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

Студентка гр. 9382 Домнина К.Д.

Преподаватель Фирсов М.А.

Санкт-Петербург
2020

Цель работы.

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

Задание.

Вариант 8.

8. Построить синтаксический анализатор для понятия *простое_логическое*.

простое_логическое::= TRUE | FALSE |*простой_идентификатор* |
NOT *простое_логическое* |
(*простое_логическое знак_операции простое_логическое*)

простой-идентификатор::=буква

знак-операции::= AND | OR

Выполнение работы.

Задание было выполнено с помощью рекурсивной функции. На вход подаётся строка – логическое выражение. Если в выражении есть внутри скобки, то алгоритм запускается заново, пока не дойдёт до самых глубоких скобок. Если скобки не найдены, запускается функция для анализа строки. В случае успеха выражение в скобках заменяется на TRUE, и так до тех пор, пока в конце программа не получит выражение без скобок, которое также проверяется на корректность. При некорректном вводе данных программа завершается с соответствующим выводом.

Функции, с помощью которых реализовано задание: check_TRUE_FALSE_LETTER(string), analize(string), findPar(string, int*, int*), printThings(int), func(string*, int). Их описание приведено далее:

bool check_TRUE_FALSE_LETTER(string str)

Назначение функции: проверяет, является ли строка верным одиночным параметром.

Входные данные: string str – переменная, которая содержит в себе проверяемую строку.

Выходые данные: функция возвращает 1, если текущая строка является либо TRUE, либо FALSE, либо буквой.

Алгоритм:

Во входной строке в переменную int NOT_pos (означающая позицию NOT в строке) записывается либо позиция, если есть NOT, либо -1, если его нет. В зависимости от этого выполняется код в условной конструкции. Если в строке найден NOT – заменяет в текущей строке NOT + единичный параметр на TRUE и по итогу возвращается является ли получившаяся строка единичным параметром, в противном случае эта проверка пройдёт без прохождения кода в if.

Если в строке найден NOT – в переменную string checkstr записывается всё после NOT. Анализирует является ли checkstr единичным параметром. Функция возвращает 0, если после NOT нет пробела, так как это означает некорректный ввод, а также если checkstr не единичный параметр.

int analize(string str)

Назначение функции: функция анализа строки.

Входные данные: string str – строка для анализа.

Выходные данные: 1 – если строка верна, 0 – если ввод некорректный.

Алгоритм:

Сначала строка проверяется в функции check_TRUE_FALSE_LETTER(string) для того, чтобы исключить момент, когда строка является единичным параметром (единичный параметр не может быть записан в скобках).

Далее проверяется, есть ли в строке AND или OR. Так как AND является в логике более приоритетным параметром, то сначала проверяется он.

Для нахождения позиции AND используется переменная int AND_pos (если найдена – позиция в строке, если не найдено - -1). Сначала проверяется символы перед и после AND – если это не пробелы – программа возвращает 0 (некорректный ввод). Если проверка пройдена – в check_TRUE_FALSE_LETTER(string) передаются сначала левая часть (до AND),

если левая часть верна – правая часть. В случае, если обе части верны – программа возвращает 1.

Аналогично, если в выражении есть OR.

void findPar(string str, int* a, int* b)

Назначение функции: функция поиска скобок.

Входные данные: string str – строка, в которой выполняется поиск, int* a – переменная, в которой записывается положение первой скобки «(«, int* b – переменная, в которой записывается положение второй соответствующей скобки «)».

Выходные данные: нет, функция изменяет значения a и b.

Алгоритм:

Int x1 = -1 и int x2 = -1 – локальные переменные, в которые будут записываться положения скобок. Int i = 0 – переменная, обозначающая позицию в строке в конкретной итерации цикла.

С помощью цикла while идём пока не найдём закрывающую скобку или пока не дойдём до конца строки. Тем самым, в x1 запишется положение последней «(« до позиции «)». А в x2 – позицию «)», если цикл завершился и мы не дошли до конца строки.

Тем самым, между ними не будет никаких дополнительных скобок.

В *a записывается значение x1, в *b – значение x2.

void printThings(int a)

Назначение функции: вывод глубины рекурсии.

Входные данные: int a – переменная, в которой лежит глубина рекурсии.

Выходные данные: нет, функция печатает глубину рекурсии через «---|».

int func(string* str, int depth)

Назначение функции: рекурсивная функция.

Входные данные: string* str – ссылка на общую строку, int depth – глубина рекурсии.

Выходные данные: функция возвращает 1, если строка верна, иначе – 0.

Алгоритм:

Сначала проверяем строку в check_TRUE_FALSE_LETTER(string) на то, является ли она единичным параметром. Возвращаем 1, если верно, иначе продолжаем проход по функции.

Int a и int b – переменные, в которые запишутся положения «(« и «)» соответственно, запуская функцию findPar(string, int*, int*).

Если в строке нет какой-то одной скобки – неверно.

Выводим глубину с помощью функции printThings(int).

Запускаем анализ строки в скобках (передаём строку в функцию analize(string)) – если она неверна – возвращаем ноль (ввод неверный), иначе – заменяем переданную строку со скобками на TRUE.

Выводим глубину и происходящие изменения (для удобства пользования программой).

Входим в рекурсию, запуская эту же функцию с новыми параметрами. Все верно – фиксируем это выводом «Nice string»

Главная функция int main():

Для удобства пользователя – выбираем откуда производим ввод, обрабатывая это в условной конструкции.

Запускаем рекурсивную функцию и на выходе имеем, верна ли строка.

Разработанный программный код см. в приложении А.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные
1.	TRUE	OK
2.	TRUE AND FALSE	NOT OK
3.	((TRUE AND FALSE) OR FALSE)	Analizing TRUE AND FALSE Making ((TRUE AND FALSE) OR FALSE) to (TRUE OR FALSE) ---|Analizing TRUE OR FALSE ---|Making (TRUE OR FALSE) to TRUE

		---|Nice string Nice string OK
4	(((FALSE AND TRUE) OR A) AND NOT B)	Analizing FALSE AND TRUE Making (((FALSE AND TRUE) OR A) AND NOT B) to ((TRUE OR A) AND NOT B) ---|Analizing TRUE OR A ---|Making ((TRUE OR A) AND NOT B) to (TRUE AND NOT B) ---|---|Analizing TRUE AND NOT B ---|---|Making (TRUE AND NOT B) to TRUE ---|---|Nice string ---|Nice string Nice string OK

Выводы.

Были получены навыки работы с рекурсией, рекурсивными функциями.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

```
#include <iostream>
#include <fstream>
#include <string>
#include <ctype.h>

using namespace std;

bool check_TRUE_FALSE_LETTER(string str) {//Проверяет, является ли строка верным одиночным
параметром
    int NOT_pos = str.find("NOT");
    if (NOT_pos != -1) {
        string checkstr;
        if (str[NOT_pos + 3] != ' ')
            return 0;
        int i = NOT_pos + 4;
        while (str[i] != ' ' && i != str.length()) {
            checkstr += str[i];
            i++;
        }
        if (checkstr.compare("TRUE") == 0 || checkstr.compare("FALSE") == 0 || (check-
str.length() == 1 && isalpha(checkstr[0]))) {
            str = str.substr(0, NOT_pos) + "TRUE" + str.substr(i, str.length() - i);
        }
        else
            return 0;

    }
    return (str.compare("TRUE") == 0 || str.compare("FALSE") == 0 || (str.length() == 1 &&
isalpha(str[0])));
}

int analize(string str) {//Функция анализа строки
    cout << "Analizing " << str << endl;
    if (check_TRUE_FALSE_LETTER(str))
        return 0;

    int AND_pos = str.find("AND");
    if (AND_pos != -1) {
        if (AND_pos == 0 || AND_pos >= str.length() - 3 || str[AND_pos - 1] != ' ' ||
str[AND_pos + 3] != ' ')
            return 0;
        if (check_TRUE_FALSE_LETTER(str.substr(0, AND_pos - 1)) &&
            check_TRUE_FALSE_LETTER(str.substr(AND_pos + 4, str.length() - AND_pos - 4))) {
            return 1;
        }
    }

    int OR_pos = str.find("OR");
    if (OR_pos != -1) {
        if (OR_pos == 0 || OR_pos >= str.length() - 2 || str[OR_pos - 1] != ' ' ||
str[OR_pos + 2] != ' ')
            return 0;
        if (check_TRUE_FALSE_LETTER(str.substr(0, OR_pos - 1)) &&
            check_TRUE_FALSE_LETTER(str.substr(OR_pos + 3, str.length() - OR_pos - 3))) {
            return 1;
        }
    }
    return 0;
```

```
}

void findPar(string str, int* a, int* b) {//Функция поиска скобок
    int x1 = -1, x2 = -1;
    int i = 0;
    while (i < str.length() && str[i] != ')') {//Идем до первой ')'
        if (str[i] == '(') {//Попутно находя соответствующую ей '('
            x1 = i;
        }
        i++;
    }
    if (i != str.length())
        x2 = i;
    *a = x1;
    *b = x2;
}

void printThings(int a) {//Вывод глубины рекурсии
    for (int i = 0; i < a; i++) {
        cout << "---";
        cout << "|";
    }
}

int func(string* str, int depth) { //Рекурсивная функция
    if (check_TRUE_FALSE_LETTER(*str))//Проверяем на один параметр
        return 1;
    int a = 0, b = 0;
    findPar(*str, &a, &b);//Находим первые скобки, в которых нет других скобок
    if (a == -1 || b == -1) {//Если в строке нет какой-то одной скобки - неверно
        return 0;
    }
    printThings(depth);
    if (analize(str->substr(a + 1, b - a - 1))) {//Запускаем анализ строки в скобках
        string tmp = str->substr(0, a) + "TRUE" + str->substr(b + 1, (str->length() - a -
4));//Меняем эту строку если все ок
        printThings(depth);
        cout << "Making " << *str << " to " << tmp << endl;
        (*str) = tmp;
        if (func(str, depth + 1)) {//Запускаем снова эту функцию
            printThings(depth);
            cout << "Nice string\n";
            return 1;
        }
        else
            return 0;
    }
    return 0;
}

int main() {
    int choose = 0;
    cout << "Choose input:\n0.CIN\n1.FILE\n";
    cin >> choose;//Выбираем между вводом с консоли и из файла
    string x;
    if (choose == 0) {
        cin.ignore(1);
        getline(cin, x);
    }
    else {
        ifstream ff;
        ff.open("Lab1.txt");
        if (ff.is_open())
            getline(ff, x);
        else
```

```
        return 0;
    cout << "In file:\n" << x << "\n";
    ff.close();
}
if (func(&x, 0) == 1) {//Запускаем рекурсию
    cout << "OK\n";
}
else {
    cout << "NOT OK\n";
}
return 0;
   }
```