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(ка) гр. 9381 _____ Шахин Н.С

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++. Разработать программу, использующую рекурсию.

Задание.

Вариант 22

Построить синтаксический анализатор для определённого далее понятия *логическое_выражение*.

логическое_выражение ::= TRUE | FALSE |*идентификатор* |

NOT (*операнд*) |*операция* (*операнды*)

идентификатор::=буква

операция::= AND | OR

операнды::= операнд |*операнд, операнды*

операнд::= логическое_выражение

Основные теоретические положения.

Синтаксический анализатор (парсер) - программа, которая определяет, является ли заданная (входная) последовательность символов логическим выражением или нет. Синтаксический анализатор удобно реализовать с помощью рекурсии. Функция, проверяющая выражение, может запускать внутри себя функции, проверяющая отдельные части выражения на соответствие заданным паттернам.

Выполнение работы.

Написание работы проводилось на базе операционной системы Linux с использованием среды разработки CLion. Для реализации программы был разработан CLI. Чтобы подать информацию из файла, необходимо запустить программу с флагом -f <имя файла>. Для ввода информации через консоль необходимо запустить программу без флагов.

После получения входной строки вызывается функция *bool checkString(string& str)*, в которую передаётся строка по ссылке. Эта функция вызывает рекурсивную функцию проверки строки *bool statement (string& str, int& pos, int indent)*, в которую передаётся копия строки, текущее положение в строке, а также глубина рекурсии. После завершения функции *statement(),* функция *checkString()* проверяет наличие лишних символов в строке и выводит сообщение об ошибке, если возвращаемое значение функции *statement()* – false. В зависимости от значения, которое вернула функция *checkString()* функция main выводит результат работы алгоритма.

Описание работы функций и алгоритма.

void skip(string& str, int& pos, int indent, int n) – функция для удаления проверенных символов строки. В функцию передаются строка по ссылке, текущее положение в строке, глубина рекурсии, и количество символов для удаления. Перед удалением вызывается функция для вывода удаляемых символов.

bool findWord (string& str, int& pos, int indent, const char* word) – функция для поиска терминальных символов в строке. В функцию передаются строка по ссылке, текущее положение в строке, глубина рекурсии, и слово для поиска. Функция возвращает TRUE или FALSE в зависимости от результата поиска. Если слово найдено, то вызывается функция skip().

bool checkLetter(string& str, int& pos, int indent) – функция для поиска буквы в строке. В функцию передаются строка по ссылке, текущее положение в строке, и глубина рекурсии. Если длина строки равна 1 и символ является буквой или после буквы стоит не буква и не цифра, то вызывается функция skip().

bool operation (string& str, int& pos, int indent) – функция для поиска «AND | OR». В функцию передаются строка по ссылке, текущее положение в строке, и глубина рекурсии. В функции вызывается find Word().

bool operands(std::string& str, int& pos, int indent) – функция для поиска *операнд* | *операнд, операнды*. В функцию передаются строка по ссылке,

текущее положение в строке, и глубина рекурсии. Для поиска «операнд» используется рекурсивная функция statement(). Для поиска запятой используется функция findWord(). Для поиска «операнды» функция operands() вызывает сама себя.

bool statement (string& str, int& pos, int indent) – основная функция алгоритма которая проверяет всё выражение. В функцию передаются строка по ссылке, текущее положение в строке, и глубина рекурсии. В функции вызываются findWord(), checkLetter(), operation(), operands() для проверки строки.

bool checkString(string& str) – в функцию передаётся исходная строка, затем делается копия строки, копия передаётся в функцию statement(). После завершения функции statement() идёт проверка на пустую строку, если в строке остались символы, то строка не подходит и выводится сообщение об ошибке.

Тестирование.

Таблица 1. Результаты тестирования

№	Входные данные	Выходные данные
1	read from file - test2.txt Вы ввели: NOT(A)	START FUNCTION STATEMENT NOT (START FUNCTION STATEMENT A END FUNCTION STATEMENT) END FUNCTION STATEMENT SUCCESS
2	read from file - test3.txt Вы ввели: AND(OR(AND(OR(NOT(A)))))	START FUNCTION STATEMENT AND (

		START FUNCTION STATEMENT OR (START FUNCTION STATEMENT AND (START FUNCTION STATEMENT OR (START FUNCTION STATEMENT NOT (START FUNCTION STATEMENT A END FUNCTION STATEMENT) END FUNCTION STATEMENT) END FUNCTION STATEMENT) END FUNCTION STATEMENT)

		END FUNCTION STATEMENT) END FUNCTION STATEMENT SUCCESS
3	read from file - test4.txt Вы ввели: AND(B,C,AND(A,B))	START FUNCTION STATEMENT AND (START FUNCTION STATEMENT B END FUNCTION STATEMENT , START FUNCTION STATEMENT C END FUNCTION STATEMENT , START FUNCTION STATEMENT AND (START FUNCTION STATEMENT A END FUNCTION STATEMENT , START FUNCTION STATEMENT B END FUNCTION STATEMENT) END FUNCTION STATEMENT

		) END FUNCTION STATEMENT SUCCESS
4	read from file - test5.txt Вы ввели: AND(TRUE,NOT(FALSE))	Вы ввели: AND(TRUE,NOT(FALSE)) START FUNCTION STATEMENT AND (START FUNCTION STATEMENT TRUE END FUNCTION STATEMENT , START FUNCTION STATEMENT NOT (START FUNCTION STATEMENT FALSE END FUNCTION STATEMENT) END FUNCTION STATEMENT) END FUNCTION STATEMENT SUCCESS

5	read from file - test6.txt Вы ввели: AND(TRUE,NOT(FALSE), AND(A,B,NOT(TRUE)))	START FUNCTION STATEMENT AND (START FUNCTION STATEMENT TRUE END FUNCTION STATEMENT

		, START FUNCTION STATEMENT NOT (START FUNCTION STATEMENT FALSE END FUNCTION STATEMENT) END FUNCTION STATEMENT , START FUNCTION STATEMENT AND (START FUNCTION STATEMENT A END FUNCTION STATEMENT , START FUNCTION STATEMENT B END FUNCTION STATEMENT , START FUNCTION STATEMENT NOT (

		START FUNCTION STATEMENT TRUE END FUNCTION STATEMENT) END FUNCTION STATEMENT) END FUNCTION STATEMENT) END FUNCTION STATEMENT SUCCESS
6	read from file - test7.txt Вы ввели: AND(TRUE,NOT((FALSE))	START FUNCTION STATEMENT AND (START FUNCTION STATEMENT TRUE END FUNCTION STATEMENT , START FUNCTION STATEMENT NOT (START FUNCTION STATEMENT >Statement expected END FUNCTION STATEMENT END FUNCTION STATEMENT END FUNCTION STATEMENT > AND(TRUE,NOT((FALSE))

		> ^ NOT_CORRECT
7	read from file - test8.txt Вы ввели: AND(NOT(A,B))	START FUNCTION STATEMENT AND (START FUNCTION STATEMENT NOT (START FUNCTION STATEMENT A END FUNCTION STATEMENT >')' expected END FUNCTION STATEMENT END FUNCTION STATEMENT > AND(NOT(A,B)) > ^ NOT_CORRECT
8	read from file - test9.txt Вы ввели: NOT(AND(A,NOT(ERR)))	START FUNCTION STATEMENT NOT (START FUNCTION STATEMENT AND (START FUNCTION STATEMENT A END FUNCTION STATEMENT ,

		START FUNCTION STATEMENT NOT (START FUNCTION STATEMENT >Statement expected END FUNCTION STATEMENT END FUNCTION STATEMENT END FUNCTION STATEMENT END FUNCTION STATEMENT > NOT(AND(A,NOT(ERR))) > ^ NOT_CORRECT
9	Вы ввели: TRUE,FALSE	START FUNCTION STATEMENT TRUE END FUNCTION STATEMENT >End of string expected. ",FALSE" > TRUE,FALSE > ^
10	Вы ввели: AAA	START FUNCTION STATEMENT >Statement expected END FUNCTION STATEMENT > AAA > ^ NOT_CORRECT

Вывод.

В ходе выполнения лабораторной работы был изучен такой вид алгоритмов, как синтаксические анализаторы. Была реализована программа,

которая анализирует строку рекурсивным методом, определяя соответствие определению.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл: main.cpp

```
#include "structs.h"
#include "algo.h"
#include "inputoutput.h"

int main(int argc, char** argv) {
    string expression = cuinput(argc, argv);
    if(expression == "HELP"){
        return 0;
    }
    cout<<"Вы ввели: "<< expression<<endl;
    bool result = checkString(expression);
    if(result){
        cout<<"\033[1;32mSUCCESS\033[0m\n";
    } else{
        cout<<"\033[1;31mNOT_CORRECT\033[0m\n";
    }
    if(argc > 1){
        if(result) {
            ofstream outfile(optarg, ios::app);
            outfile << "SUCCESS\n";
        } else{
            ofstream outfile(optarg, ios::app);
            outfile << "NOT_CORRECT\n";
        }
    }
    return 0;
}
```

Файл: structs.h

```
#ifndef STRUCTS_H
#define STRUCTS_H

#include <iostream>
#include <string>
```

```
#include <cstring>
#include <unistd.h>
#include <fstream>

using namespace std;

#endif //STRUCTS_H
```

Файл: algo.h

```
#ifndef ALGO_H
#define ALGO_H

#include "structs.h"
#include "inputoutput.h"

#define TRUE_S "TRUE"
#define FALSE_S "FALSE"
#define NOT_S "NOT"
#define AND_S "AND"
#define OR_S "OR"
#define OPEN_BRACKET "("
#define CLOSE_BRACKET ")"
#define COMMA ","

bool statement (string& str, int& pos, int indent);

bool operands(string& str, int& pos, int indent);

bool operation (string& str, int& pos, int indent);

bool findWord (string& str, int& pos, int indent, const char* word);

bool checkLetter(string& str, int& pos, int indent);

bool checkString(string& str);

void skip (string& str, int& pos, int indent, int n = 1);
#endif
```

Файл: algo.cpp

```
#include "algo.h"

void skip(string& str, int& pos, int indent, int n){
    if (str.length() >= n) {
        // Вывод удаляемых символов
        proceedOutput(str.substr(0, n), indent);
        str = str.substr(n);
        pos++;
    }
}

bool findWord(string& str, int& pos, int indent, const char* word){
    int len = strlen(word);
    // Сравнение начала строки с word
    if (!str.compare(0, len, word) && (len == str.length() || (len <
str.length() && (!isalnum(str[len]) || !isalnum(word[0])))){
        skip(str, pos, indent, len);
        return true;
```

```
    }
    return false;
}

bool checkLetter(string& str, int& pos, int indent){
    // Проверка, является ли первый символ строки буквой
    if ((str.length() == 1 && isalpha(str[0])) || (str.length() > 1 &&
isalpha(str[0]) && !isalnum(str[1]))) {
        skip(str, pos, indent);
        return true;
    }
    return false;
}

// AND | OR
bool operation (string& str, int& pos, int indent){
    if (findWord(str, pos, indent, AND_S)) {
        return true;
    }
    else if (findWord(str, pos, indent, OR_S )) {
        return true;
    }
    return false;
}

// OPERAND | OPERAND,OPERANDS
bool operands(std::string& str, int& pos, int indent) {
    if (statement(str, pos, indent + 1)) {
        if (findWord(str, pos, indent, COMMA )) {
            return operands(str, pos, indent);
        }
        return true;
    }
    return false;
}

bool statement (string& str, int& pos, int indent){
    info(indent, START);
    //поиск TRUE
    if (findWord(str, pos, indent, TRUE_S )) {
        info(indent, END);
        return true;
    }
    // поиск FALSE
    else if (findWord(str, pos, indent, FALSE_S )) {
        info(indent, END);
        return true;
    }
    //происк буквы
    else if (checkLetter(str, pos, indent)) {
        info(indent, END);
        return true;
    }
    else if (findWord(str, pos, indent, NOT_S )) {
        if (findWord(str, pos, indent, OPEN_BRACKET )) {
            if (statement(str, pos, indent + 1) && findWord(str, pos,
indent, CLOSE_BRACKET )) {
                info(indent, END);
```

```
                return true;
            }
            else proceedErr("')' expected", pos);
        }
        else proceedErr("'(' expected", pos);
    }
    else if (operation(str, pos, indent + 1)) {
        if (findWord(str, pos, indent, OPEN_BRACKET )) {
            if (operands(str, pos, indent + 1) && findWord(str, pos,
indent, CLOSE_BRACKET )) {
                info(indent, END);
                return true;
            }
            else proceedErr("')' expected", pos);
        }
        else proceedErr("'(' expected", pos);
    }
    proceedErr("Statement expected", pos);
    info(indent, END);
    return false;
}

bool checkString(string& str){
    std::string copy = string(str);
    int position = 0;
    if (statement(copy, position, 1)) {
        if (copy.empty()) {
            return true;
        }
        else proceedErr("End of string expected. \"" + copy, position);
    }
    // Вывод ошибки
    cout << "> " << str << endl << "> ";
    for (int i = 0; i < str.length() - copy.length(); ++i) {
        cout << " ";
    }
    cout << "^" << endl;
    return false;
}
```

Файл: inputoutput.h

```
#ifndef INPUTOUTPUT_H
#define INPUTOUTPUT_H
#include "structs.h"
#define START 1
#define END 0

string cuinput(int argc, char** argv);

void proceedOutput(string output, int indent);

void proceedErr(const string& err, int& pos);

void info(int indent, int flag);
```

Файл: inputoutput.cpp

```
#include "inputoutput.h"
```

```
string cuinput(int argc, char** argv){
    if(argc == 1) {
        cout<<"Введите выражение: ";
        string res;
        cin >> res;
        return res;
    }
    int option = 0;
    while ((option = getopt(argc,argv,"hf:"))!=-1){
        switch (option) {
            case    'h':    cout<<"Синтаксический    анализатор    для
выражений.\nДля того чтобы прочитать выражение из файла используйте флаг
-f <название файла>.\n"; return "HELP";
            case 'f': cout<<"read from file - "<<optarg<<endl;
                ifstream infile(optarg);
                if (!infile) {
                    cout << "> File can't be open!" << endl;
                    return "";
                }
                string str;
                infile >> str;
                return str;
            break;
        }
    }
}

void proceedOutput(string output, int indent){
    for(int i = 0; i < indent; i++){
        cout<<"  ";
    }
    cout<<output<<endl;
}

void proceedErr(const string& err, int& pos){
    if(pos!= -1){
        pos = -1;
        cout<<">"<<err<<endl;
    }
}

void info(int indent, int flag){
    for(int i = 0; i < indent; i++){
        cout<<"  ";
    }
    if(flag){
        cout <<"\033[1;34mSTART FUNCTION STATEMENT\033[0m\n";
    } else{
        cout <<"\033[1;34mEND FUNCTION STATEMENT\033[0m\n";
    }
}
```