

**МИНОБРНАУКИ РОССИИ**  
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**  
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**  
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**  
**Кафедра МОЭВМ**

**ОТЧЕТ**  
**по лабораторной работе №1**  
**по дисциплине «Алгоритмы и структуры данных»**  
**ТЕМА: РЕКУРСИИ**  
**Вариант № 21**

Студент гр. 8304  
Преподаватель

Рыжиков А. В.  
Фирсов К. В.

Санкт-Петербург  
2019

## 1 Цель работы.

Построить синтаксический анализатор для определённого далее понятия *логическое выражение*.

$$\text{логическое\_выражение} ::= \text{TRUE} \mid \text{FALSE} \mid \text{идентификатор} \mid \\ \text{NOT (операнд)} \mid \text{операция (операнды)}$$
$$\text{идентификатор} ::= \text{буква}$$
$$\text{операция} ::= \text{AND} \mid \text{OR}$$
$$\text{операнды} ::= \text{операнд} \mid \text{операнд, операнды}$$
$$\text{операнд} ::= \text{логическое\_выражение}$$

*! Буквой назовём маленькую и большую букву английского алфавита.*

## 2 Входные данные

Входными данные являются строки, находящиеся в файле test.txt.

## 3 Описание программы

Программа решает поставленную задачу при помощи рекурсии.

### 3.1 Зависимости и объявление функций

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

bool isItLogicalExpression(string basicString, int *count);
bool isTrue(string basicString, int *count);
bool isFalse(string basicString, int *count);
bool isLetter(string basicString, int *count);
bool isOperandWithNot(string basicString, int *count);
bool isOperand(string basicString, int *count);
bool isOperationWithOperands(string basicString, int *count);
bool isOperands(string basicString, int *count);
bool isOperands(string basicString, int *count);
bool check(string basicString, int *count);
void work(string name);
```

### 3.2 Функция main.

Программа решает поставленную задачу при помощи рекурсии, чтение происходит из файла test.txt .

```
int main() {
    ifstream fin;
    fin.open("test.txt");

    if (fin.is_open()) {
        cout << "Reading from file:" << "\n";

        int super_count = 0;

        while (!fin.eof()) {

            super_count++;

            string str;
            getline(fin, str);

            cout << "test #" << super_count << " \"" + str + "\"" << "\n";
```

```

        work(str);
    }
} else {
    cout << "File not opened";
}

fin.close();

return 0;
}

```

## Функция work.

Функция проверяет подданные данные на принадлежность к логическому выражению и выводит отладочную информация. Явным признаком успешной проверки (в случае если выражение является логическим выражением) служит то, что количество пройденных символов равно количеству СИМВОЛОВ В САМОМ ВЫРАЖЕНИИ.

```

void work(string name) {
    int count = 0;
    if (isItLogicalExpression(name, &count)) {
        if (count == name.size()) {

            cout << "OK" << " ";
            cout << count << " ";
            cout << name.size() << "\n";

        } else {
            cout << "Not OK" << " ";
            cout << count << " ";
            cout << name.size() << "\n";
        }
    } else {
        cout << "Not OK (from logical)" << "\n";
    }
}

```

### 3.3 Функция *isItLogicalExpression(string basicString, int \*count)*

Проверяет является ли поданное выражение логическим выражением.

```
bool isItLogicalExpression(string name, int *count) {  
    int num = *count;  
    if (isTrue(name, count)) {  
        return true;  
    } else {  
        if (isFalse(name, count)) {  
            return true;  
        } else {  
            if (isOperandWithNot(name, count)) {  
                return true;  
            } else {  
                if (isOperationWithOperands(name, count)) {  
                    return true;  
                } else {  
                    *count = num;  
                    if (isLetter(name, count)) {  
                        return true;  
                    }  
                }  
            }  
        }  
    }  
    return false;  
}
```

Проверка идёт на принадлежность к значениям:

- 1) True 2) FALSE
- 3) идентификатор 4) NOT(операнд)
- 5) операция(операнды)

### 3.4 Функции

1) *isTrue(string basicString, int \*count)*

2) *isFalse(string basicString, int \*count)*

3) *isLetter(string basicString, int \*count)*

#### 4) *check(string basicString, int \*count)*

```
bool isTrue(string basicString, int *count) {
    int num = *count;
    if (basicString[num] == 'T' && basicString[num + 1] == 'R' && basicString[num + 2]
== 'U' &&
        basicString[num + 3] == 'E') {
        *count = *count + 4;
        return true;
    } else {
        return false;
    }
}

bool isFalse(string basicString, int *count) {
    int num = *count;
    if (basicString[num] == 'F' && basicString[num + 1] == 'A' && basicString[num + 2]
== 'L' &&
        basicString[num + 3] == 'S' && basicString[num + 4] == 'E') {
        *count = *count + 5;
        return true;
    } else {
        return false;
    }
}

bool isLetter(string basicString, int *count) {
    int num = *count;
    if (isalpha(basicString[num]) && check(basicString, count)) {
        *count = *count + 1;
        return true;
    } else {
        return false;
    }
}
```

Функции проверяют принадлежность кусочка строки (на который указывает параметр `int *count`) к словам TRUE, FALSE, идентификатор (то есть буква).

**3.5** Функция *check(string basicString, int \*count)* является проверкой того, что буква не является частью слов TRUE, FALSE, OR, NOT, AND .

```
bool check(string basicString, int *count) {
    int num = *count;

    if (basicString[num] == 'T' && basicString[num + 1] == 'R' && basicString[num + 2]
== 'U' &&
        basicString[num + 3] == 'E') {
        return false;
    }

    if (basicString[num] == 'F' && basicString[num + 1] == 'A' && basicString[num + 2]
== 'L' &&
        basicString[num + 3] == 'S' && basicString[num + 4] == 'E') {
        return false;
    }

    if (basicString[num] == 'N' && basicString[num + 1] == 'O' && basicString[num + 2]
== 'T') {
        return false;
    }

    if (basicString[num] == 'A' && basicString[num + 1] == 'N' && basicString[num + 2]
== 'D') {
        return false;
    }

    if (basicString[num] == 'O' && basicString[num + 1] == 'R') {
        return false;
    }

    return true;
}
```

**3.6** Функция *isOperandWithNot(string basicString, int \*count)* проверяет кусочек строки на принадлежность к значению **NOT** (*операнд*) и запускает функция *isOperand()*.

```
bool isOperandWithNot(string basicString, int *count) {
    int num = *count;
    if (basicString[num] == 'N' && basicString[num + 1] == 'O' && basicString[num + 2]
== 'T' &&
        basicString[num + 3] == '(') {
```

```

        *count = *count + 4;
        if (isOperand(basicString, count)) {
            int s = *count;
            if (basicString[s] == ')') {
                *count = *count + 1;
                return true;
            } else {
                return false;
            }
        }
    } else {
        return false;
    }
}

```

### 3.7 Функция *isOperationWithNot(string basicString, int \*count)*

проверяет кусочек строки на принадлежность к значению **операция (операнды)** и запускает функция *isOperands()*.

```

bool isOperationWithOperands(string basicString, int *count) {
    int num = *count;
    if (basicString[num] == 'O' && basicString[num + 1] == 'R' && basicString[num + 2]
    == '(') {
        *count = *count + 3;
        if (isOperands(basicString, count)) {
            int s = *count;
            if (basicString[s] == ')') {
                *count = *count + 1;
                return true;
            }
        }
    }

    if (basicString[num] == 'A' && basicString[num + 1] == 'N' && basicString[num + 2]
    == 'D' &&
        basicString[num + 3] == '(') {
        *count = *count + 4;
        if (isOperands(basicString, count)) {
            int s = *count;
            if (basicString[s] == ')') {
                *count = *count + 1;
                return true;
            }
        }
    }

    return false;
}

```



### 3.8 Функция *isOperands* (string basicString, int \*count)

проверяет кусочек строки на принадлежность к значению *операнды* и запускает функция *isOperand()* и *isOperands()*.

```
bool isOperands(string basicString, int *count) {
    int num = *count;

    if (isOperand(basicString, count)) {
        int s = *count;

        if (basicString[s] != ',') {
            return true;
        } else {
            *count = num;
        }
    }

    if (isOperand(basicString, count)) {
        int s = *count;

        if (basicString[s] == ',') {
            *count = *count + 1;
            if (isOperands(basicString, count)) {
                return true;
            }
        } else {
            *count = num;
        }
    }

    return false;
}
```

### 3.9 Функция *isOperand* (string basicString, int \*count)

проверяет кусочек строки на принадлежность к значению *операнд*.

Запускает функцию *isItLogicalExpression(basicString, count)*

```
bool isOperand(string basicString, int *count) {
    if (isItLogicalExpression(basicString, count)) {
        return true;
    }

    return false;
}
```

## 4 Тесты

1.1 Покажем что, программа корректно работает на простых примерах (TRUE, FALSE, идентификатор):

Ответ ok

- 1) TRUE
- 2) FALSE
- 3) A
- 4) q

Ответ not ok

- 1) TRU
- 2) Q8
- 3) AAAA

1.2 Покажем что, программа корректно работает на простых рекурсивных примерах (операция, операнды, операнд):

Ответ ok

- 1) NOT(TRUE)
- 2) NOT(FALSE)
- 3) OR(L)
- 4) AND(TRUE)

Ответ not ok

- 1) NOT(TRUE
- 2) OT(FALSE)
- 3) OR(La)
- 4) AND(TR

1.3 Покажем что, программа корректно работает на сложных рекурсивных примерах (операция, операнды, операнд):

Ответ ok

- 1) OR(TRUE,OR(TRUE))
- 2) AND(q,l,AND(TRUE,d,OR(k,g,f,d)))
- 3) AND(l,h,NOT(OR(TRUE,y,FALSE)))
- 4) AND(l,h,NOT(OR(TRUE,y,AND(c,f))))
- 5) OR(OR(OR(OR(OR(OR(q))))))

Ответ not ok

- 1) OR(TRUE,OR(TRUE).
- 2) AND(q,l,AND(,d,OR(k,g,f,d)))
- 3) AND(q,l,AND(qd,OR(k,g,f,d)))
- 4) AND(q,l,AND(7,f,OR(k,g,f,d)))
- 5) OR(OR(OR(OR(OR(OR(q))).))

**Вывод:** был построен синтаксический анализатор для понятия логическое\_выражение. Изучена работа рекурсии для реализации решения по проверке выражения на логическое\_выражение. Были написаны тесты и проверена работоспособность программы.