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

ОТЧЕТ

по лабораторной работе №3
по дисциплине «Алгоритмы и структуры данных»
Тема: «Стек»

Студент гр. 8381 _____ Сосновский Д.Н.

Преподаватель _____ Жангиров Т.Р.

Вариант 7-в

Санкт-Петербург

2019

Цель работы

Ознакомиться со структурой данных «Стек» и с её помощью реализовать алгоритм вычисления значения функции определенного вида.

Постановка задачи

Дана формула вида

< формула > ::= < цифра > | *M* (< формула > , < формула >) |
m (< формула > , < формула >)

< цифра > ::= 0 | 1 | ... | 9

где *M* обозначает функцию *max*, а *m* – функцию *min*. Вычислить (как целое число) значение данной формулы. Например, *M* (5, *m*(6, 8)) = 6.

Ход работы

Мною была разработана программа, реализующая расчёт данной функции. Для этого используется структура данных «Стек» - каждый значимый элемент строки (“m”, “M”, или цифры) добавляются в стек, скобки означают, что необходимо рассчитать верхнюю функцию. Таким образом в конце остаётся один элемент, который является ответом. Для программы был разработан интерфейс, реализованный с помощью Qt. Интерфейс оснащён подсказами для полей, отметки расчёта по шагам, а так же кнопкой для расчёта. В случае расчёта по шагам программа показывает содержимое стека, а также обработанный символ на каждом шаге. Программа поддерживает запуск с консоли с повторным применением алгоритма без повторного запуска программы. Примеры работы программы приведены в приложении в разделе «Примеры работы программы», исходный код программы приведён в листинге 1.

Оценка сложности работы программы

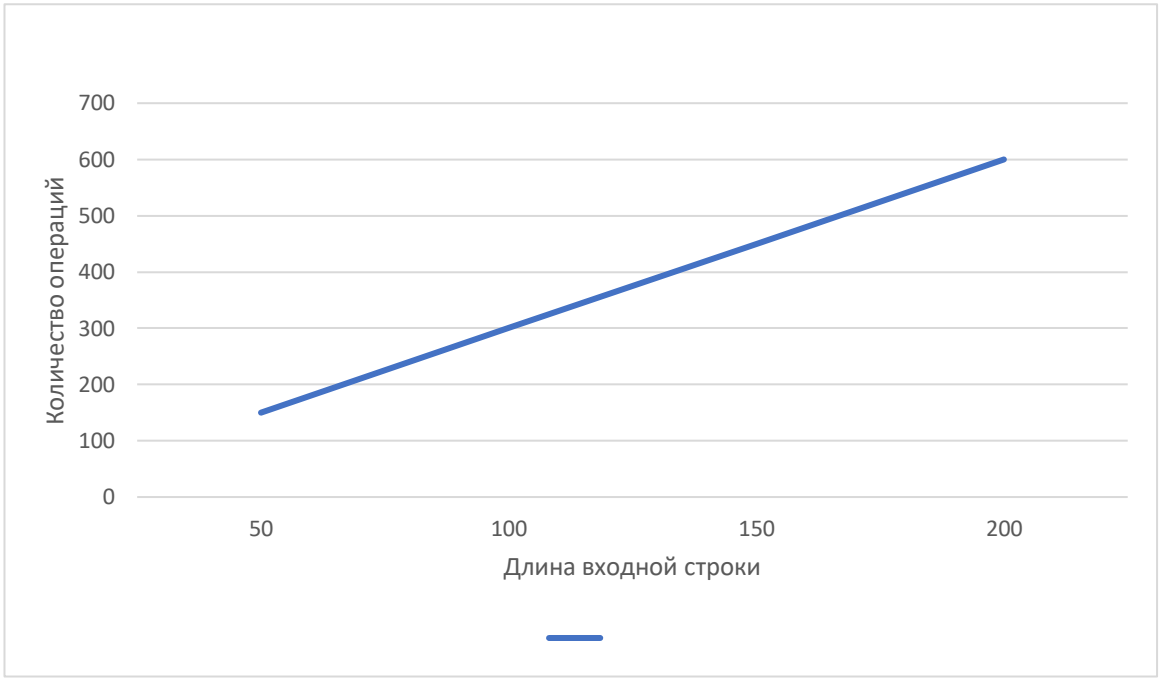
Программа работает за O(n), так как каждый символ обрабатывается один раз и число обработок символов возрастёт на k при увеличении исходного количества символов на k.

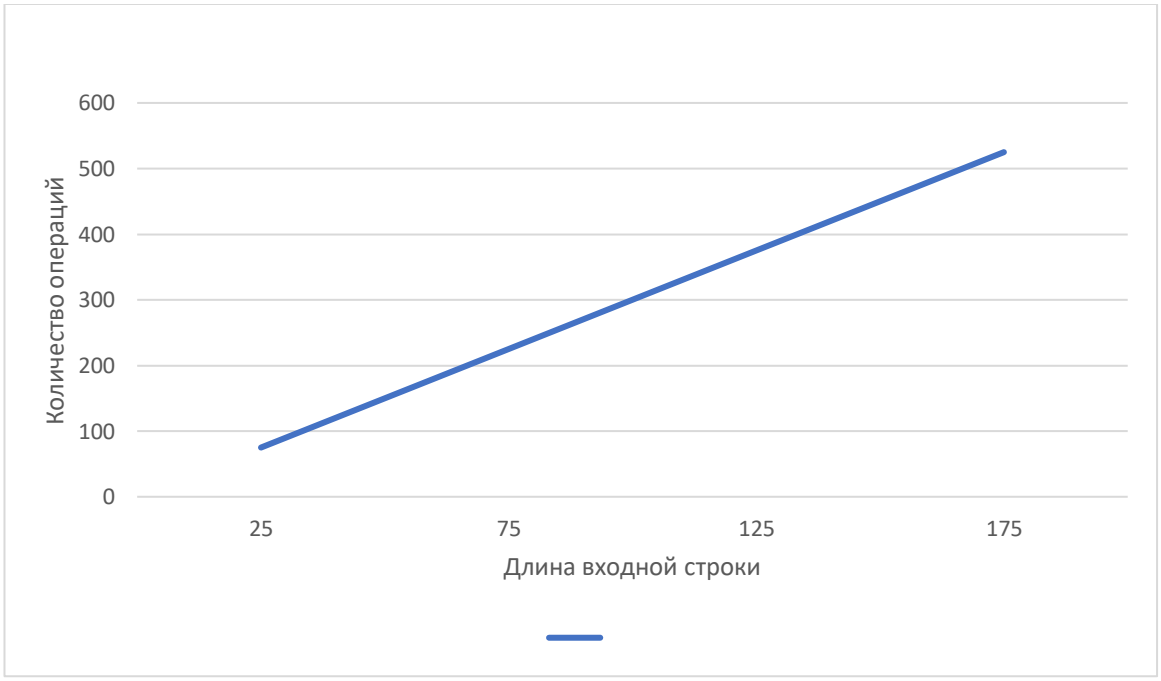
Вывод

В ходе выполнения данной работы я полностью освоил структуру данных «Стек» и теперь умею применять её для реализации алгоритмов.

ПРИЛОЖЕНИЕ

Графики сложности работы программы





В обоих случаях наблюдаем линейную зависимость, следовательно сложность программы O(n).

Примеры работы программы

В данном приложении приведены пример работы программы.

Входная функция	Ответ
9	9
m(4, 6)	4
M(4, 6)	6
M(5, 5)	5
m(5, M(5, 6))	5
m(m(m(5, 4), 3), 2)	2
M(M(M(5, 4), 3), 2)	5

Программа работает правильно.

Листинг 1.

В данном листинге приведён исход программы.

```
Function.h
#pragma once
#include "stack.h"
#include "QMessageBox"
#include <iostream>
#include <conio.h>
class Function
{
public:
    Function(QString functionString);

    QString getStr();

    int calculate();

    int calculateStepByStep();

    int calculateStepByStepForConsole();
private:
```

```
    QString functionString;

};
```

Mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H
#include "function.h"
#include <QMainWindow>

namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

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

private slots:
    void on_calculateButton_clicked();

private:
    Ui::MainWindow *ui;
};

#endif // MAINWINDOW_H
```

Stack.h

```
#pragma once
#include "QString"
#include "QCharRef"
#include "stackelement.h"
class Stack
{
public:
    Stack();

    void add(QChar element);
    QChar take();
    QString getStr();
private:
    StackElement* head;
};
```

StackElement.h

```
#pragma once
#include "QCharRef"

class StackElement
{
public:
    StackElement(QChar value);

    StackElement* nextElement;

    QChar value;
};
```

Function.cpp

```
#include "function.h"

Function::Function(QString functionString)
{
    for(auto i : functionString)  //remove spaces from the input
functionString
    {
        if(i == " ")
            functionString.remove(i);
    }

    this->functionString = functionString;
}

bool isLeftBracket(QChar element)
{
    if(element.unicode() == '(') return true;
    else return false;
}

bool isRightBracket(QChar element)
{
    if(element.unicode() == ')') return true;
    else return false;
}

int max(int firstNumber, int secondNumber)
{
    if(firstNumber >= secondNumber) return firstNumber;
    else return secondNumber;
}

int min(int firstNumber, int secondNumber)
```

```
{
    if(firstNumber <= secondNumber) return firstNumber;
    else return secondNumber;
}

int doOperation(int firstNumber, int secondNumber, QChar
operationLetter)
{
    if(operationLetter == 'M')
    {
        return max(firstNumber, secondNumber);
    }
    else if(operationLetter == 'm')
    {
        return min(firstNumber, secondNumber);
    }
}

int calculateLastKnownExpression(Stack & stack)
{
    int firstNumber = stack.take().digitValue();
    int secondNumber = stack.take().digitValue();
    QChar operationLetter = stack.take();

    int result = doOperation(firstNumber, secondNumber,
operationLetter);

    return result;
}

QString Function::getStr() { return functionString; }

int Function::calculate()
{
    Stack stack;

    for(auto currentElement : functionString)
    {
        if(currentElement == ',') continue;
        else if(isRightBracket(currentElement) == false)
        {
            if(isLeftBracket(currentElement) == false)
stack.add(currentElement);
        }
        else {
            int lastExpressionResult =
calculateLastKnownExpression(stack);
            stack.add(QString::number(lastExpressionResult).at(0));
        }
```

```
    }

    QChar ans = stack.take();
    return ans.digitValue();
}

int Function::calculateStepByStep()
{
    Stack stack;

    for(auto currentElement : functionString)
    {
        if(currentElement == ',') { }
        else if(isRightBracket(currentElement) == false)
        {
           if(isLeftBracket(currentElement) == false)
stack.add(currentElement);
        }
        else {
            int lastExpressionResult =
calculateLastKnownExpression(stack);
            stack.add(QString::number(lastExpressionResult).at(0));
        }

        QMessageBox msgBox;
        msgBox.setWindowTitle("Состояние стека");
        msgBox.setText(stack.getStr() + "\n" + "Обработанный символ:
" + currentElement);
        QPushButton *continueButton = msgBox.addButton(("Continue"),
QMessageBox::AcceptRole);
        msgBox.exec();
    }

    QChar ans = stack.take();
    return ans.digitValue();

}


int Function::calculateStepByStepForConsole()
{
    Stack stack;

    for(auto currentElement : functionString)
    {
        if(currentElement == ',') { }
        else if(isRightBracket(currentElement) == false)
        {
```

```
            if(isLeftBracket(currentElement) == false)
stack.add(currentElement);
        }
        else {
            int lastExpressionResult =
calculateLastKnownExpression(stack);
            stack.add(QString::number(lastExpressionResult).at(0));
        }

        std::cout << "Обработанный элемент: " <<
QString(currentElement).toStdString() << "\n";
        std::cout << "Содержимое стека:" << "\n";
        std::cout << stack.getStr().toStdString();
        std::cout << "\n";

        std::cout << "Для продолжения нажмите любую клавишу" <<
"\n";
        getch();
    }

    QChar ans = stack.take();
    return ans.digitValue();

}
```

Main.cpp

```
#include "mainwindow.h"
#include <QApplication>

int main(int argc, char *argv[])
{
    bool console = false;

    for(int i = 1; i < argc; i++)
    {
        if(!strcmp("console", argv[i]))
        {
            console = true;
        }
    }

    if(console)
    {
        bool onExit = false;
        while(onExit == false)
        {
            std::cout << "Введите функцию: " << "\n";
            std::string inputrstr;
```

```
            std::getline(std::cin, inputrstr);

            bool steps = false;

            char clientAns = '0';

            while(clientAns != 'y' && clientAns != 'n')
            {
                std :: cout << "Рассчитать по шагам? [y/n]" << "\n";
                std :: cin >> clientAns;
            }

            if(clientAns == 'y') steps = true;
            else steps = false;

            Function func(QString::fromUtf8(inputrstr.c_str()));

            int ans;
            if(steps == true) ans =
func.calculateStepByStepForConsole();
            else ans = func.calculate();

            std :: cout << "Ответ: " << ans << "\n";


            clientAns = '0';
            std::cout << "Ещё раз? [y/n]" << "\n";
            std::cin >> clientAns;
            if(clientAns == 'n') onExit = true;
        }
    }
    else
    {
        QApplication a(argc, argv);
        MainWindow w;
        w.show();
        return a.exec();
    }

    return 0;
}
```

Mainwindow.cpp

```
#include "mainwindow.h"
#include "ui_mainwindow.h"

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
```

```
{
    ui->setupUi(this);
}

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

void MainWindow::on_calculateButton_clicked()
{
    QString inp = ui->input->text();

    Function function(inp);
    int ans;
    if(ui->stepByStepChecked->isChecked() == false)
    {
        ans = function.calculate();
    }
    else ans = function.calculateStepByStep();

    ui->output->setText(QString::number(ans));
}
```

Stack.cpp

```
#include "stack.h"

Stack::Stack()
{
    head = nullptr;
}

void Stack::add(QChar element)
{
    StackElement* newStackElement = new StackElement(element);

    newStackElement->nextElement = head;

    head = newStackElement;
}

QChar Stack::take()
{
    StackElement* previousHead = head;

    head = previousHead->nextElement;

    QChar answer = previousHead->value;
```

```
    delete previousHead;

    return answer;
}

QString Stack::getStr()
{
    QString stackAsString = nullptr;
    StackElement* temporary = head;
    while(temporary != nullptr)
    {
        QString valueStr = temporary->value;
        if(valueStr.at(0).isDigit() == true) stackAsString += "|    "
+ valueStr + "  |" + "\n";
        else stackAsString += "|  " + valueStr + "  |" + "\n";
        temporary = temporary->nextElement;
    }

    stackAsString += "|_____|";
    return stackAsString;
}
```

StackElement.cpp

```
#include "stackelement.h"

StackElement::StackElement(QChar value):
    value(value)
{
    nextElement = nullptr;
}
```