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. 8381 Киреев К.А.

Преподаватель Жангиров Т.Р.

Санкт-Петербург
2019

Цель работы.

Реализовать программу, использующую рекурсию, с целью ознакомления с основными понятиями и приёмами рекурсивного программирования и получения навыков программирования рекурсивных процедур и функций на языке программирования C++.

Задание.

Разработать программу, которая по заданному *простому_логическому* выражению, не содержащему вхождений простых идентификаторов, вычисляет значение этого выражения.

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

простой-идентификатор::=буква

знак-операции::= AND | OR

Основные теоретические положения.

В программировании рекурсия тесно связана с функциями, точнее именно благодаря функциям в программировании существует такое понятие как рекурсия или рекурсивная функция. Простыми словами, рекурсия – определение части функции (метода) через саму себя, то есть это функция, которая вызывает саму себя, непосредственно (в своём теле) или косвенно (через другую функцию). Типичными рекурсивными задачами являются задачи: нахождения n!, числа Фибоначчи. Вообще говоря, всё то, что решается итеративно можно решить рекурсивно, то есть с использованием рекурсивной функции. Всё решение сводится к решению основного или, как ещё его называют, базового случая. Существует такое понятие как шаг рекурсии или рекурсивный вызов. В случае, когда рекурсивная функция вызывается для решения сложной задачи (не базового случая) выполняется некоторое

количество рекурсивных вызовов или шагов, с целью сведения задачи к более простой. И так до тех пор пока не получим базовое решение.

Выполнение работы.

Сначала программа получает на вход строку, содержащую логическое выражение. Далее логические идентификаторы заменяются на числа или соответствующий знак с помощью функции replaceStr(), которая заменят все вхождения слова на соответствующий идентификатор. Например, TRUE заменяется на 1, OR заменяется на |. В зависимости от полученного далее в рекурсии выражения выводится TRUE или FALSE.

Далее вызывается рекурсивная функция calc(), которая принимает на вход исходную строку. В функции вызывается рекурсивная функция bracket(). Функция проверяет текущий символ на соответствие скобке, отрицанию или простому символу. При соответствии скобке вызывается функция calc(), где считается значение логического или. При соответствии отрицанию также вызывается функция calc(), но выводится значение обратное полученному. Далее проверяется следующий символ на соответствие логическому и, при котором снова вызывается рекурсивная функция bracket().

Полученные результаты либо выводятся на консоль.

Примеры работы программы.

№	Исходные данные	Результат
1	TRUE	TRUE
2	FALSE	FALSE
3	NOT TRUE	FALSE
4	NOT FALSE	TRUE
5	(TRUE AND NOT FALSE)	TRUE
6	(NOT FALSE OR FALSE)	TRUE
7	(((TRUE OR FALSE) AND (TRUE	FALSE

	AND NOT TRUE)) AND TRUE)	
8	(TRUE AND FALSE) OR (TRUE OR FALSE)	TRUE
9	(TRUE OR TRUE) OR TRUE	TRUE
10	(FALSE OR TRUE) AND FALSE	FALSE

Выводы.

В ходе выполнения лабораторной работы была реализована программа, позволяющая пользователю определить значение простого логического выражения. Были изучены основные понятия и приёмы рекурсивного программирования, получены навыки программирования рекурсивных функций на языке программирования C++.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

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

int bracket(string s);
int calc(string s);
string replaceStr(string str, string a, string b);
int rec=0, pos=0;

int main(){
    int result;
    string s;
    cout << "Enter boolean expression: " << endl;
    getline(cin, s);
    s = replaceStr(s, "NOT TRUE", "0");
    s = replaceStr(s, "NOT FALSE", "1");
    s = replaceStr(s, "NOT", "!");
    s = replaceStr(s, "TRUE", "1");
    s = replaceStr(s, "FALSE", "0");
    s = replaceStr(s, "OR", "|");
    s = replaceStr(s, "AND", "&");
    s = replaceStr(s, " ", "");
    s.insert(s.size(), " ");
    result = calc(s);
    if(result)
        cout << "Result: TRUE" << endl;
    else
        cout << "Result: FALSE" << endl;
    cout << "Depth of Recursion: " << rec;
    return 0;
}

string replaceStr(string str, string a, string b){
    int start = 0;
    while((start = str.find(a, start)) != string::npos){
        str.replace(start, a.length(), b);
        start += b.length();
    }
    return str;
}

int calc(string s){
    int x = bracket(s);
    char c = s.at(pos++);
    if(c == '|')
```

```
        return x || calc(s);
    else{
        pos--;
        return x;
    }
}

int bracket(string s){
    int x;
    char c = s.at(pos++);
    if(c == '('){
        rec++;
        x = calc(s);
        pos++;
    }
    else if(c == '!'){
        rec++;
        pos++;
        x = !calc(s);
        pos++;
    }
    else{
        x = c - '0';
    }
    c = s.at(pos++);
    if(c == '&')
        return x && bracket(s);
    else{
        pos--;
        return x;
    }
}
```