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

ОТЧЕТ

по лабораторной работе №1
по дисциплине «Алгоритмы и структуры данных»
Тема: Рекурсия.

Студент гр. 7383 _____ Кирсанов А.Я.

Преподаватель _____ Размочева Н.В.

Санкт-Петербург
2018

ОГЛАВЛЕНИЕ

1. ЦЕЛЬ РАБОТЫ

Цель работы: познакомиться с основными понятиями и приемами рекурсивного программирования, получить навыки программирования рекурсивных процедур и функций на языке программирования C++.

Формулировка задачи: 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

2. РЕАЛИЗАЦИЯ ЗАДАЧИ

Программа состоит из пяти функций:

1. main
2. Logical
3. Mark
4. Tilda
5. Error

В функции main реализована возможность использовать два типа ввода. Сначала пользователь выбирает тип ввода (1 – из файла и 2 – с клавиатуры). Чтобы завершить программу, нужно нажать 3. После выбора способа ввода предлагается либо ввести имя файла, либо ввести проверяемое выражение вручную. При правильном вводе всех данных программа выведет результат: выражение либо верно, либо неверно. Во втором случае выведется сообщение, указывающее на ошибку в выражении. После чего пользователю снова будет предложено выбрать способ ввода до тех пор, пока он не завершит программу, введя «3».

Функция Logical возвращает false, если выражение неверно. Сначала функция вызывает саму себя до тех пор, пока не закончатся открывающиеся скобки. Если в какой – то момент вместо скобки будет простое логическое, функция вызовет функцию Mark, которая проверят, следует ли за простым логическим знак операции. Если нет, функция Logical вернет false. Затем идет проверка на то, следует ли за знаком операции простое логическое (или открывающаяся скобка). Если нет, функция возвращает false. Свертывание рекурсии начинается при первой встрече закрывающей скобки. Суммы открывающих и закрывающих скобок должны быть равны, иначе функция вернет false. Если все предыдущие условия соблюдены, функция возвращает в main true, что свидетельствует о том, что выражение верно.

Функция Tilda вызывается каждый раз после считывания следующего символа и проверяет стоит ли отрицание перед простым логическим. Если да, то функция возвращает следующий за отрицанием символ. Два отрицания подряд не допускаются условием.

3. ТЕСТИРОВАНИЕ

Сборка и тестирование программы производилось в среде разработки QT на Linux Ubuntu 16.04 LTS.

В ходе тестирование были использованы различные выражения, заведомо правильные или неправильные. Результаты тестирования представлены в приложении А.

При тестировании программы были обнаружены и исправлены различные ошибки:

1) Функция Logical возвращала true при двойном отрицании «~~». В функцию Tilda добавлено дополнительное условие.
2) Функция Logical возвращала false при отрицании выражения, находящегося в скобках. Исправлено добавлением проверки на отрицание после каждого считывания символа.
3) Функция Logical возвращала false простом логическом, находящимся после знака операции не в скобках. Например: ((a|b)^a). В функцию Logical добавлено дополнительное условие для проверки является ли символ буквой.

4. ВЫВОД

В ходе работы были освоены методы рекурсивного программирования на языке C++. Стоит отметить, что рекурсивный подход проигрывает подходу, использующему списки в задачах с жестко поставленными условиями (например: (простое логическое + знак операции + простое логическое)). Однако, такая задача позволило более глубоко изучить принцип работы восходящей рекурсии и улучшить свои навыки в программировании.

ПРИЛОЖЕНИЕ А.

ТЕСТОВЫЕ СЛУЧАИ

Таблица 1 — Тестовые случаи

Входные данные	Вывод программы
Анализатор выражения: 1 - чтение из файла, 2 - ввод с клавиатуры, 3 - выход из программы. 2 Введите строку ((a^b)|(a||))	((a^b)|(a|| err#7 ! - Отсутствует выражение НЕТ, ЭТО НЕ ВЫРАЖЕНИЕ!
Анализатор выражения: 1 - чтение из файла, 2 - ввод с клавиатуры, 3 - выход из программы. 2 Введите строку ((0|1)|(b^d))	((0|1)|(b^d)) ЭТО ВЫРАЖЕНИЕ!
Анализатор выражения: 1 - чтение из файла, 2 - ввод с клавиатуры, 3 - выход из программы. 2 Введите строку ~(a^b)	~(a^b) ЭТО ВЫРАЖЕНИЕ!
Анализатор выражения: 1 - чтение из файла, 2 - ввод с клавиатуры, 3 - выход из программы. 2 Введите строку (((a^b)|(b|a))|(k^1))	(((a^b)|(b|a))|(k^1)) ЭТО ВЫРАЖЕНИЕ!
Анализатор выражения: 1 - чтение из файла, 2 - ввод с клавиатуры, 3 - выход из программы.	(~((~a^~b)|~(~b|~a))|~(~k|~1)) ЭТО ВЫРАЖЕНИЕ!

Продолжение таблицы

2 Введите строку (~((~a^~b)|~(~b|~a))|~(~k|~1))	
Анализатор выражения: 1 - чтение из файла, 2 - ввод с клавиатуры, 3 - выход из программы. 1 Введите имя файла input	(((a^b| err#4 ! - Отсутствует ')' НЕТ, ЭТО НЕ ВЫРАЖЕНИЕ!
Анализатор выражения: 1 - чтение из файла, 2 - ввод с клавиатуры, 3 - выход из программы. 1 Введите имя файла input	(~((~a^~b)|~(~b|~a))|~(~k|~1)) ЭТО ВЫРАЖЕНИЕ!
Анализатор выражения: 1 - чтение из файла, 2 - ввод с клавиатуры, 3 - выход из программы. 1 Введите имя файла input	(~((~a^~b)|~(~b|) err#7 ! - Отсутствует выражение НЕТ, ЭТО НЕ ВЫРАЖЕНИЕ!
Анализатор выражения: 1 - чтение из файла, 2 - ввод с клавиатуры, 3 - выход из программы. w	err#8 ! - Неверный выбор способа ввода
Анализатор выражения: 1 - чтение из файла, 2 - ввод с клавиатуры, 3 - выход из программы. 4	err#8 ! - Неверный выбор способа ввода

ПРИЛОЖЕНИЕ Б.

ИСХОДНЫЙ КОД ПРОГРАММЫ

```
//Вариант 8
// Program SyntaxAnalysisOfBracket;
// вариант с синхронным выводом входной строки (до места ошибки включительно)
/* Определения (синтаксис)
Expression = выражение, Logical = простое логическое, Mark = знак операции, Tilda - знак '~'
простое_логическое::= TRUE | FALSE | простой_идентификатор |
NOT простое_логическое
(простое_логическое знак_операции простое_логическое)
простой-идентификатор::=буква
знак-операции::= AND | OR
*/
#include <iostream>
#include <fstream>
#include <iomanip>
#include <cctype>
using namespace std ;
bool Logical (ifstream &infile, char s);
bool Mark (char s );
char Tilda(ifstream &infile, char s);
void Error (short k);

int main ()
{
    char str[1000];
    char str1[1];
    int k = 0;
    bool b;
    char s;
    setlocale (0,"Rus");

    while(k != 3){
        b = false;
        cout << endl <<"Анализатор выражения:" << endl << "1 - чтение из файла, 2 - ввод с клавиатуры, 3 - выход из программы." << endl;
        cin >> k;

        switch (k) {
        case 1:{
            cout << "Введите имя файла" << endl;
            cin >> str;
```

```
            ifstream infile(str);
            if (!infile) { cout << "Входной файл не открыт!" << endl;
break; }
            b = Logical(infile, s);
            break;
        }
        case 2:{
            cout << "Введите строку" << endl;
            cin.getline(str1, 1);
            cin.getline(str, 1000);
            ofstream fout("expression");
            fout << str;
            fout.close();
            ifstream infile("expression");
            if (!infile) { cout << "Входной файл не открыт!" << endl;
break; }
            if(infile>>s)
            {
                cout<<s;
                      s = Tilda(infile, s);
                b = Logical(infile, s);
            }
            else { Error(0); b = false; }
            if(infile>>s && b){ Error(1); b = false; }

            remove("expression");
            break;
        }
        case 3: break;
        default:{ Error(6); break; }
        }
        cout << endl;
        if(k == 1 || k == 2){
            if (b) cout << "ЭТО ВЫРАЖЕНИЕ!" << endl;
            else cout <<"НЕТ, ЭТО НЕ ВЫРАЖЕНИЕ!" << endl;
        }
    }


    return 0;
}


bool Logical (ifstream &infile, char s){
    if(s == '(')
```

```
    {
        if(infile>>s)
        {
                cout<<s;
                s = Tilda(infile, s);
                if(!Logical(infile, s))
                    return false;
        }
        else{ Error(7); return false; }

        if(infile>>s)
        {
            cout<<s;
            s = Tilda(infile, s);
            if(!Mark(s)){ Error(5); return false; }
        }
        else{ Error(5); return false; }
        if(infile>>s)
        {
            cout<<s;
            s = Tilda(infile, s);
            if(!Logical(infile, s))
                    return false;
        }
        else{ Error(5); return false; }
        if(infile>>s)
        {
            cout<<s;
            s = Tilda(infile, s);
            if (s == ')')
                return true;
            else{ Error(4); return false; }
        }
        else{ Error(4); return false; }
        }
    else if(isalpha(s))
        return true;
    else{ Error(2); return false; }
}


bool Mark (char s){
    if((s == '^') || (s == '|')) return true;
    else{ Error(5); return false; }
}
```

```
char Tilda(ifstream &infile, char s){
    if( s == '~' ){
        if((infile >> s) && (s != '^') && (s != '|') && (s != '~'))
{ cout << s; return s; }
        else Error(3);
    }
    return s;
}

void Error (short k)
{
    cout << endl << "err#" << k << endl;
    switch (k) {
    case 0: cout << "! - Пустая входная строка" << endl; break;
    case 1: cout << "! - Лишние символы во входной строке" << endl;
break;
    case 2: cout << "! - Недопустимый начальный символ" << endl; break;
    case 3: cout << "! - Отсутствует выражение после '~'." << endl;
break;
    case 4: cout << "! - Отсутствует ')'" << endl; break;
    case 5: cout << "! - Отсутствует знак операции" << endl; break;
    case 6: cout << "! - Неверный выбор способа ввода." << endl; break;
    case 7: cout << "! - Отсутствует выражение" << endl; break;
    default : cout << "! - ...";break;
    };
}
```