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

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Алгоритмы и структуры данных»

Тема: Стек и очередь

Студент гр. 8381 Звегинцева Е.Н.

Преподаватель Жангиров Т.Р.

Санкт-Петербург
2019

Цель работы

Изучение и сравнение различных методов численного интегрирования на примере составных формул прямоугольников, трапеций, Симпсона.

Задание

В заданном текстовом файле F записано логическое выражение (ЛВ) в следующей форме:

<ЛВ> ::= true | false | (!<ЛВ>) | (<ЛВ> ^ <ЛВ>) | (<ЛВ> v <ЛВ>)

где знаки !, ^ и v обозначают соответственно отрицание, конъюнкцию и дизъюнкцию. Вычислить значение этого выражения.

Выполнение работы

В ходе данной работы была разработана дополнительная структура данных стек, а так же все необходимые методы для работы с ним, исходный код представлен в приложении А.

Для упрощения рассчетов, были написаны методы преобразования входных данных в постфиксную форму и подсчета этого выражения.

Постфиксной *формой записи выражения aDb называется запись, в которой знак операции размещен за операндами: abD*

Методы:

- bool **test_operation**(char) – проверка на корректность знака операции
- void **toPostfixForm**(string &, string&, string&) – приводит строку в постфиксную форму, посредством стека
- void **step**(char, Stack<int>&, string&,string&) – вычисляет подвыражения, доставая элементы из стека и совершая с ними отрицание, конъюнкцию и дизъюнкцию
- int **calc**(const string&, string&,string&) – вычисляет выражение, проходясь по всем операциям, вызывая метод **onestep**

- void **onestep**(char, Stack<int>&, string &,string&) - определяет является ли входной символ знаком операции, если да вызывает функцию **step,** если нет, определяет является ли символ числом, если да, то кладет элемент в стек

На всех этапах работы при некорректности какого-либо ввода, программа будет сообщать об ошибке.

Также был разработан графический интерфейс, с возможность считывать данные как с консоли, так и из файла, а также записывать результат в него.

Оценка эффективности алгоритма.

Алгоритм вычисления значения логического выражения посредством записи его в постфиксной форме с использованием стека имеет линейную сложность O(n).

При чтении выражения слева направо в вершину стека помещаются операнды. Как только при чтении встречается знак арифметической операции, из стека извлекаются два последних операнда, к ним применяется текущая операция, и результат записывается обратно в вершину стека. По завершении работы алгоритма в стеке оказывается один элемент – значение арифметического выражения.

Оценка алгоритма осуществляется соотношением количества элементов к количеству операций, в виду того, что мы опускаем время выполнения различных логических операций в следствии с сильной зависимость от косвенных факторов(загруженность системы и тп). У нас наблюдается точное соответствие числа операндов количеству чисел, которые должны находится в стеке для корректного завершения процесса вычисления.

На основе промежуточных выводов через qDebug() в ходе выполнения программы и последующего анализа выходной последовательности операций был построен график, представленный на рис. 1.

Как мы видим, наш алгоритм обладает линейной сложностью.

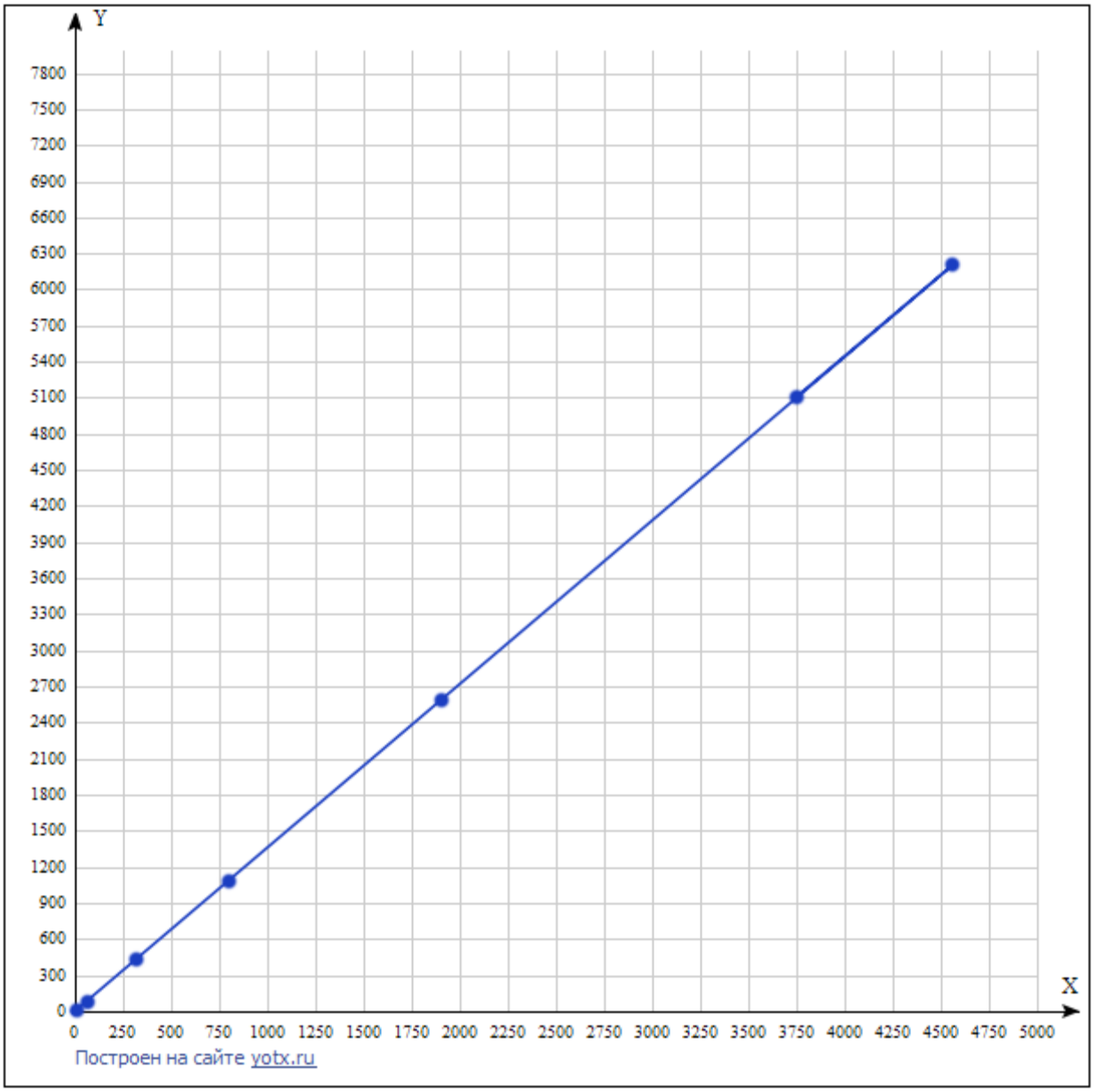

Рисунок 1 - анализ эффективности алгоритма

Тестирование программы.

Результаты тестирования программы представлены в табл.1.

Таблица 1 - Результаты тестирования программы

Входное выражение	Результат вычислений
((1^0)v(!(1v0)))	0
1	1
(2v0)	Uncorrect test!
(!(((1v0)^1)v(!(0v0))))	0
((1v0)^1	Check input string!

Выводы.

В ходе лабораторной работы был разработан способ вычисления логического выражения в постфиксной форме и имеет линейную сложность. А также реализован пошаговый режим выполнения алгоритма, консольный и графический интерфейсы.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

```
#include "mainwindow.h"

#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();
    return a.exec();
}
```

Название файла: mainwindow.cpp

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
static Stack< int> stack{};
static int i = 0;
static string answer{};
static string inputTest{};
static string err{};
static string postfixForm;
MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);
}

MainWindow::~MainWindow()
{
    delete ui;
}


void MainWindow::on_result_clicked()
{
    string result, postfixForm, error;
    string str = ui->input->text().toUtf8().constData();
    toPostfixForm(str, postfixForm, error);
    if(!error.empty()){
        ui->output->setText(QString::fromStdString(error));
        return;
    }
    int ans = calc(postfixForm, result, err);
    string str1 = "Результат вычислений: ";
    if(!err.empty())
        str1 += err;
    else
        str1 += to_string(ans);
    ui->output->setText(QString::fromStdString(str1));

}
```

```
void MainWindow::on_step_clicked()
{
    string str = ui->input->text().toUtf8().constData(), error;
    if(inputTest.compare(str)!=0){
        i=0;
        answer.clear();
        err.clear();
        inputTest.clear();
        inputTest.append(str);
        toPostfixForm(str, postfixForm, error);
        if(!error.empty()){
            ui->output->setText(QString::fromStdString(error));
            return;
        }
        ui->output->clear();
        stack.clear();
        stack.setSize(postfixForm.length());
    }
    if(!err.empty()){
        return;
    }
    if(i>=postfixForm.length() && stack.length()==1){
        i=0;
        stack.pop();
        answer.clear();
        err.clear();
    }
    onestep(postfixForm[i], stack, answer, err);
    i++;
    if(answer.empty()){
        ui->output->setText(QString::fromStdString(err));
    }
    else if(err.empty()){
        ui->output->setText(QString::fromStdString(answer));
    }
    else{
        ui->output->setText(QString::fromStdString(answer)+QString::from-
StdString(err));
    }
}

void MainWindow::on_dataFromFile_clicked()
{
    QString fileName = QFileDialog::getOpenFileName(this,
            tr("Open TXT File"),  QDir::homePath(),
            tr("TXT text (*.txt);;All Files (*)"));
    ifstream sourceFile(fileName.toUtf8().constData());
    string input;
    sourceFile >> input;
    ui->input->setText(QString::fromStdString(input));
}

void MainWindow::on_saveOutput_clicked()
{
    QString fileName = QFileDialog::getOpenFileName(this,
            tr("Open TXT File"),  QDir::homePath(),
            tr("TXT text (*.txt);;All Files (*)"));
    ofstream sourceFile(fileName.toUtf8().constData());
    sourceFile << ui->output->toPlainText().toUtf8().constData();
}
```

Название файла: methods.cpp

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
static Stack< int> stack{};
static int i = 0;
static string answer{};
static string inputTest{};
static string err{};
static string postfixForm;
MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);
}

MainWindow::~MainWindow()
{
    delete ui;
}


void MainWindow::on_result_clicked()
{
    string result, postfixForm, error;
    string str = ui->input->text().toUtf8().constData();
    toPostfixForm(str, postfixForm, error);
    if(!error.empty()){
        ui->output->setText(QString::fromStdString(error));
        return;
    }
    int ans = calc(postfixForm, result, err);
    string str1 = "Результат вычислений: ";
    if(!err.empty())
        str1 += err;
    else
        str1 += to_string(ans);
    ui->output->setText(QString::fromStdString(str1));

}

void MainWindow::on_step_clicked()
{
    string str = ui->input->text().toUtf8().constData(), error;
    if(inputTest.compare(str)!=0){
        i=0;
        answer.clear();
        err.clear();
        inputTest.clear();
        inputTest.append(str);
        toPostfixForm(str, postfixForm, error);
        if(!error.empty()){
            ui->output->setText(QString::fromStdString(error));
            return;
        }
        ui->output->clear();
        stack.clear();
        stack.setSize(postfixForm.length());
    }
```

```
    if(!err.empty()){
        return;
    }
    if(i>=postfixForm.length() && stack.length()==1){
        i=0;
        stack.pop();
        answer.clear();
        err.clear();
    }
    onestep(postfixForm[i], stack, answer, err);
    i++;
    if(answer.empty()){
        ui->output->setText(QString::fromStdString(err));
    }
    else if(err.empty()){
        ui->output->setText(QString::fromStdString(answer));
    }
    else{
        ui->output->setText(QString::fromStdString(answer)+QString::from-
StdString(err));
    }
}

void MainWindow::on_dataFromFile_clicked()
{
    QString fileName = QFileDialog::getOpenFileName(this,
            tr("Open TXT File"),  QDir::homePath(),
            tr("TXT text (*.txt);;All Files (*)"));
    ifstream sourceFile(fileName.toUtf8().constData());
    string input;
    sourceFile >> input;
    ui->input->setText(QString::fromStdString(input));
}

void MainWindow::on_saveOutput_clicked()
{
    QString fileName = QFileDialog::getOpenFileName(this,
            tr("Open TXT File"),  QDir::homePath(),
            tr("TXT text (*.txt);;All Files (*)"));
    ofstream sourceFile(fileName.toUtf8().constData());
    sourceFile << ui->output->toPlainText().toUtf8().constData();
}
```

Название файла: postfixForm.cpp

```
#include "postfixform.h"

bool test_operation(char buf)
{
    return (buf == '!' || buf=='^' || buf=='v')?  true: false;
}

void toPostfixForm(string & str, string & postfixForm, string & err)
{
    reverse(str.begin(), str.end());

        Stack<char> stack;
        for(unsigned long i=0; i<str.length(); i++){
            if(test_operation(str[i])){
                if(((i+1)!=str.length() && str[i+1]==')') || stack.isEmpty() ||
*stack.onTop()==')')
```

```
                    stack.push(str[i]);
                else{
                    if(!stack.isEmpty()){
                        postfixForm+=*stack.onTop();
                        stack.pop();
                        stack.push(str[i]);
                    }else{
                        err = "check input string\n";
                        return;
                    }
                }
            }else if(str[i]==')') {
                stack.push(str[i]);
            }else if(str[i]=='(') {
                char buf;
                do{
                    if(!stack.isEmpty()){
                        buf = *stack.onTop();
                        stack.pop();
                        if(buf != ')')
                            postfixForm+=buf;
                    }else{
                        err = "check input string\n";
                        return;
                    }
                }while(buf != ')');

            }else if(str[i]==' '){
                continue;
            }else{
                postfixForm+=str[i];
            }
        }
        if(postfixForm=="" || postfixForm.empty()){
            err = "check input string\n";
            return;
        }


}
```

Название файла: headers.h

```
#ifndef HEADERS_H
#define HEADERS_H

#include <iostream>
#include <fstream>
#include <QFileDialog>
#include <QPushButton>
using namespace std;

#endif // HEADERS_H
```

Название файла: mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include "headers.h"
```

```
#include "postfixform.h"
#include "methods.h"
QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

private slots:
    void on_result_clicked();

    void on_step_clicked();

    void on_dataFromFile_clicked();

    void on_saveOutput_clicked();

private:
    Ui::MainWindow *ui;
};
#endif // MAINWINDOW_H
```

Название файла: methods.h

```
#ifndef METHODS_H
#define METHODS_H

#include "postfixform.h"
#include "stack.h"
#include <math.h>

void step(char, Stack<int>&, string&,string&);
int calc(const string&, string&,string&);
void onestep(char, Stack<int>&, string &,string&);

#endif // METHODS_H
```

Название файла: postfixform.h

```
#ifndef PREFIXFORM_H
#define PREFIXFORM_H

#include "headers.h"
#include "stack.h"
bool test_operation(char);
void toPostfixForm(string &, string&, string&);

#endif // PREFIXFORM_H
```

Название файла: stack.h

```
#ifndef stack_h
#define stack_h

#include "headers.h"

template<class T>
class Stack
{
    public:
        Stack(size_t n=50)
        : SIZE{ n }, top{}, items{ new T[SIZE]{} }
        {}
        ~Stack() {delete[] items;}
        void push(T);
        void pop();
        T* onTop() const;
        bool isFull() const{return SIZE == top;}
        bool isEmpty() const{return top==0;}
        size_t size() const {return SIZE;}
        size_t length() const {return top;}
        void clear(){top=0;}
        void setSize(size_t n){SIZE=n; items=new T[SIZE]{}; top=0;}
    private:
        size_t SIZE;
        size_t top;
        T *items;
};

template<class T>
void Stack<T>::push(T item)
{
    if(!isFull()){
        items[top++] = item;
    }
    else
        cout<<"Стек полон\n";
}

template<class T>
void Stack<T>::pop()
{
    if(!isEmpty()){
        top--;
    }
    else
        cout<<"Стек пуст\n";
}

template<class T>
T *Stack<T>::onTop() const
{
    if(!isEmpty())
        return &items[top-1];
    else{
        cout<<"Стек пуст\n";
        return nullptr;
    }
}
#endif
```