(A))	Данные файла: A(A(A)) True
5.	A(A(A)A;A)	Данные файла: A(A(A)A;A) Нет закрывающей скобки False
6.	A(A	Данные файла: A(A Нет закрывающей скобки False
7.	A(A;;A)	Данные файла: A(A;;A) Скобки должны начинаться с A False

Выводы.

Ознакомились с основными понятиями и приёмами рекурсивного программирования, получили навыки программирования рекурсивных процедур и функций на языке программирования C++ и создания синтаксического анализатора. Научились работать с потоками ввода и вывода, использовали такие функции, как tellg(), seekg() и методы peek(), putback(), get().

Разработаны 2 программы, выполняющие считывание с клавиатуры и из файла исходных данных.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: with_console.cpp

```
#include <iostream>


char s;
bool rowBrackets();
bool first_error = true;   // error retry check


bool bracket() {
    if (std::cin.peek() == 'A') {
        std::cin.get(s);
        if (std::cin.peek() == '\n')
            return true;
        else if (std::cin.peek() == '(') {
            std::cin.get(s);
            if (rowBrackets())
                if (std::cin.peek() == ')') {
                    std::cin.get(s);
                    return true;
                }
                else if (first_error) {
                    std::cout << "Нет закрывающей скобки" <<
std::endl;
                    first_error = false;
                }
        }
        else
            return true;
    }
    else if (first_error) {
        std::cout << "Скобки должны начинаться с A" << std::endl;
        first_error = false;
    }
    return false;
}


bool rowBrackets() {
    if (bracket()) {
        if (std::cin.peek() == ';') {
            std::cin.get(s);
            return rowBrackets();
        }
        return true;
    }
    else if (first_error) {
        std::cout << "Это не ряд_скобок" << std::endl;
        first_error = false;
    }
    return false;
}
```

```
int main() {
    std::cout << "Введите строку для проверки: ";
    bool result = bracket();
    std::cin.peek() == '\n' ? (result ? std::cout << "True" :
std::cout << "False") : (first_error ? std::cout << "Лишние символы
в конце" << std::endl << "False" : std::cout << "False");
    std::cout << '\n';
    return 0;
}
```

Название файла: with_file.cpp

```
#include <fstream>
#include <cstring>
#include <iostream>


char s;
static std::ifstream in("../Tests/tests.txt");
static std::ofstream out("../Tests/out.txt");
bool rowBrackets();
bool first_error = true;


bool bracket() {
    if (in.peek() == 'A') {
        in.get(s);
        if (in.peek() == EOF || in.peek() == '\n')
            return true;
        else if (in.peek() == '(') {
            in.get(s);
            if (rowBrackets())
                if (in.peek() == ')') {
                    in.get(s);
                    return true;
                }
                else if (first_error) {
                    out << "Нет закрывающей скобки" << std::endl;
                    first_error = false;
                }
        }
        else
            return true;
    }
    else if (first_error) {
        out << "Скобки должны начинаться с A" << std::endl;
        first_error = false;
    }
    return false;
}


bool rowBrackets() {
    if (bracket()) {
        if (in.peek() == ';') {
            in.get(s);
            return rowBrackets();
```

```
        }
        return true;
    }
    else if (first_error) {
        out << "Это не ряд_скобок" << std::endl;
        first_error = false;
    }
    return false;
}


bool check_EL(bool result) {
    if (in.peek() == EOF || in.peek() == '\n') {
        in.get(s);
        if (result)
            out << "True";
        else
            out << "False";
    }
    else {
        if (first_error)
            out << "Лишние символы в конце" << std::endl <<
"False";
        else
            out << "False";
    }
    out << "\n\n";
    first_error = true;
    while (s != '\n' && !in.eof())
        in.get(s);
}


int main() {
    char* k = (char*)calloc(150, sizeof(char));
    while (!in.eof()) {
        in.getline(k, 151);
        out << "Данные файла: " << k << std::endl;
        if (strlen(k) == 0 && in.peek() == EOF) {     // if last
line is \n
            in.get(s);
            check_EL(false);
            break;
        }
        if (in.eof())                                        // if last
line without \n
            in.seekg(-strlen(k), std::ios::cur);
        else                                              //no last line
            in.seekg(-strlen(k) - 1, std::ios::cur);
        bool result = bracket();
        check_EL(result);
    }
    in.close();
    out.close();
    free(k);
    return 0;
}
```