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. Not
5. Space
6. Error

В функции main производится ввод выражения. Выражение можно считать из файла, или ввести вручную с клавиатуры. В обоих случаях строке str0 присваивается вводимое выражение, а затем str0 передается в поток strstream. Данный поток позволяет считывать слова, разделенные пробелом независимо от количества пробелов между словами. Перед передачей в поток main вызывает функцию Space, которая ставит пропущенные пробелы до и после скобок. Это обусловлено особенностями работы потока strstream. Функция main производит проверку на то, пустая ли строка и допустимый ли начальный символ. Если проверка прошла успешна, main вызывает функцию Logical.

Функция Logical проверяет правильность написания выражения. Она рекурсивная и вызывает саму себя каждый раз, когда встретит открывающую скобку. После раскрытия первых скобок идет проверка стоит ли простое логическое или его отрицание после скобки, далее проверка на знак – функция Mark. Отрицание знака недопустимо, такие ошибки обрабатываются в функции Not. Следом идет проверка на простое логическое или его отрицание после знака. Затем проверка на закрывающую скобку. Если не выполняется хотя бы одно условие на предыдущих этапах, функция выведет соответствующую ошибку. Для облегчения вывода ошибок существует

функция Error.

При отсутствии ошибок функция Logical передаст в функцию main true. Если после отработки функции Logical будут обнаружены дополнительные символы в потоке strstream, функция main выведет ошибку о лишних символах. Если лишние символы отсутствуют, функция main выведет: «Это выражение!». Иначе: «Это не выражение!».

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 «NOT NOT». В функцию Not добавлено дополнительное условие.
2) Функция Logical возвращала false при отрицании выражения, находящегося в скобках. Исправлено добавлением проверки на отрицание после каждого считывания следующего слова.
3) Функция main всегда выводила строку «Лишние символы во входной строке», даже при правильно введенном выражении. В функцию Space добавлена строка, ставящая в конец производной строки символ конца строки ‘\0’.

4. ВЫВОД

В ходе работы были освоены методы рекурсивного программирования на языке C++. Изучен поток strstream, позволяющий считывать слова, разделенные пробелом из строки. Приобретены навыки работы с отладчиком QT. Получены навыки работы с типом string, отсутствовавшем в Си. В программе анализатора скобок были обнаружены и исправлены различные ошибки. При последующем тестировании ошибок обнаружено не было.

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 ((TRUE OR FALSE) OR A)	((TRUE OR FALSE) OR A) ЭТО ВЫРАЖЕНИЕ!
Анализатор выражения: 1 - чтение из файла, 2 - ввод с клавиатуры, 3 - выход из программы. 2 Введите строку ((TRUE OR FALSE) OR A	((TRUE OR FALSE) OR A err#4 ! - Отсутствует ')' НЕТ, ЭТО НЕ ВЫРАЖЕНИЕ!
Анализатор выражения: 1 - чтение из файла, 2 - ввод с клавиатуры, 3 - выход из программы. 2 Введите строку NOT(A OR B)	NOT (A OR B) ЭТО ВЫРАЖЕНИЕ!
Анализатор выражения: 1 - чтение из файла, 2 - ввод с клавиатуры, 3 - выход из программы. 2 Введите строку ((A OR B)AND(TRUE AND FALSE))	((A OR B) AND (TRUE AND FALSE)) ЭТО ВЫРАЖЕНИЕ!
Анализатор выражения: 1 - чтение из файла, 2 - ввод с клавиатуры, 3 - выход из программы. 1 Введите имя файла input	(A OR err#7 ! - Отсутствует простое логическое НЕТ, ЭТО НЕ ВЫРАЖЕНИЕ!
Анализатор выражения: 1 - чтение из файла, 2 - ввод с клавиатуры, 3 - выход из программы. 2 Введите строку NOT	NOT err#3 ! - Отсутствует выражение после NOT.

ПРИЛОЖЕНИЕ Б.

ИСХОДНЫЙ КОД ПРОГРАММЫ

```
//Вариант 8
// Program SyntaxAnalysisOfBracket;
// вариант с синхронным выводом входной строки (до места ошибки включительно)
/* Определения (синтаксис)
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 <strstream>
#include <fstream>
using namespace std ;

bool Logical (strstream &x, string word);
bool Mark (string word );
string Not(strstream &x, string word);
string Space(char str0[]);
void Error (short k);

int main ()
{
    string word;
    string str;
    char str0[1000];
    short k;
    bool b;
    char c = '\0';
    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:{
            strstream x;
            cout << "Введите имя файла" << endl;
```

```
            cin >> str;
            ifstream outfile(str);
            if (!outfile) { cout << "Входной файл не открыт!" << endl;
break; }
            outfile.read(str0, 1000);
            outfile.close();
            str = Space(str0);
            x << str;
            if(x >> word){
                cout << word << " ";
                word = Not(x, word);
                if(word[0] && word[0] != '('){ Error(2); return 0; }
                b = Logical(x, word);
            }
            else { Error(0); b = false; }
            x >> c;
            if(c != '\0' && b){ Error(1); b = false; }
            break;
        }
        case 2:{
            cout << "Введите строку" << endl;
            cin.get();
            cin.getline(str0, 1000);
            strstream x;
            str = Space(str0);

            x << str;
            if(x >> word)
            {
                cout << word << " ";
                word = Not(x, word);
                if(word[0] != '\0' && word[0] != '('){ Error(2); break; }
                if(word[0] == '\0'){ Error(0); break; }
                b = Logical(x, word);
            }
            else { Error(0); b = false; }
            x >> c;
            if(c != '\0' && b){ Error(1); b = false; }

            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 (strstream &x, string word){
    if(word[0] == '(')
    {
        if(x >> word){
            cout << word << " ";
            word = Not(x, word);
            if(word[0] == '('){
                if(!Logical(x, word)) return false;
            }
            else{
                if(!Logical(x, word)) return false;
            }
        }
        else{ Error(7); return false; }

        if(x >> word){
            cout << word << " ";
            word = Not(x, word);
            if(!Mark(word)) return false;
        }
        else{ Error(5); return false; }

        if(x >> word){
            cout << word << " ";
            word = Not(x, word);
            if(!Logical(x, word)) return false;
        }
        else{ Error(7); return false; }

        if(x >> word){
            cout << word << " ";
            if(word[0] == ')') return true;
        }
        else{ Error(4); return false; }

        }
```

```
    if((isalpha(word[0]) && word.length() == 1)) return true;
    if((word == "FALSE") || (word == "TRUE")) return true;
    else{ Error(7); return false; }
}


bool Mark (string word){
    if(word[0] == '\0') return false;
    if((word == "AND") || (word == "OR")) return true;
    else{ Error(5); return false; }
}

string Not(strstream &x, string word){
    if( word == "NOT" ){
        if(x >> word){
            if((word == "AND") || (word == "OR")){ Error(8); word[0] =
'\0'; return word; }
            cout << word << " "; return word;

        }
        else{  Error(3); word[0] = '\0'; return word; }
    }
    return word;
}

string Space(char str0[]){
    int i = 0, k = 0;
    string str1;
    char str[1000];
    while(str0[i] != '\0'){
        if(str0[i] == '(' || str0[i] == ')'){
            str[k] = ' ';
            str[k+1] = str0[i];
            str[k+2] = ' ';
            k += 3;
            i++;
        }
        else{
            if(str0[i] == '\n') break;
            str[k] = str0[i];
            i++;
            k++;
        }
    }
    str[k] = '\0';
```

```
    str1 = str;
    return str1;
}


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 NOT." << 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 простое логическое" << endl; break;
    case 8: cout << "! - Недопустимо отрицание операции" << endl; break;
    default : cout << "! - ...";break;
    };
}
```