

**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: Программирование рекурсивных алгоритмов**

Студент гр. 7383

Ласковенко Е.А.

Преподаватель

Размочаева Н.В.

Санкт-Петербург

2018

## Содержание

Цель работы.....	3
Задача.....	3
Реализация задачи.....	4
Тестирование программы.....	10
Выводы.....	11
Приложение А. Изменения кода.....	12
Приложение Б. Тестовые случаи.....	12
Приложение В. Код программы.....	16

## Цель работы

Познакомиться с основными понятиями и приемами рекурсивного программирования, получить навыки программирования рекурсивных функций на языке программирования C++.

## Задача

Построить синтаксический анализатор для определяемого далее понятия *константное\_выражение*:

*константное\_выражение ::= ряд\_цифр | константное\_выражение  
знак\_операции константное\_выражение*

*знак\_операции ::= + | - | \**

*ряд\_цифр ::= цифра | цифра ряд\_цифр*

## **Реализация задачи**

### **Описание функций и переменных**

#### **Функция `bool const_expression(istream &is, bool cur=false):`**

Рекурсивная функция, которая берет из входного потока `istream is` символ, выводит его на экран, проверяет является ли данный символ цифрой или знаком операции, и, если символ не нарушает заданного определения, продолжает рекурсивный запуск до тех пор, пока в потоке не закончатся данные или пока введенный символ не нарушит определение. В результате функция возвращает значение `true` или `false`.

#### **Функция `int main():`**

В функции `main` происходит объявление потоков для файла `filebuf file` и для данных с консоли `stringbuf exp`, происходит вывод на экран сообщений, содержащих информацию о меню программы, далее, в зависимости от введенной цифры (или невалидных данных) выполняется тело цикла, в котором содержатся условия типа `switch-case`.

При введении пользователем „1“, программа запрашивает у пользователя имя файла, и если файл с введенным названием существует, объявляется входной поток `istream is_file(&file)`, инициализируется адресом файла-буфера, затем вызывается рекурсивная функция, описанная выше, аргументом которой является входной поток файла-буфера. После завершения работы рекурсии поток файла закрывается и происходит вывод на экран сообщения, по которому можно судить о валидности входных данных, содержащихся в файле, в соответствии с заданием. Далее пользователю предлагается выбрать следующий пункт меню.

При введении пользователем „2“, программа запрашивает у пользователя выражение, объявляется входной поток `istream is_str(&exp)`, инициализируется адресом строки-буфера, затем помещается в строку-буфер считанная строка с консоли, далее вызывается рекурсивная функция, описанная выше, аргументом которой является входной поток строки-буфера.

После завершения работы рекурсии происходит вывод на экран сообщения о правильности выражения, содержащегося в файле. После пользователю предлагается выбрать следующий пункт меню.

При введении пользователем „0“ происходит выход из тела цикла и программа завершает свою работу.

При введении любых других данных пользователю предлагается выбрать один из пунктов меню, описанных выше.

#### **Поток `istream &is`:**

Является первым аргументом рекурсивной функции. Функция потока — содержание в нем входных данных.

#### **Переменная `bool cur`:**

Является вторым аргументом рекурсивной функции. Представляет собой флаг. Необходима для содержания в ней двух значений — `true` или `false`, которые свидетельствуют о том, являлся ли считанный символ в предыдущем запуске функции цифрой или нет. Начальное значение — `false`.

#### **Поток `filebuf file`:**

Представляет собой буфер файла, в котором содержатся данные, содержащиеся в файле.

#### **Переменная `string file_name`:**

Строка, содержащая название файла, с которого необходимо считать выражение.

#### **Поток `stringbuf exp`:**

Представляет собой буфер строку, в который помещаются данные, считанные с консоли.

#### **Переменная `string temp_str`:**

Строка, необходимая для считывания и помещения в нее входных данных с консоли.

#### **Переменная `int c_temp`:**

Целочисленная переменная, которая нужна для условий switch-case.  
Начальное значение - «-1».

**Переменная bool res:**

Переменная, которая хранит в себе значение true, если введенное выражение верно или false, если входные данные не соответствуют определению выражения. Начальное значение — false.

**Поток istream is\_file(&file):**

Копия класса входного потока, инициализируемая при считывании с файла данных и помещаемая в рекурсивную функцию.

**Поток istream is\_str(&exp):**

Копия класса входного потока, инициализируемая при считывании с консоли данных и помещаемая в рекурсивную функцию.

**Переменная char c:**

Символьная переменная, в которую помещаются символы с потока.

Все связи между функциями реализованы в Makefile, программа собирается в файл с названием «start». При необходимости в Makefile прописана цель «clean» для удаления всех скомпилированных функций и файла «main».

## Описание работы алгоритма

Возьмем следующие входные данные:  $2 * 57$ . Очевидно, что это выражение является правильным.

```
Enter "1" to input data from file.  
Enter "2" to input data from console.  
Enter "0" to exit.  
2  
Enter an expression:  
2 * 57
```

Рисунок 1 — Ввод входных данных

На первом шаге выполнения алгоритма в символьную переменную вводится цифра 2, она имеет код ASCII, равный 50, следовательно первое условие (проверка на цифру) срабатывает, флаг становится true.

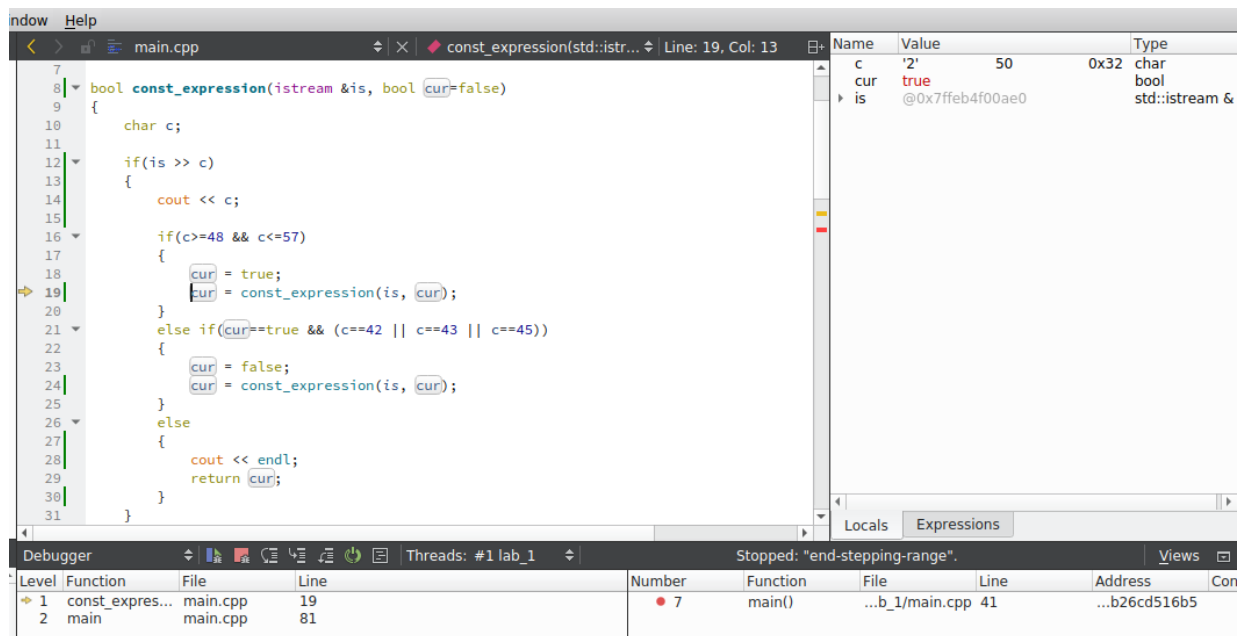


Рисунок 2 — Первый шаг алгоритма

На втором шаге в символьную переменную вводится знак операции \*, он имеет код ASCII, равный 42, следовательно срабатывает второе условие, флаг становится false.

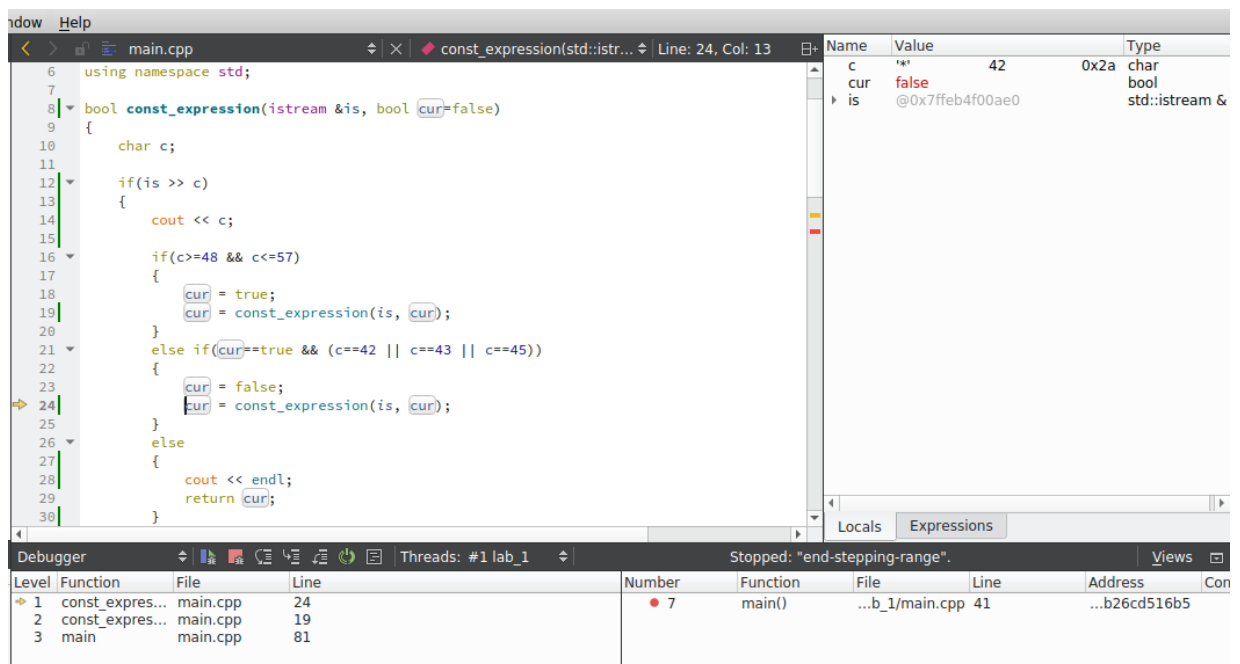
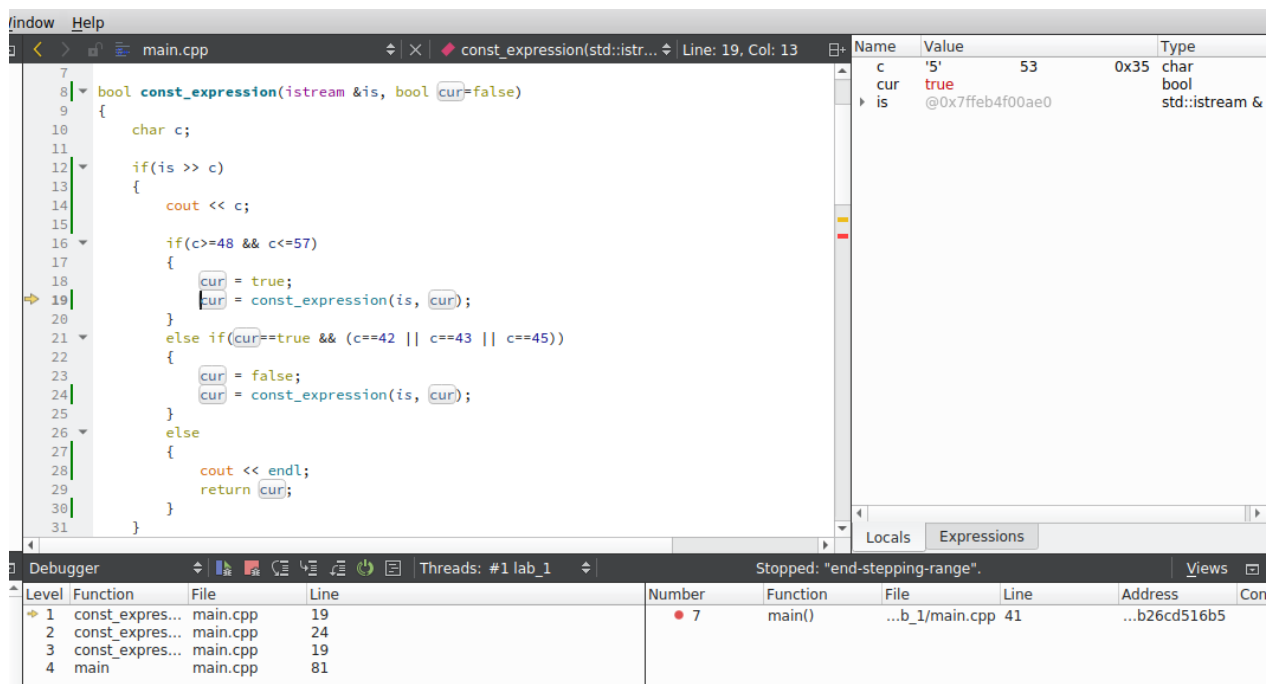


Рисунок 3 — Второй шаг алгоритма

На третьем шаге выполнения алгоритма в символьную переменную вводится цифра 5, она имеет код ASCII, равный 53, следовательно первое



условие срабатывает, флаг становится true.

Рисунок 4 — Третий шаг алгоритма

На четвертом шаге в символьную переменную вводится цифра 7, она имеет код ASCII, равный 55, первое условие срабатывает, флаг остается true.



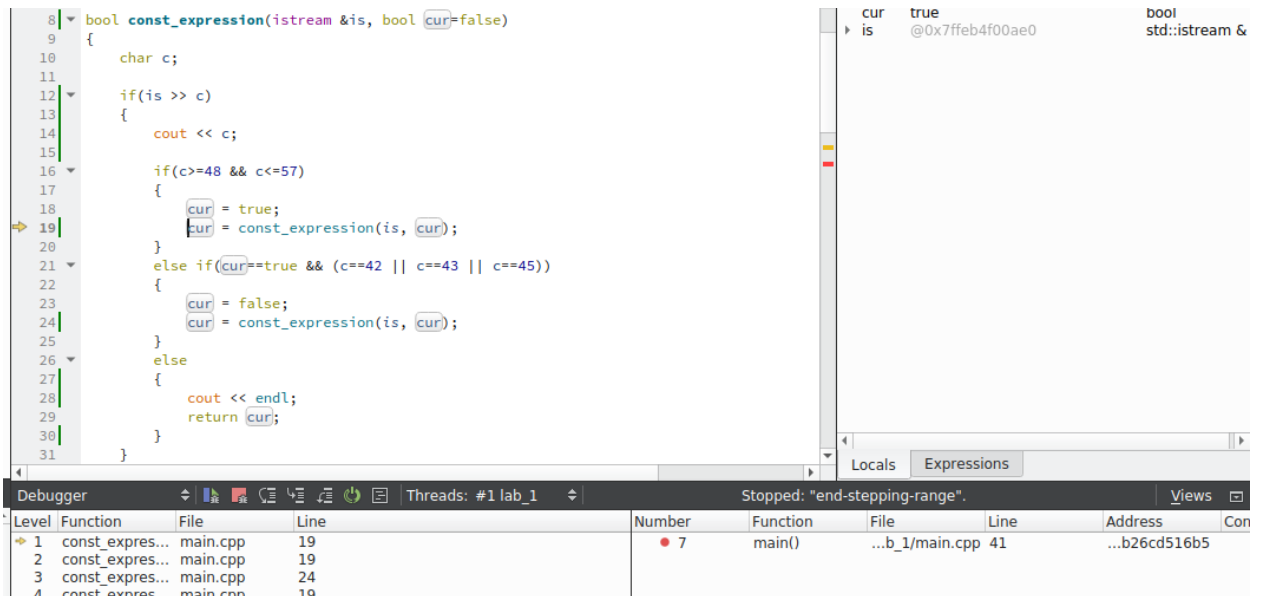


Рисунок 5 — Четвертый шаг алгоритма

На пятом шаге в символьную переменную ничего не заносится, так как входной поток пуст, все условия пропускаются, функция доходит до терминального условия, возвращает значение флага — true.

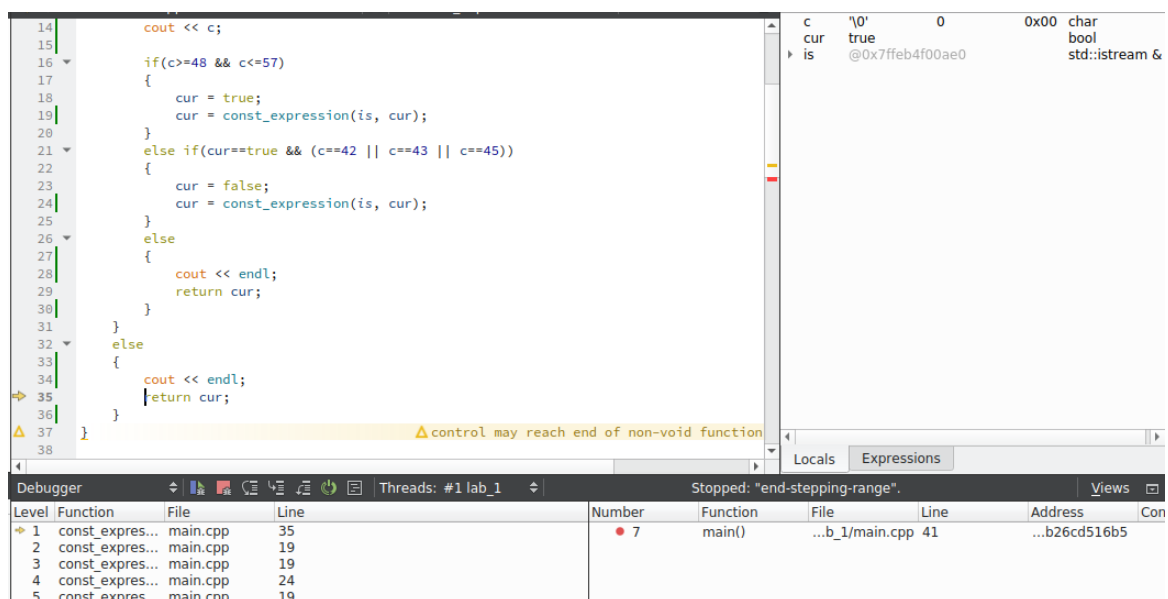


Рисунок 6 — Пятый шаг алгоритма

```

Enter "1" to input data from file.
Enter "2" to input data from console.
Enter "0" to exit.
2
Enter an expression:
2 * 57
2*57
It is an expression.

```

Рисунок 7 — Результат работы алгоритма

## **Тестирование программы**

### **Процесс тестирования**

Программа собрана в операционной системе Kubuntu 18.02, с использованием компилятора G++. В других ОС и компиляторах тестирование не проводилось.

### **Результаты тестирования**

Тестовые случаи представлены в Приложении.

По результатам тестирования было показано, что поставленная задача была выполнена.

## **Выводы**

В ходе лабораторной работы были получены навыки работы с рекурсивными функциями. Рекурсивные функции позволяют упростить написание и повысить читабельность кода, рекурсия удобна в использовании при работе с повторяющимися типами данных.

## ПРИЛОЖЕНИЕ А

### Тестовые случаи

Входные данные	Вывод	Верно?
$2 * 57$	It is an expression.	да
$+ 5546$	It isn't an expression.	да
1561	It is an expression.	да
$2525 *$	It isn't an expression.	да
-	It isn't an expression.	да
few	It isn't an expression.	да

## ПРИЛОЖЕНИЕ Б

### Изменения кода

При реализации меню возникла следующая ошибка — после успешного завершения алгоритма и выбора ввода новых данных программа перестала правильно считывать входные данные. Ниже представлены скриншоты, демонстрирующие исходный код и метод решения:

```
        cur = false;
        cur = const_expression(is, cur);
    }
    else
        return cur;
}
else
    return cur;|
}

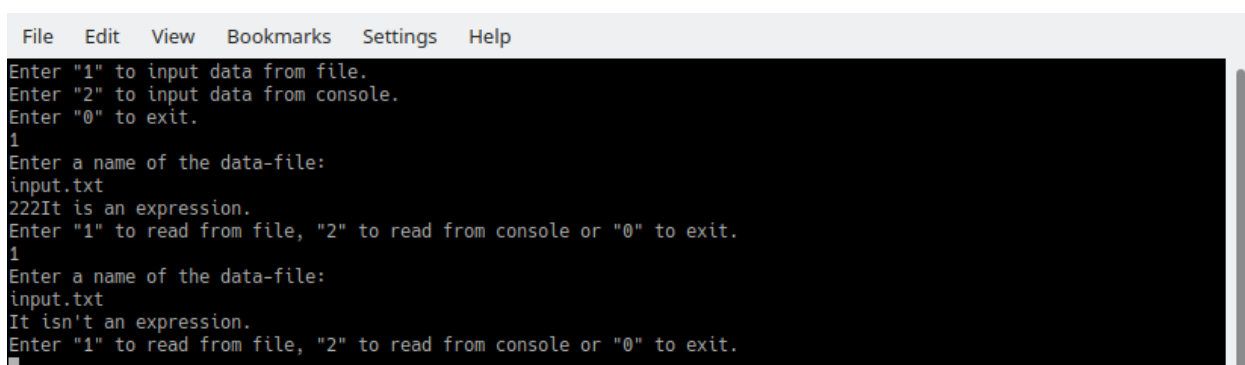
int main()
{
    filebuf file;
    istream is_file(&file);
    string file_name;
    stringbuf exp;
    istream is_str(&exp);
    string temp_str;
    int c_temp = -1;
    bool res = false;

    cout << "Enter \"1\" to input data from file." << endl
         << "Enter \"2\" to input data from console." << endl
         << "Enter \"0\" to exit." << endl;
    cin >> c_temp;

    while(c_temp!=0)
    {
        switch(c_temp)
        {
            case 1:
                cout << "Enter a name of the data-file:" << endl;
                cin >> file_name;
                if(!file.open(file_name, ios::in))
                    cout << "Input file isn't opened." << endl;
                else
                {

```

Рисунок 8 — Часть исходного кода до решения ошибки

The screenshot shows a text-based interface of a program. It starts with three prompts: "Enter '1' to input data from file.", "Enter '2' to input data from console.", and "Enter '0' to exit.". The user enters '1'. Then it asks "Enter a name of the data-file:" and the user enters "input.txt". The program then outputs "222It is an expression." which is the error. It then shows the same prompts again. The user enters '1' again, enters "input.txt", and the program outputs "It isn't an expression." which is the fix. The interface has a menu bar with "File", "Edit", "View", "Bookmarks", "Settings", and "Help".

```
File Edit View Bookmarks Settings Help
Enter "1" to input data from file.
Enter "2" to input data from console.
Enter "0" to exit.
1
Enter a name of the data-file:
input.txt
222It is an expression.
Enter "1" to read from file, "2" to read from console or "0" to exit.
1
Enter a name of the data-file:
input.txt
It isn't an expression.
Enter "1" to read from file, "2" to read from console or "0" to exit.
```

Рисунок 9 — Результат работы программы после решения ошибки

```

cout << "Enter \"1\" to input data from file." << endl
<< "Enter \"2\" to input data from console." << endl
<< "Enter \"0\" to exit." << endl;
cin >> c_temp;

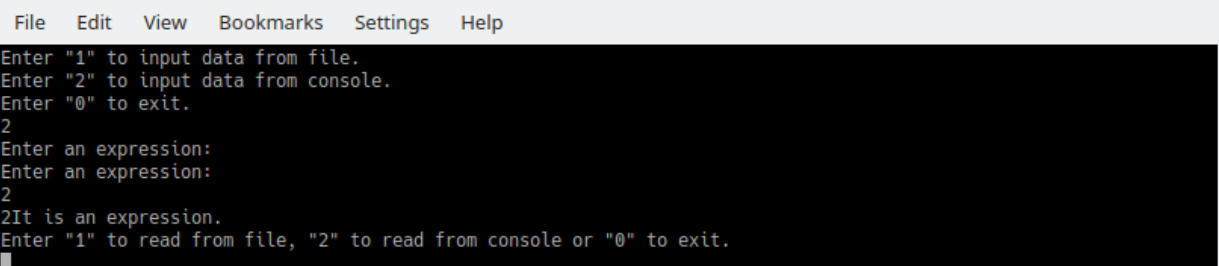
while(c_temp!=0)
{
    switch(c_temp)
    {
        case 1:
            cout << "Enter a name of the data-file:" << endl;
            cin >> file_name;
            if(!file.open(file_name, ios::in))
                cout << "Input file isn't opened." << endl;
            else
            {
                istream is_file(&file);
                res = const_expression(is_file);
                file.close();
                c_temp = 3;
            }

            break;
        case 2:
            cout << "Enter an expression:" << endl;
            getline(cin, temp_str);
            if(temp_str != "\\0")
            {
                istream is_str(&exp);
                exp.str(temp_str);
                res = const_expression(is_str);
                c_temp = 3;
            }
            break;
    }
}

```

Рисунок 10 — Часть исходного кода после решения ошибки

Также возникла ошибка — при выборе пункта меню ввода с консоли первый пробег по циклу завершался, данные не считывались. Исходный код и метод решения представлены ниже:



The screenshot shows a text-based menu with three options: '1' for file input, '2' for console input, and '0' to exit. The user enters '2' and is prompted to enter an expression. They enter '2', which is then displayed as 'It is an expression.' The program then prompts the user to choose between '1' to read from file, '2' to read from console, or '0' to exit.

```

File Edit View Bookmarks Settings Help
Enter "1" to input data from file.
Enter "2" to input data from console.
Enter "0" to exit.
2
Enter an expression:
Enter an expression:
2
It is an expression.
Enter "1" to read from file, "2" to read from console or "0" to exit.

```

Рисунок 11 — Результат работы программы до решения ошибки

```

int main()
{
    filebuf file;
    string file_name;
    stringbuf exp;
    string temp_str;
    int c_temp = -1;
    bool res = false;

    cout << "Enter \"1\" to input data from file." << endl
         << "Enter \"2\" to input data from console." << endl
         << "Enter \"0\" to exit." << endl;
    cin >> c_temp;
    cin.ignore();

    while(c_temp!=0)
    {
        switch(c_temp)
        {
            case 1:
                cout << "Enter a name of the data-file:" << endl;
                cin >> file_name;
                cin.sync();
                if(!file.open(file_name, ios::in))
                    cout << "Input file isn't opened." << endl;
                else
                {
                    istream is_file(&file);
                    res = const_expression(is_file);
                    file.close();
                    c_temp = 3;
                }
            }
        }
    }
}

```

Рисунок 12 — Часть исходного кода после решения ошибки

После очередного тестирования обнаружилась еще одна ошибка алгоритма. Выражение, к примеру, «6545у» считалось правильным, хотя это, очевидно, не так.

```

Do you want to enable algorithm visualization? [Y/N]
Y
Wrong answer. Type 'Y' or 'N'.
Do you want to enable algorithm visualization? [Y/N]
N
Enter "1" to input data from file.
Enter "2" to input data from console.
Enter "0" to exit.
2
Enter an expression:
6545y
6545y
Result: It is an expression.
Enter "1" to read from file, "2" to read from console or "0" to exit.

```

Рисунок 13 — Результат работы программы до решения ошибки

```

1
if(visual)
{
    cout <<"STEP"<<'{'<<step<<": " << " IS OPERATOR."<<endl;
    step++;
}
cur = false;
cur = const_expression(is, visual, step,cur);
}
else
{
    if(visual)
        cout <<"STEP"<<'{'<<step<<": " << " IS INVALID. Algorithm ends."<<endl;
        cout << endl;
        return cur = false;
    }
}
else
{
    cout << endl;
    return cur;
}

```

⚠ control may reach end of non-void function

Рисунок 14 — Часть исходного кода после решения ошибки

## ПРИЛОЖЕНИЕ В

### Код программы

#### 1. Код **main.cpp**:

```
#include <iostream>
#include <fstream>
#include <sstream>
#include "function.hpp"
using namespace std;
int main()
{
    filebuf file;
    string file_name;
    stringbuf exp;
    string temp_str;
    int c_temp = -1;
    bool res = false;
    int visual = 0;
    char vis_ch;

    while(c_temp!=0)
    {
        cout << "Do you want to enable algorithm visualization? [Y/N]" <<
endl;
        cin >> vis_ch;
        cin.ignore();
        if(vis_ch == 'Y' || vis_ch == 'N')
        {
            c_temp = 0;
            if(vis_ch == 'Y')
                visual = 1;
            else
                visual = 0;
        }
        else
            cout << "Wrong answer. Type 'Y' or 'N'." << endl;
    }
    cout << "Enter \"1\" to input data from file." << endl
        << "Enter \"2\" to input data from console." << endl
        << "Enter \"0\" to exit." << endl;
    cin >> c_temp;
    if(c_temp == 3)
        c_temp = -1;
    cin.ignore(); //Игнорим символ переноса каретки
    while(c_temp!=0)
    {
        switch(c_temp)
        {
            case 1: //Кейс считывания с файла
                cout << "Enter a name of the data-file:" << endl;
                cin >> file_name;
                if(!file.open(file_name, ios::in))
                    cout << "Input file isn't opened." << endl;
                else
                {
                    istream is_file(&file);
                    streamsize size = file.in_avail();
                    streamsize c_size = 0;
                    for(c_size=0; c_size<size - 1; c_size++)
                        temp_str += file.sgetc();
```



```

        cout << "File contains: " << temp_str << endl;;
        temp_str = '\0';
        res = const_expression(is_file, visual);
        file.close();
        c_temp = 3;
    }
    break;
case 2: //Кейс считывания с консоли
    cout << "Enter an expression:" << endl;
    getline(cin, temp_str);
    if(temp_str != "\0")
    {
        istream is_str(&exp); //Объявляем входной поток,
инициализируем адресом строки-буфера
        exp.str(temp_str); //Помещаем в строку-буфер считанную строку
с входного потока
        res = const_expression(is_str, visual);
        temp_str = '\0';
        c_temp = 3;
    }
    break;
case 0: //Кейс выхода
    break;
case 3: //Кейс проверки
    cout << "Result: ";
    if(res)
        cout << "It is an expression." << endl;
    else
        cout << "It isn't an expression." << endl;
    c_temp = -1;
    break;
default: //Кейс других входных данных
    cout << "Enter \"1\" to read from file, \"2\" to read from
console or \"0\" to exit." << endl;
    cin >> c_temp;
    cin.ignore(); //Игнорим '\n'
    break;
}
}
return 0;
}

```

## 2. Код function.cpp:

```

#include <iostream>
#include "function.hpp"
using namespace std;
bool const_expression(istream &is, int visual, int step, bool cur){
    char c;
    if(is >> c){
        if(!visual)
            cout << c;

        if(c>=48 && c<=57){
            if(visual){
                cout <<"STEP"<<{' '<<step<<"}: "<<c<<" IS DIGIT."<<endl;
                step++;
            }
            cur = true;
            cur = const_expression(is, visual, step, cur);
        }
    }
}

```

```

else if(cur==true && (c==42 || c==43 || c==45)){
    if(visual){
        cout <<"STEP"<<'{'<<step<<": "<<c<<" IS OPERATOR."<<endl;
        step++;
    }
    cur = false;
    cur = const_expression(is, visual, step,cur);
}
else if(visual && cur==false){
    cout <<"STEP"<<'{'<<step<<": "<<c<<" IS ODD OPERATOR OR INVALID
TOKEN. Algorithm ends."<<endl;
    cout << endl;
    return cur = false;
}
else{
    if(visual)
        cout <<"STEP"<<'{'<<step<<": "<<c<<" IS INVALID TOKEN.
Algorithm ends."<<endl;
    cout << endl;
    return cur = false;
}
}
else{
    if(visual && cur==false){
        cout <<"STEP"<<'{'<<step<<": "<<c<<" MISSING DIGIT. Algorithm
ends."<<endl;
        cout << endl;
        return cur = false;
    }
    cout << endl;
    return cur;
}
}
}

```

### 3. Код function.hpp:

```

#pragma once
#include <iostream>
using namespace std;
bool const_expression(istream &is, int visual, int step=1, bool cur=false);

```

### 4. Код Makefile:

```

all: main.o function.o
    g++ main.o function.o -o start
main.o: main.cpp function.hpp
    g++ -c main.cpp
function.o: function.cpp function.hpp
    g++ -c function.cpp
clean:
    rm *.o start

```