*Федеральное государственное бюджетное образовательное учреждение* *высшего образования*

|  |  |
| --- | --- |
|  | ***«Московский государственный технический университет  имени Н.Э. Баумана»***  ***(национальный исследовательский университет)***  ***(МГТУ им. Н.Э. Баумана)*** |

ФАКУЛЬТЕТ \_\_\_\_ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ\_\_\_\_\_\_\_\_

КАФЕДРА \_\_\_\_\_\_Компьютерные Системы и сети (ИУ6)\_\_\_\_\_\_\_\_\_\_\_

**Отчет**

**по лабораторной работе № 9**

**Название лабораторной работы: Программирование с использованием библиотеки Qt**

**Дисциплина: Объектно-ориентированное программирование**

Студент гр. ИУ6-22Б  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. П. Плютто**



(Подпись, дата) (И.О. Фамилия)



Преподаватель  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ О. А. Веселовская**

(Подпись, дата) (И.О. Фамилия)

Москва, 2023

Задание 1

#include <QApplication>

#include <QHBoxLayout>

#include <QVBoxLayout>

#include <QSlider>

#include <QSpinBox>

#include <QPushButton>

// Преобразуем входную последовательность символов в кодировку UNICODE

#define RUS( str ) codec->toUnicode(str)

int main(int argc, char \*argv[])

{

QApplication app(argc, argv);

// Обеспечиваем перекодировку русских букв из кодировки,

// в которой написана программа.

// "Windows-1251" - для Windows, "KOI8-R" - Linux и т.д.

// Создаем главное окно

QWidget \*hbox = new QWidget();

hbox->setWindowTitle( ("Введите Ваш Возраст") );

QSpinBox \*spinBox = new QSpinBox( hbox );

QSlider \*slider = new QSlider(Qt::Horizontal, hbox );

spinBox->setRange(0, 130);

slider->setRange(0, 130);

spinBox->setValue(35);

QPushButton \* btn = new QPushButton( ("Завершение"), hbox );

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

QVBoxLayout \*layout = new QVBoxLayout; // выравнивание по горизонтали

//QVBoxLayout \*layout = new QVBoxLayout; // выравнивание по вертикали

layout->setContentsMargins(5,5,5,5); // устанавливаем внешние границы

layout->setSpacing(5); // устанавливаем интервал элементов внутри

hbox->setLayout(layout); // связываем layout с hbox

// устанавливаем порядок следования элементов

layout->addWidget(spinBox);

layout->addWidget(slider);

layout->addWidget(btn);

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// связываем сигнал изменения spinBox со слотом slider

QObject::connect(spinBox, SIGNAL(valueChanged(int)),

slider, SLOT(setValue(int)));

// связываем сигнал изменения slider со слотом spinBox

QObject::connect(slider, SIGNAL(valueChanged(int)),

spinBox, SLOT(setValue(int)));

// связываем сигнал нажатия btn со слотом close главного окна

QObject::connect(btn, SIGNAL(clicked(bool)),

hbox, SLOT(close()));

hbox->show(); // отображаем окно

return app.exec(); // запускаем цикл обработки сообщений

}

#include "mainwindow.h"

#include "ui\_mainwindow.h"

MainWindow::MainWindow(QWidget \*parent)

: QMainWindow(parent)

, ui(new Ui::MainWindow)

{

ui->setupUi(this);

}

MainWindow::~MainWindow()

{

delete ui;

}

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

QT\_BEGIN\_NAMESPACE

namespace Ui { class MainWindow; }

QT\_END\_NAMESPACE

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

MainWindow(QWidget \*parent = nullptr);

~MainWindow();

private:

Ui::MainWindow \*ui;

};

#endif // MAINWINDOW\_H

<?xml version="1.0" encoding="UTF-8"?>

<ui version="4.0">

<class>MainWindow</class>

<widget class="QMainWindow" name="MainWindow">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>800</width>

<height>600</height>

</rect>

</property>

<property name="windowTitle">

<string>MainWindow</string>

</property>

<widget class="QWidget" name="centralwidget"/>

<widget class="QMenuBar" name="menubar"/>

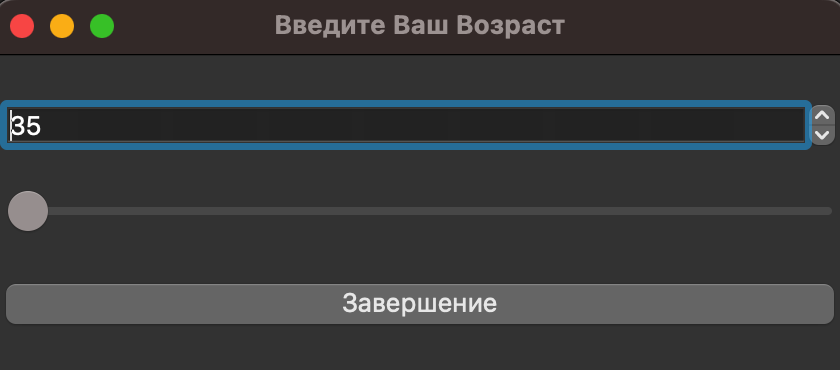
<widget class="QStatusBar" name="statusbar"/>

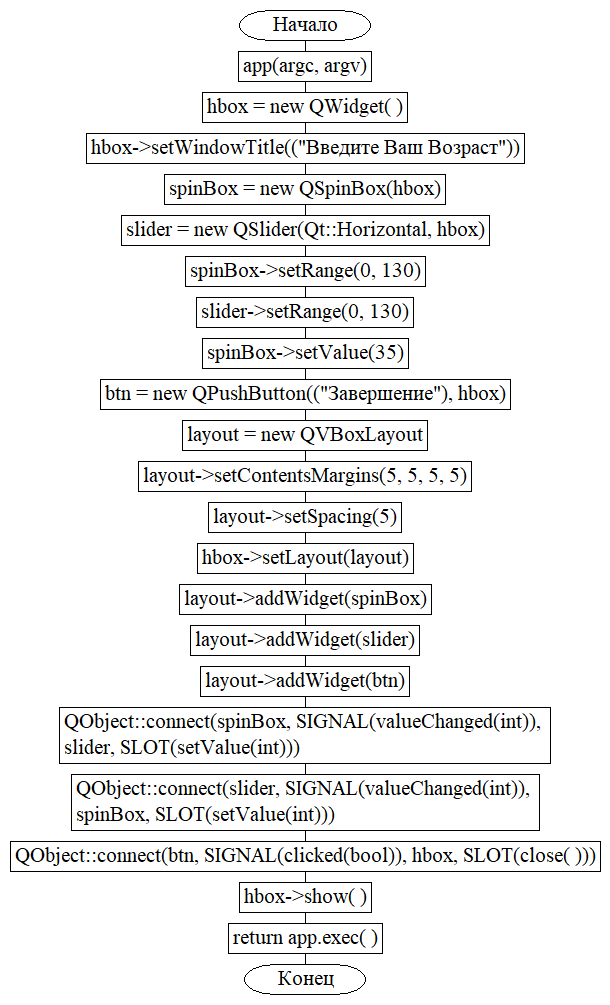
</widget>

<resources/>

<connections/>

</ui>





Задание 2

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

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\_pushButton\_clicked();

private:

Ui::MainWindow \*ui;

};

#endif // MAINWINDOW\_H

#include "mainwindow.h"

#include <QApplication>

#include <QSplitter>

int main(int argc, char \*argv[])

{

QApplication a(argc, argv);

QSplitter \* splitter = new QSplitter(Qt::Vertical);

MainWindow \* dialog1 = new MainWindow();

MainWindow \* dialog2 = new MainWindow();

splitter->addWidget( dialog1 );

splitter->addWidget( dialog2 );

splitter->show();

return a.exec();

}

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <QMessageBox>

MainWindow::MainWindow(QWidget \*parent)

: QMainWindow(parent)

, ui(new Ui::MainWindow)

{

ui->setupUi(this);

}

MainWindow::~MainWindow()

{

delete ui;

}

void MainWindow::on\_pushButton\_clicked()

{

QApplication::quit();

}

<?xml version="1.0" encoding="UTF-8"?>

<ui version="4.0">

<class>MainWindow</class>

<widget class="QMainWindow" name="MainWindow">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>588</width>

<height>395</height>

</rect>

</property>

<property name="windowTitle">

<string>MainWindow</string>

</property>

<widget class="QWidget" name="centralwidget">

<layout class="QVBoxLayout" name="verticalLayout\_3">

<item>

<layout class="QHBoxLayout" name="horizontalLayout\_3">

<item>

<widget class="QTextEdit" name="textEdit"/>

</item>

<item>

<layout class="QVBoxLayout" name="verticalLayout\_2">

<item>

<widget class="QDial" name="dial"/>

</item>

<item>

<widget class="QSpinBox" name="spinBox"/>

</item>

<item>

<spacer name="verticalSpacer">

<property name="orientation">

<enum>Qt::Vertical</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>20</width>

<height>40</height>

</size>

</property>

</spacer>

</item>

</layout>

</item>

</layout>

</item>

<item>

<layout class="QHBoxLayout" name="horizontalLayout\_4">

<item>

<spacer name="horizontalSpacer">

<property name="orientation">

<enum>Qt::Horizontal</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>40</width>

<height>20</height>

</size>

</property>

</spacer>

</item>

<item>

<widget class="QLabel" name="label">

<property name="text">

<string>Нажмите кнопку -&gt;</string>

</property>

</widget>

</item>

<item>

<widget class="QPushButton" name="pushButton">

<property name="text">

<string>Выйти</string>

</property>

</widget>

</item>

</layout>

</item>

</layout>

</widget>

</widget>

<resources/>

<connections>

<connection>

<sender>dial</sender>

<signal>sliderMoved(int)</signal>

<receiver>spinBox</receiver>

<slot>setValue(int)</slot>

<hints>

<hint type="sourcelabel">

<x>540</x>

<y>97</y>

</hint>

<hint type="destinationlabel">

<x>544</x>

<y>164</y>

</hint>

</hints>

</connection>

<connection>

<sender>spinBox</sender>

<signal>valueChanged(int)</signal>

<receiver>dial</receiver>

<slot>setValue(int)</slot>

<hints>

<hint type="sourcelabel">

<x>505</x>

<y>161</y>

</hint>

<hint type="destinationlabel">

<x>505</x>

<y>123</y>

</hint>

</hints>

</connection>

<connection>

<sender>pushButton</sender>

<signal>clicked()</signal>

<receiver>MainWindow</receiver>

<slot>reject()</slot>

<hints>

<hint type="sourcelabel">

<x>524</x>

<y>460</y>

</hint>

<hint type="destinationlabel">

<x>539</x>

<y>335</y>

</hint>

</hints>

</connection>

<connection>

<sender>spinBox</sender>

<signal>textChanged(QString)</signal>

<receiver>textEdit</receiver>

<slot>append(QString)</slot>

<hints>

<hint type="sourcelabel">

<x>487</x>

<y>153</y>

</hint>

<hint type="destinationlabel">

<x>387</x>

<y>203</y>

</hint>

</hints>

</connection>

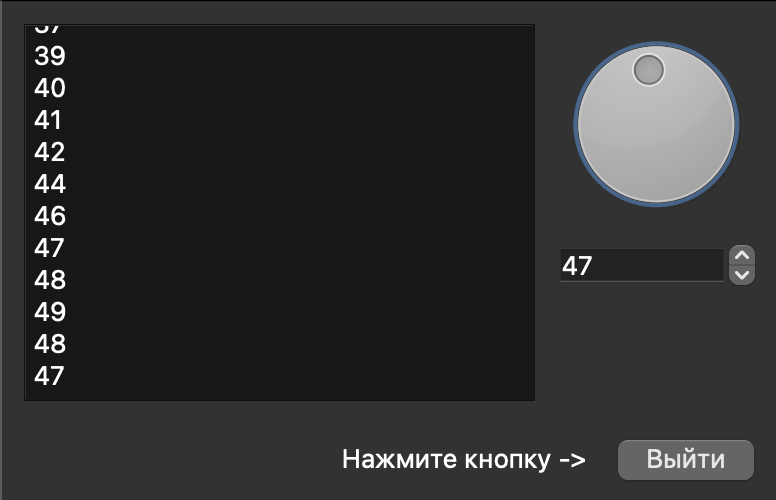
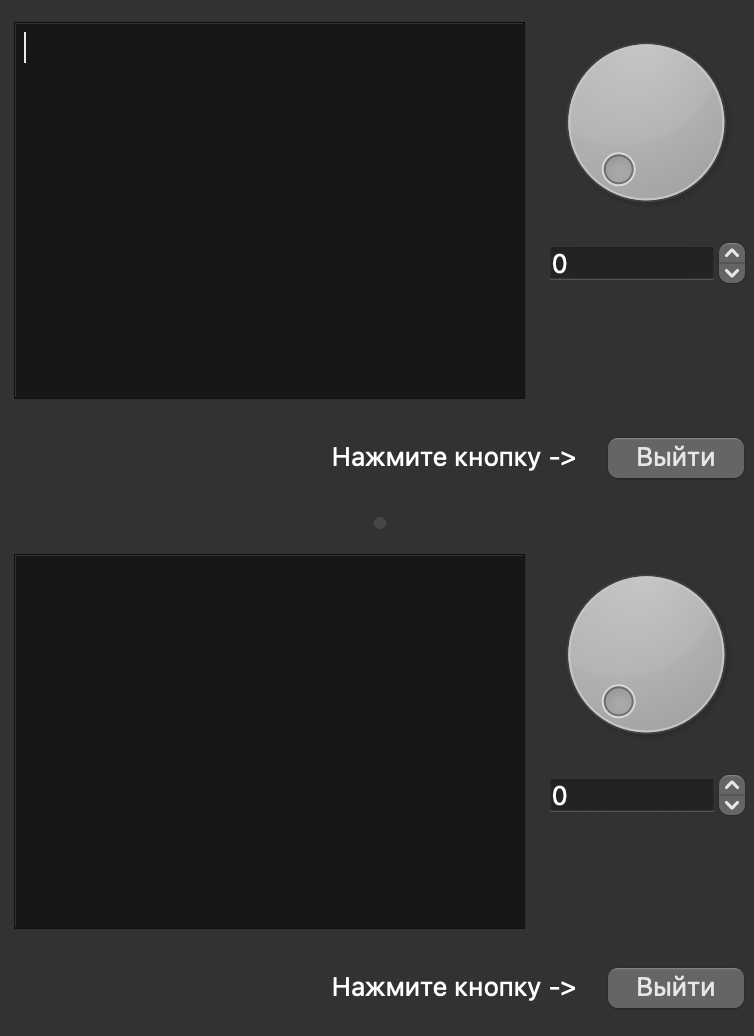
</connections>

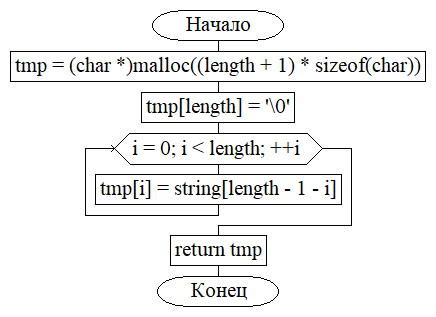
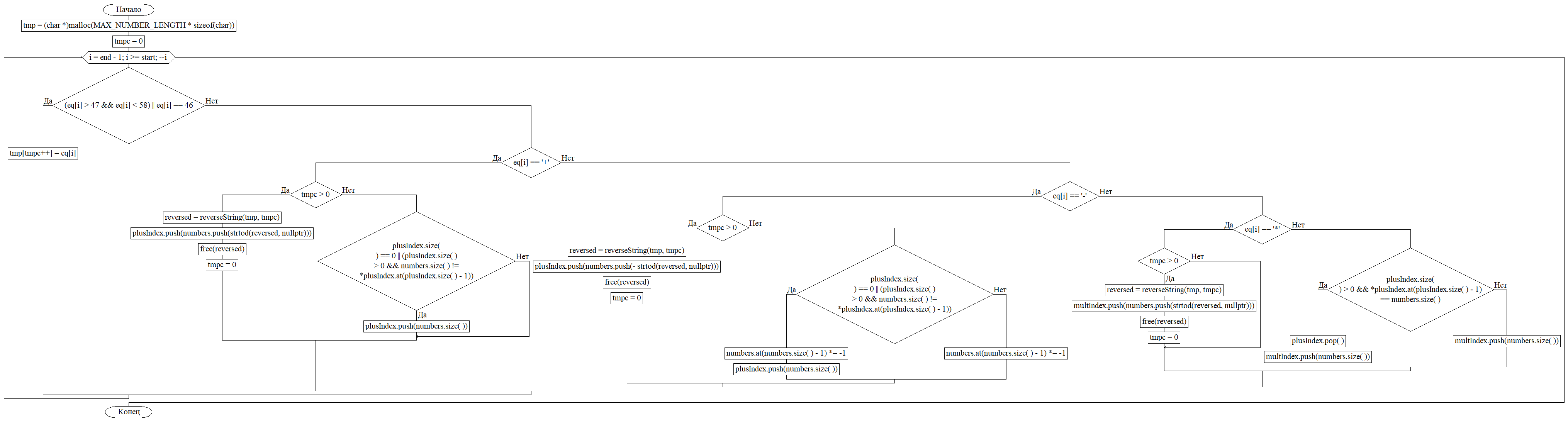
<slots>

<slot>reject()</slot>

</slots>

</ui>





Задание 3

#ifndef CALC\_H

#define CALC\_H

#include <QMainWindow>

#include <QString>

QT\_BEGIN\_NAMESPACE

namespace Ui { class calc; }

QT\_END\_NAMESPACE

class calc : public QMainWindow

{

Q\_OBJECT

public:

calc(QWidget \*parent = nullptr);

~calc();

QString clc;

private slots:

void on\_but0\_clicked();

void on\_but1\_clicked();

void on\_but2\_clicked();

void on\_but3\_clicked();

void on\_but4\_clicked();

void on\_but5\_clicked();

void on\_but6\_clicked();

void on\_but7\_clicked();

void on\_but8\_clicked();

void on\_but9\_clicked();

void on\_butFloat\_clicked();

void on\_butSum\_clicked();

void on\_butMin\_clicked();

void on\_butMultiply\_clicked();

void on\_butDevide\_clicked();

void on\_butDeg\_clicked();

void on\_butLog\_clicked();

void on\_butSin\_clicked();

void on\_butCos\_clicked();

void on\_butPM\_clicked();

void on\_butDel\_clicked();

void on\_butCE\_clicked();

void on\_butC\_clicked();

void on\_butEqually\_clicked();

private:

Ui::calc \*ui;

};

#endif // CALC\_H

#ifndef SOLVE\_SOLVE\_H

#define SOLVE\_SOLVE\_H

#include <math.h>

float solve(const char\* eq,char start,char end,const float\* vars = nullptr);

char\* reverseString(const char\* string,char length);

#endif //SOLVE\_SOLVE\_H

#ifndef SOLVE\_VECTOR\_H

#define SOLVE\_VECTOR\_H

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define DEFAULT\_ARRAY\_SIZE 10

#define DEFAULT\_INCREMENT 10

#define MAX\_NUMBER\_LENGTH 10

template<typename T>

class Vector{

T\* data;

char length;

char currentIndex;

public:

Vector();

~Vector();

char resize(char size=0);

T \*at(int index);

char push(T value); //return size() after adding new element

char pop();

char size(); //return number of elements in data array

T\* getData();

char insert(T value, char index);

char remove(char startIndex, char endIndex); //start and endindex will also be removed! return number of elements after removal

char clear();

};

template <typename T>

char Vector<T>::clear() {

currentIndex=0;

//check if array exists. if not use malloc

if(!data)

{

data = (T\*)malloc(DEFAULT\_ARRAY\_SIZE\*sizeof(T));

}else{

data = (T\*)realloc(data,DEFAULT\_ARRAY\_SIZE\*sizeof(T));

}

if(!data) {

length = 0;

return 0;

}else {

length = DEFAULT\_ARRAY\_SIZE;

return length;

}

}

template <typename T>

char Vector<T>::insert(T value, char index) {

if(currentIndex>=length) {

if (!resize(length + DEFAULT\_INCREMENT))

return 0;

}

if(index!=currentIndex && index >=0 && index <currentIndex)

{

memmove(at(index+1),at(index),sizeof(T)\*(currentIndex-index));

data[index]=value;

}else if(index == currentIndex)

data[index]=value;

return ++currentIndex;

}

template<typename T>

char Vector<T>::remove(char startIndex, char endIndex){

if(startIndex >=0 && startIndex<size() && endIndex>=startIndex && endIndex<size()){

//we do not overwrite values that have index greater than currentIndex! we memcpy all elements after endindex to starrtindex

memmove(at(startIndex),at(endIndex),sizeof(T)\*(currentIndex-1-endIndex));

currentIndex -= (endIndex-startIndex)+1;//adjust number of elements counter

}

return currentIndex;

}

template<typename T>

T\* Vector<T>::getData(){

return data;

}

template<typename T>

Vector<T>::Vector():currentIndex(0){

data = (T\*)malloc(DEFAULT\_ARRAY\_SIZE\*sizeof(T));

if(!data)

length=0;

else

length=DEFAULT\_ARRAY\_SIZE;

}

template<typename T>

Vector<T>::~Vector(){

free(data);

}

template<typename T>

T \* Vector<T>::at(int index) {

if(index<0 || index>= length)

return NULL;

return &data[index];

}

template<typename T>

char Vector<T>::resize(char size) {

//check if array exists. if not use malloc

if(!data)

{

data = (T\*)malloc(DEFAULT\_ARRAY\_SIZE\*sizeof(T));

}else{

data = (T\*)realloc(data,size\*sizeof(T));

}

if(!data) {

length = 0;

return 0;

}else {

length = size;

return size;

}

}

template <typename T>

char Vector<T>::size() {

return currentIndex;

}

template <typename T>

char Vector<T>::push(T value) {

if(currentIndex>=length) {

if (!resize(length + DEFAULT\_INCREMENT))

return 0;

}

data[currentIndex]=value;

return ++currentIndex;

}

template <typename T>

char Vector<T>::pop() {

if(currentIndex>0) {

return --currentIndex;

}

return 0;

}

#endif //SOLVE\_VECTOR\_H

#include "calc.h"

#include "ui\_calc.h"

#include <QTextCursor>

#include "solve.cpp"

calc::calc(QWidget \*parent)

: QMainWindow(parent)

, ui(new Ui::calc)

{

ui->setupUi(this);

}

calc::~calc()

{

delete ui;

}

void calc::on\_but0\_clicked()

{

clc += "0";

ui->textEdit->setText(clc);

for(auto \_:clc) ui->textEdit->moveCursor(QTextCursor::Right);

}

void calc::on\_but1\_clicked()

{

clc += "1";

ui->textEdit->setText(clc);

for(auto \_:clc) ui->textEdit->moveCursor(QTextCursor::Right);

}

void calc::on\_but2\_clicked()

{

clc += "2";

ui->textEdit->setText(clc);

for(auto \_:clc) ui->textEdit->moveCursor(QTextCursor::Right);

}

void calc::on\_but3\_clicked()

{

clc += "3";

ui->textEdit->setText(clc);

for(auto \_:clc) ui->textEdit->moveCursor(QTextCursor::Right);

}

void calc::on\_but4\_clicked()

{

clc += "4";

ui->textEdit->setText(clc);

for(auto \_:clc) ui->textEdit->moveCursor(QTextCursor::Right);

}

void calc::on\_but5\_clicked()

{

clc += "5";

ui->textEdit->setText(clc);

for(auto \_:clc) ui->textEdit->moveCursor(QTextCursor::Right);

}

void calc::on\_but6\_clicked()

{

clc += "6";

ui->textEdit->setText(clc);

for(auto \_:clc) ui->textEdit->moveCursor(QTextCursor::Right);

}

void calc::on\_but7\_clicked()

{

clc += "7";

ui->textEdit->setText(clc);

for(auto \_:clc) ui->textEdit->moveCursor(QTextCursor::Right);

}

void calc::on\_but8\_clicked()

{

clc += "8";

ui->textEdit->setText(clc);

for(auto \_:clc) ui->textEdit->moveCursor(QTextCursor::Right);

}

void calc::on\_but9\_clicked()

{

clc += "9";

ui->textEdit->setText(clc);

for(auto \_:clc) ui->textEdit->moveCursor(QTextCursor::Right);

}

void calc::on\_butFloat\_clicked()

{

clc += ".";

ui->textEdit->setText(clc);

for(auto \_:clc) ui->textEdit->moveCursor(QTextCursor::Right);

}

void calc::on\_butSum\_clicked()

{

clc += "+";

ui->textEdit->setText(clc);

for(auto \_:clc) ui->textEdit->moveCursor(QTextCursor::Right);

}

void calc::on\_butMin\_clicked()

{

clc += "-";

ui->textEdit->setText(clc);

for(auto \_:clc) ui->textEdit->moveCursor(QTextCursor::Right);

}

void calc::on\_butMultiply\_clicked()

{

clc += "\*";

ui->textEdit->setText(clc);

for(auto \_:clc) ui->textEdit->moveCursor(QTextCursor::Right);

}

void calc::on\_butDevide\_clicked()

{

clc += "/";

ui->textEdit->setText(clc);

for(auto \_:clc) ui->textEdit->moveCursor(QTextCursor::Right);

}

void calc::on\_butDeg\_clicked()

{

clc += "^";

ui->textEdit->setText(clc);

for(auto \_:clc) ui->textEdit->moveCursor(QTextCursor::Right);

}

void calc::on\_butLog\_clicked()

{

clc = "log"+clc+"(";

ui->textEdit->setText(clc);

for(auto \_:clc) ui->textEdit->moveCursor(QTextCursor::Right);

}

void calc::on\_butSin\_clicked()

{

clc = "sin("+clc+")";

ui->textEdit->setText(clc);

for(auto \_:clc) ui->textEdit->moveCursor(QTextCursor::Right);

}

void calc::on\_butCos\_clicked()

{

clc = "cos("+clc+")";

ui->textEdit->setText(clc);

for(auto \_:clc) ui->textEdit->moveCursor(QTextCursor::Right);

}

void calc::on\_butPM\_clicked()

{

if (clc.length() > 0){

if (clc[0] == '-')

clc = clc.mid(1, clc.length());

else clc = "-"+clc;

} else clc = "-";

ui->textEdit->setText(clc);

for(auto \_:clc) ui->textEdit->moveCursor(QTextCursor::Right);

}

void calc::on\_butDel\_clicked()

{

if (clc.length() > 0){

clc = clc.mid(0, clc.length()-1);

ui->textEdit->setText(clc);

for(auto \_:clc) ui->textEdit->moveCursor(QTextCursor::Right);

}

}

void calc::on\_butCE\_clicked()

{

clc = "";

ui->textEdit->setText(clc);

}

void calc::on\_butC\_clicked()

{

on\_butCE\_clicked();

}

void calc::on\_butEqually\_clicked()

{

std::string sv = clc.toStdString();

float b = solve(&sv[0],0,clc.length());

clc = QString::number(b);;

ui->textEdit->setText(clc);

for(auto \_:clc) ui->textEdit->moveCursor(QTextCursor::Right);

}

#include "solve.h"

#include "vector.h"

#include <math.h>

char \*reverseString(const char \*string, char length)

{

auto tmp = (char \*)malloc((length + 1) \* sizeof(char));

if (!tmp)

return nullptr;

tmp[length] = '\0';

for (int i = 0; i < length; ++i)

{

tmp[i] = string[length - 1 - i];

}

return tmp; // make sure to free the returned pointer

}

float solve(const char \*eq, char start, char end, const float \*vars)

{

// check if eq is valid

// if not, return NaN

if (end <= start)

return NAN;

// create new arrays to store numbers and operators

Vector<float> numbers;

Vector<char> plusIndex;

Vector<char> multIndex;

Vector<char> powIndex;

auto tmp = (char \*)malloc(MAX\_NUMBER\_LENGTH \* sizeof(char));

if (!tmp)

return NAN;

char tmpc = 0;

char \*reversed;

// parse string eq

// parsing from right to left allows us to

// e.g. multiply the number to the right of binary operators with -1 for subtraction

for (char i = end - 1; i >= start; --i)

{

// check if current char is number or decimalpoint '.' using ascii

if ((eq[i] > 47 && eq[i] < 58) || eq[i] == 46)

{

if (tmpc >= MAX\_NUMBER\_LENGTH)

return NAN;

tmp[tmpc++] = eq[i];

}

// if current char is + or - add tmp to numbers array if it is not empty

// if it is empty: ignore if current char is '+' and mult the last number in numbers with -1 if it is '-'.

//->do not add to plusIndex array

else if (eq[i] == '+')

{

if (tmpc > 0)

{

reversed = reverseString(tmp, tmpc);

if (!reversed)

return NAN;

plusIndex.push(numbers.push(strtod(reversed, nullptr)));

free(reversed);

tmpc = 0;

}

// handling wrong or weird inputs

else if (i == end - 1)

{

return NAN;

}

// these two extra cases are necessary because a calculation like a++--+b is valid and equal to a+b

else if (plusIndex.size() == 0 || (plusIndex.size() > 0 && numbers.size() != \*plusIndex.at(

plusIndex.size() - 1)))

{

plusIndex.push(numbers.size());

}

}

else if (eq[i] == '-')

{

if (tmpc > 0)

{

reversed = reverseString(tmp, tmpc);

if (!reversed)

return NAN;

plusIndex.push(numbers.push(-strtod(reversed, nullptr)));

free(reversed);

tmpc = 0;

}

// handling wrong or weird inputs

else if (i == end - 1)

{

return NAN;

}

// these two extra cases are necessary because a calculation like a++--+b is valid and equal to a+b

else if (plusIndex.size() == 0 || (plusIndex.size() > 0 && numbers.size() != \*plusIndex.at(

plusIndex.size() - 1)))

{

\*numbers.at(numbers.size() - 1) \*= -1;

plusIndex.push(numbers.size());

}

else

{

\*numbers.at(numbers.size() - 1) \*= -1;

}

}

// check for multiplication

else if (eq[i] == '\*')

{

if (tmpc > 0)

{

reversed = reverseString(tmp, tmpc);

if (!reversed)

return NAN;

multIndex.push(numbers.push(strtod(reversed, nullptr)));

free(reversed);

tmpc = 0;

}

else if (i == end - 1 || i == start)

{

return NAN;

}

// this case is for a\*-b. because - is pushed into the plusIndex array we need to remove it.

else if (plusIndex.size() > 0 && \*plusIndex.at(plusIndex.size() - 1) == numbers.size())

{

plusIndex.pop();

multIndex.push(numbers.size());

}

else

{

multIndex.push(numbers.size());

}

}

else if (eq[i] == '/')

{

if (tmpc > 0)

{

reversed = reverseString(tmp, tmpc);

if (!reversed)

return NAN;

multIndex.push(numbers.push((float)1 / strtod(reversed, nullptr)));

free(reversed);

tmpc = 0;

}

else if (i == end - 1 || i == start)

{

return NAN;

}

// this case is for a/-b. because - is pushed into the plusIndex array we need to remove it.

else if (plusIndex.size() > 0 && \*plusIndex.at(plusIndex.size() - 1) == numbers.size())

{

plusIndex.pop();

(\*numbers.at(numbers.size() - 1)) = 1 / (\*numbers.at(numbers.size() - 1));

multIndex.push(numbers.size());

}

else

{

(\*numbers.at(numbers.size() - 1)) = 1 / (\*numbers.at(numbers.size() - 1));

multIndex.push(numbers.size());

}

}

else if (eq[i] == '^')

{

if (tmpc > 0)

{

reversed = reverseString(tmp, tmpc);

if (!reversed)

return NAN;

multIndex.push(numbers.push(strtod(reversed, nullptr)));

powIndex.push(numbers.size());

free(reversed);

tmpc = 0;

}

else if (i == end - 1 || i == start)

{

return NAN;

} // this case is for a/-b. because - is pushed into the plusIndex array we need to remove it.

else if (plusIndex.size() > 0 && \*plusIndex.at(plusIndex.size() - 1) == numbers.size())

{

plusIndex.pop();

multIndex.push(numbers.size());

powIndex.push(numbers.size());

}

else

{

multIndex.push(numbers.size());

powIndex.push(numbers.size());

}

}

// if we find a closing bracket, try to find a matching opening bracket and call solve recursively

else if (eq[i] == ')')

{

// try to find a matching '(':

char numClosingBrackets = 0;

char foundMatching = 0;

for (char j = i - 1; j >= start; --j)

{

if (eq[j] == ')')

++numClosingBrackets;

else if (eq[j] == '(' && numClosingBrackets > 0)

--numClosingBrackets;

else if (eq[j] == '(' && numClosingBrackets == 0)

{

// matching '(' found

if (!foundMatching)

{

numbers.push(solve(eq, j + 1, i, vars));

i = j; // skip the part between () in parsing

foundMatching = 1;

}

}

}

if (!foundMatching)

return NAN;

}

else

{

// unary operators:

// trig functions work with rad not deg!

if (i > 2 && eq[i] == 'n' && eq[i - 1] == 'i' && eq[i - 2] == 's' && eq[i - 3] == 'a')

{

if (numbers.size())

\*numbers.at(numbers.size() - 1) = asin(\*numbers.at(numbers.size() - 1));

i -= 3;

if (plusIndex.size() > 0 && \*plusIndex.at(plusIndex.size() - 1) == numbers.size())

{

plusIndex.pop();

}

}

else if (i > 1 && eq[i] == 'n' && eq[i - 1] == 'i' && eq[i - 2] == 's')

{

if (numbers.size())

\*numbers.at(numbers.size() - 1) = sin(\*numbers.at(numbers.size() - 1));

i -= 2;

if (plusIndex.size() > 0 && \*plusIndex.at(plusIndex.size() - 1) == numbers.size())

{

plusIndex.pop();

}

}

else if (i > 2 && eq[i] == 's' && eq[i - 1] == 'o' && eq[i - 2] == 'c' && eq[i - 3] == 'a')

{

if (numbers.size())

\*numbers.at(numbers.size() - 1) = acos(\*numbers.at(numbers.size() - 1));

i -= 3;

if (plusIndex.size() > 0 && \*plusIndex.at(plusIndex.size() - 1) == numbers.size())

{

plusIndex.pop();

}

}

else if (i > 1 && eq[i] == 's' && eq[i - 1] == 'o' && eq[i - 2] == 'c')

{

if (numbers.size())

\*numbers.at(numbers.size() - 1) = cos(\*numbers.at(numbers.size() - 1));

i -= 2;

if (plusIndex.size() > 0 && \*plusIndex.at(plusIndex.size() - 1) == numbers.size())

{

plusIndex.pop();

}

}

else if (i > 2 && eq[i] == 'n' && eq[i - 1] == 'a' && eq[i - 2] == 't' && eq[i - 3] == 'a')

{

if (numbers.size())

\*numbers.at(numbers.size() - 1) = atan(\*numbers.at(numbers.size() - 1));

i -= 3;

if (plusIndex.size() > 0 && \*plusIndex.at(plusIndex.size() - 1) == numbers.size())

{

plusIndex.pop();

}

}

else if (i > 1 && eq[i] == 'n' && eq[i - 1] == 'a' && eq[i - 2] == 't')

{

if (numbers.size())

\*numbers.at(numbers.size() - 1) = tan(\*numbers.at(numbers.size() - 1));

i -= 2;

if (plusIndex.size() > 0 && \*plusIndex.at(plusIndex.size() - 1) == numbers.size())

{

plusIndex.pop();

}

}

else if (i > 0 && eq[i] == 'n' && eq[i - 1] == 'l')

{

if (numbers.size())

\*numbers.at(numbers.size() - 1) = log(\*numbers.at(numbers.size() - 1));

i -= 3;

if (plusIndex.size() > 0 && \*plusIndex.at(plusIndex.size() - 1) == numbers.size())

{

plusIndex.pop();

}

}

else if (i > 1 && eq[i] == 'g' && eq[i - 1] == 'o' && eq[i - 2] == 'l')

{

if (numbers.size())

\*numbers.at(numbers.size() - 1) = log10(\*numbers.at(numbers.size() - 1));

i -= 2;

if (plusIndex.size() > 0 && \*plusIndex.at(plusIndex.size() - 1) == numbers.size())

{

plusIndex.pop();

}

}

// constants

else if (i > 0 && eq[i] == 'i' && eq[i - 1] == 'p')

{

numbers.push(M\_PI);

i -= 1;

}

else if (eq[i] == 'e')

{

numbers.push(M\_E);

}

else if (i > 1 && eq[i] == 's' && eq[i - 1] == 'n' && eq[i - 2] == 'a')

{

if (vars)

numbers.push(vars[0]);

else

numbers.push(NAN);

i -= 2;

}

else

return NAN;

}

}

// push to numbers array one last time

if (tmpc > 0)

{

reversed = reverseString(tmp, tmpc);

if (!reversed)

return NAN;

numbers.push(strtod(reversed, nullptr));

free(reversed);

tmpc = 0;

}

// parsing eq is done.

// start computing result:

// rules:

// brackets first (already done in parsing)

// mult and div before add and sub

// equal priority operations from left to right

// we do not need to worry about the last point as we are replacing all divisions (a/b) with a\*(1/b)

// and all subtractions (a+b) with (a+(-b))

//+ and \* are commutative.

if (numbers.size() == 0)

return NAN;

if (powIndex.size() > 0)

{

for (char i = powIndex.size() - 1; i >= 0; --i)

{

// check if '\*' is associated with two numbers:

if (\*powIndex.at(i) >= numbers.size())

return NAN;

(\*numbers.at(\*powIndex.at(i) - 1)) = pow((\*numbers.at(\*powIndex.at(i))), (\*numbers.at(\*powIndex.at(i) - 1)));

(\*numbers.at(\*powIndex.at(i))) = 1;

}

}

if (multIndex.size() > 0)

{

for (char i = multIndex.size() - 1; i >= 0; --i)

{

// check if '\*' is associated with two numbers:

if (\*multIndex.at(i) >= numbers.size())

return NAN;

(\*numbers.at(\*multIndex.at(i) - 1)) \*= (\*numbers.at(\*multIndex.at(i)));

}

}

float result = \*numbers.at(0);

if (numbers.size() > 1)

{ // if numbers.size == 1 we have a leading +-. we ignore it.

for (char i = 0; i < plusIndex.size(); ++i)

{

result += \*numbers.at(\*plusIndex.at(i));

}

}

free(tmp);

return result;

}

<?xml version="1.0" encoding="UTF-8"?>

<ui version="4.0">

<class>calc</class>

<widget class="QMainWindow" name="calc">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>374</width>

<height>296</height>

</rect>

</property>

<property name="windowTitle">

<string>calc</string>

</property>

<widget class="QWidget" name="centralwidget">

<layout class="QVBoxLayout" name="verticalLayout">

<item>

<widget class="QTextEdit" name="textEdit">

<property name="font">

<font>

<pointsize>20</pointsize>

</font>

</property>

<property name="html">

<string>&lt;!DOCTYPE HTML PUBLIC &quot;-//W3C//DTD HTML 4.0//EN&quot; &quot;http://www.w3.org/TR/REC-html40/strict.dtd&quot;&gt;

&lt;html&gt;&lt;head&gt;&lt;meta name=&quot;qrichtext&quot; content=&quot;1&quot; /&gt;&lt;meta charset=&quot;utf-8&quot; /&gt;&lt;style type=&quot;text/css&quot;&gt;

p, li { white-space: pre-wrap; }

hr { height: 1px; border-width: 0; }

li.unchecked::marker { content: &quot;\2610&quot;; }

li.checked::marker { content: &quot;\2612&quot;; }

&lt;/style&gt;&lt;/head&gt;&lt;body style=&quot; font-family:'.AppleSystemUIFont'; font-size:20pt; font-weight:400; font-style:normal;&quot;&gt;

&lt;p style=&quot;-qt-paragraph-type:empty; margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;&quot;&gt;&lt;br /&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>

</property>

</widget>

</item>

<item>

<layout class="QHBoxLayout" name="horizontalLayout">

<item>

<widget class="QPushButton" name="butDel">

<property name="text">

<string>&lt;-</string>

</property>

</widget>

</item>

<item>

<widget class="QPushButton" name="butCE">

<property name="text">

<string>CE</string>

</property>

</widget>

</item>

<item>

<widget class="QPushButton" name="butC">

<property name="text">

<string>C</string>

</property>

</widget>

</item>

</layout>

</item>

<item>

<layout class="QHBoxLayout" name="horizontalLayout\_2">

<item>

<layout class="QGridLayout" name="gridLayout">

<item row="0" column="0">

<widget class="QPushButton" name="but7">

<property name="text">

<string>7</string>

</property>

</widget>

</item>

<item row="0" column="1">

<widget class="QPushButton" name="but8">

<property name="text">

<string>8</string>

</property>

</widget>

</item>

<item row="0" column="2">

<widget class="QPushButton" name="but9">

<property name="text">

<string>9</string>

</property>

</widget>

</item>

<item row="0" column="3">

<widget class="QPushButton" name="butDevide">

<property name="text">

<string>/</string>

</property>

</widget>

</item>

<item row="0" column="4">

<widget class="QPushButton" name="butDeg">

<property name="text">

<string>x^y</string>

</property>

</widget>

</item>

<item row="1" column="0">

<widget class="QPushButton" name="but4">

<property name="text">

<string>4</string>

</property>

</widget>

</item>

<item row="1" column="1">

<widget class="QPushButton" name="but5">

<property name="text">

<string>5</string>

</property>

</widget>

</item>

<item row="1" column="2">

<widget class="QPushButton" name="but6">

<property name="text">

<string>6</string>

</property>

</widget>

</item>

<item row="1" column="3">

<widget class="QPushButton" name="butMultiply">

<property name="text">

<string>\*</string>

</property>

</widget>

</item>

<item row="1" column="4">

<widget class="QPushButton" name="butLog">

<property name="text">

<string>logx(y)</string>

</property>

</widget>

</item>

<item row="2" column="0">

<widget class="QPushButton" name="but1">

<property name="text">

<string>1</string>

</property>

</widget>

</item>

<item row="2" column="1">

<widget class="QPushButton" name="but2">

<property name="text">

<string>2</string>

</property>

</widget>

</item>

<item row="2" column="2">

<widget class="QPushButton" name="but3">

<property name="text">

<string>3</string>

</property>

</widget>

</item>

<item row="2" column="3">

<widget class="QPushButton" name="butMin">

<property name="text">

<string>-</string>

</property>

</widget>

</item>

<item row="2" column="4">

<widget class="QPushButton" name="butSin">

<property name="text">

<string>sinx</string>

</property>

</widget>

</item>

<item row="3" column="0">

<widget class="QPushButton" name="but0">

<property name="text">

<string>0</string>

</property>

</widget>

</item>

<item row="3" column="1">

<widget class="QPushButton" name="butPM">

<property name="text">

<string>-/+</string>

</property>

</widget>

</item>

<item row="3" column="2">

<widget class="QPushButton" name="butFloat">

<property name="text">

<string>.</string>

</property>

</widget>

</item>

<item row="3" column="3">

<widget class="QPushButton" name="butSum">

<property name="text">

<string>+</string>

</property>

</widget>

</item>

<item row="3" column="4">

<widget class="QPushButton" name="butCos">

<property name="text">

<string>cosx</string>

</property>

</widget>

</item>

</layout>

</item>

<item>

<widget class="QPushButton" name="butEqually">

<property name="text">

<string>=</string>

</property>

</widget>

</item>

</layout>

</item>

</layout>

</widget>

</widget>

<resources/>

<connections/>

</ui>

Задание 4

#include <QtWidgets/QApplication>

#include <QtWidgets/QTextEdit>

#include <QtWidgets/QPushButton>

#include <QtWidgets/QVBoxLayout>

#include <QtWidgets/QWidget>

#include <QtCore/QObject>

class Widget : public QWidget {

public:

Widget(QWidget \*parent = nullptr) : QWidget(parent) {

m\_textEdit = new QTextEdit();

m\_textEdit2 = new QTextEdit();

m\_button = new QPushButton("Convert");

QVBoxLayout \*layout = new QVBoxLayout();

layout->addWidget(m\_textEdit);

layout->addWidget(m\_button);

layout->addWidget(m\_textEdit2);

setLayout(layout);

connect(m\_button, &QPushButton::clicked, this, &Widget::onButtonClicked);

}

public slots:

void onButtonClicked() {

QString text = m\_textEdit->toPlainText();

QString textLower = "\nall lower: " + text.toLower();

QString textUpper = "\nALL UPPER: " + text.toUpper();

m\_textEdit2->setPlainText("input: " + text+textLower+textUpper);

}

private:

QTextEdit \*m\_textEdit;

QTextEdit \*m\_textEdit2;

QPushButton \*m\_button;

};

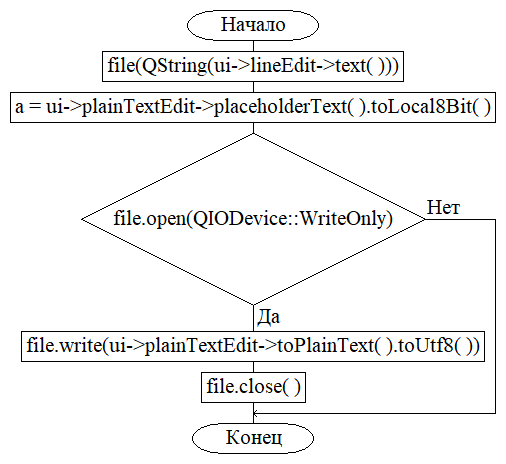
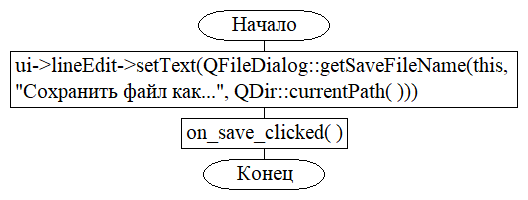
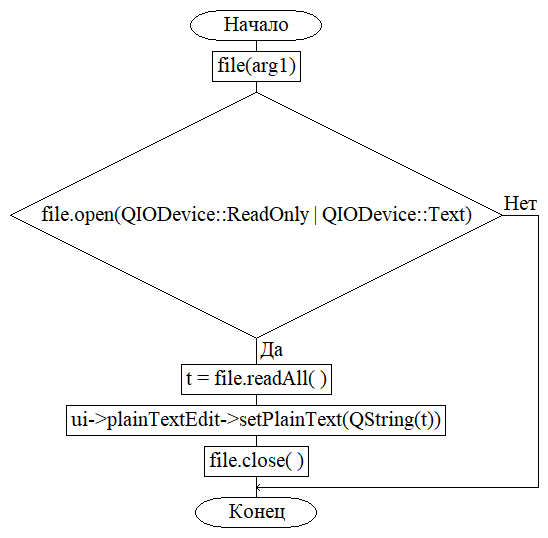
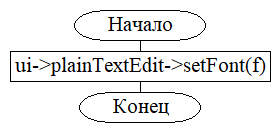
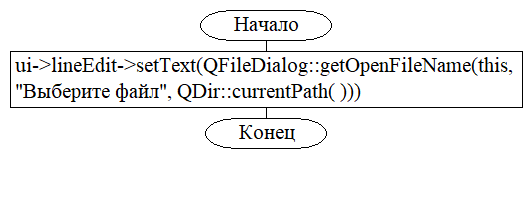
int main(int argc, char \*argv[]) {

QApplication app(argc, argv);

Widget \*window = new Widget();

window->show();

return app.exec();

}

Вывод: я научился работать в среде Qt.