Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

Пермский национальный исследовательский политехнический университет

Электротехнический факультет

Кафедра информационных технологий и автоматизированных систем

**ОТЧЁТ**

**по дисциплине «Информатика»**

Семестр: 2

На темы:

«АРМ машиниста паровоза»,

«Решение задачи коммивояжера»

Выполнил студент ИВТ-22-2б:

Зайцев Павел Денисович

Пермь 2023

[Введение 3](#_Toc8486)

[Реализация калькулятора временных величин 4](#_Toc18761)

[Полный код программы 9](#_Toc13416)

[Диаграмма классов 19](#_Toc5347)

[Результат работы 20](#_Toc26782)

[Решение задачи коммивояжёра 21](#_Toc15833)

[Полный код программы 24](#_Toc32014)

[Диаграмма классов 32](#_Toc12849)

[Результат работы 33](#_Toc27701)

[Вывод 35](#_Toc23758)

[Используемое программное обеспечение 36](#_Toc28226)

# Введение

Автоматизированное рабочее место - индивидуальный комплекс технических и программных средств, предназначенный для автоматизации профессионального труда специалиста и обеспечивающий подготовку, редактирование, поиск и выдачу на экран и печать необходимых ему документов и данных. Автоматизированное рабочее место обеспечивает оператора всеми средствами, необходимыми для выполнения определённых функций.

Задача [коммивояжёра](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BC%D0%B8%D0%B2%D0%BE%D1%8F%D0%B6%D1%91%D1%80" \o "Коммивояжёр) (или TSP от [англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA" \o "Английский язык) *travelling salesman problem*) — одна из самых известных задач [комбинаторной оптимизации](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%B1%D0%B8%D0%BD%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F_%D0%BE%D0%BF%D1%82%D0%B8%D0%BC%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F" \o "Комбинаторная оптимизация), заключающаяся в поиске самого выгодного [маршрута](https://ru.wikipedia.org/w/index.php?title=%D0%9C%D0%B0%D1%80%D1%88%D1%80%D1%83%D1%82_(%D1%82%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D0%B3%D1%80%D0%B0%D1%84%D0%BE%D0%B2)&action=edit&redlink=1" \o "Маршрут (теория графов) (страница отсутствует)), проходящего через указанные городам (вершинам графа) хотя бы по одному разу с последующим возвратом в исходный город (вершину графа).

# **Реализация калькулятора временных величин**

Функциональный набор профессиональных навыков, которые должен реализовывать АРМ машиниста паровоза: управление температурой в системе, регулирование давления в системе, наполнение котла водой, управление скоростью движения паровоза, следование расписанию прибытия со станции A на станцию B.

1. Действия, производимые при нажатии на определённую кнопку, реализованы с помощью слотов – методов класса, обычно вызываемых при нажатии на определённую кнопку. Связать определённую кнопку можно двумя способами. Первый способ - настроить связь между выбранной кнопкой и вызываемым слотом с помощью функции connect, указав кнопку, событие, вызывающее слот и сам слот:  
   connect(ui->b0,SIGNAL(clicked()),this,SLOT(number()));  
   Второй способ – создать слот для каждой кнопки отдельно:  
   void Mainwindow::on\_startButtom\_clicked(){}
2. Дизайн создан с применением таблиц стилей.

ui->progressBar->setStyleSheet("\*{color: rgb(255, 255, 0); background-color: qlineargradient(x1: 0, y1: 0, x2: 0, y2: 1, stop: 0 rgba(100,0,0,255), stop: 1 rgba(200,0,0,255)); border: 2px solid grey; border-radius: 5px;}");

1. Для работы программы в реальном времени все арифметические операции и обрисовка проэкта происходит в слоте TimerSLOT класса <QTimer>.

void secondwindow::**TimerSLOT**(){

sec++;

if(sec>=60){

min++;

sec-=60;

}

if (min>=60){

hour++;

min-=60;

}

if(hour==24&&min==60&& sec==60){

sec=0;

min=0;

hour=0;

}

//Отображение времени

ui->timeEdit->setTime(QTime(hour,min,sec,0));

//Заполнение бака c водой

if (peredach==0){

if(bak\_woda<water){

if (water<600){

bak\_woda+=10;

}

else {

bak\_woda+=25;

}

}

else{

bak\_woda=water;

}

ui->progressBar->setValue((bak\_woda\*1000)/water);

}

//Заполнение котла с паром

kof\_water=temperature/100;

if (bak\_woda>0){

if(bak\_pressure<pressure){

bak\_pressure+=kof\_water;

bak\_woda-=kof\_water;

}

else{

bak\_pressure=pressure;

}

}

if (bak\_pressure==pressure){

bak\_pressure-=10;

QMessageBox::warning(this, "Осторожно","Давление слишком большое");

temperature=0;

ui->tempview->display(temperature);

}

//Отображение давления

ui->progressBar\_2->setValue((bak\_pressure\*1000)/pressure);

ui->progressBar->setValue((bak\_woda\*1000)/water);

//Работа с передачами и паром

switch (peredach) {

case 1:

if (bak\_pressure>=2)

{

bak\_pressure-=2; bak\_road+=5;

}

break;

case 2:

if (bak\_pressure>=6){

bak\_pressure-=6; bak\_road+=15;

}

break;

case 3: if(bak\_pressure>=10){

bak\_pressure-=10; bak\_road+=20;

}

break;

case 4:

if(bak\_pressure>=14) {

bak\_pressure-=14; bak\_road+=30;

}

break;

}

ui->progressBar->setValue((bak\_woda\*1000)/water);

ui->progressBar\_3->setValue((bak\_road\*100000)/road);

ui->marker->setGeometry(((bak\_road\*1000)/road),20,25,30);

ui->label\_road->setText(QString::number(bak\_road)+' '+':');

if (bak\_road>=road){

QMessageBox::information(this,"Конец","Вы доехали в В от точки А");

timer->stop();

close();

}

}

1. Класс АРМа(secondwindow) может только при работе с основным классом (**MainWindow**), так как окно с АРМом вызывается по указателю, который является полем основного класса.

class **MainWindow** : public QMainWindow

{

private:

Ui::MainWindow \*ui;

secondwindow \*vv;

bool flag;

};

## Полный код программы

АРМ машиниста паровоза реализован с использованием нескольких файлов:

1. Код заголовочного файла, содержащего описание класса MainWindow:

ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include "secondwindow.h"

#include <QWidget>

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\_2\_clicked**();

void **on\_pushButton\_exit\_clicked**();

private:

Ui::MainWindow \*ui;

secondwindow \*vv;

bool flag;

};

#endif // MAINWINDOW\_H

1. Код файла описывающего методы класса MainWindow:

#include "mainwindow.h"

#include "ui\_mainwindow.h"

MainWindow::**MainWindow**(QWidget \*parent)

: QMainWindow(*parent*)

, ui(new Ui::MainWindow)

{

ui->setupUi(this);

flag=0;

}

MainWindow::~***MainWindow***()

{

delete ui;

}

void MainWindow::**on\_pushButton\_2\_clicked**()

{

if (flag==0){

vv=new secondwindow(this);

vv->show();

flag=1;

}

else{

flag=0;

vv->close();

}

}

void MainWindow::**on\_pushButton\_exit\_clicked**()

{

close();

}

1. Код заголовочного файла, содержащего описание класса secondwindow:

#ifndef SECONDWINDOW\_H

#define SECONDWINDOW\_H

#include <QMainWindow>

#include <QLCDNumber>

#include <QLabel>

#include <QTimer>

#include <QTime>

namespace **Ui** {

class **secondwindow**;

}

class **secondwindow** : public QMainWindow

{

Q\_OBJECT

public:

explicit **secondwindow**(QWidget \*parent = nullptr);

~***secondwindow***();

private slots:

void **on\_horizontalSlider\_valueChanged**(int value);

void **on\_toolButton\_1\_clicked**();

void **on\_toolButton\_2\_clicked**();

void **on\_randomButton\_clicked**();

void **TimerSLOT**();

void **on\_startButton\_clicked**();

void **on\_endButton\_clicked**();

void **on\_spinBox\_water\_valueChanged**(int arg1);

void **on\_spinBox\_pressure\_valueChanged**(int arg1);

void **on\_spinBox\_road\_valueChanged**(int arg1);

void **on\_spinBox\_temp\_valueChanged**(int arg1);

void **on\_timeEdit\_userTimeChanged**(const QTime &time);

private:

Ui::secondwindow \*ui;

int temperature;

int peredach;

int sec;

int min;

int hour;

QTimer \*timer;

int road;

int pressure;

int water;

int kof\_water;

int bak\_woda;

int bak\_pressure;

int bak\_road;

};

#endif // SECONDWINDOW\_H

1. Код файла описывающего методы класса secondwindow:

#include "secondwindow.h"

#include "ui\_secondwindow.h"

#include <QTimer>

#include <QMessageBox>

secondwindow::**secondwindow**(QWidget \*parent) :

QMainWindow(*parent*),

ui(new Ui::secondwindow)

{

ui->setupUi(this);

temperature=0;

peredach=0;

//Временные переменные и таймер

sec=0;

min=0;

//Переменные для баки

hour=0;

road=0;

water=0;

bak\_woda=0;

bak\_road=0;

kof\_water=0;

pressure=0;

bak\_pressure=0;

//Подключение таймера

timer=new QTimer(this);

connect(timer,SIGNAL(timeout()),this,SLOT( TimerSLOT()));

//ПЕРЕМЕЩЕНИЕ МАРКЕРА

ui->marker->setGeometry(0,20,25,30);

//настройка цветов progressbar-ов

ui->progressBar->setOrientation(Qt::Vertical);

ui->progressBar->setStyleSheet("\*{background-color: blue; border: 1px solid black; }");

ui->progressBar->setStyleSheet("\*{color: rgb(255, 255, 0); background-color: qlineargradient(x1: 0, y1: 0, x2: 0, y2: 1, stop: 0 rgba(100,0,0,255), stop: 1 rgba(200,0,0,255)); border: 2px solid grey; border-radius: 5px;}");

ui->progressBar->setStyleSheet("\*::chunk {background-color: qlineargradient(x1: 0, y1: 0, x2: 0, y2: 1, stop: 0 rgba(0,200,255,255), stop: 1 rgba(0,0,255,255));}");

ui->progressBar\_2->setOrientation(Qt::Vertical);

ui->progressBar\_2->setStyleSheet("\*{background-color: red; border: 1px solid black; }");

ui->progressBar\_2->setStyleSheet("\*{color: rgb(255, 0, 0); background-color: qlineargradient(x1: 0, y1: 0, x2: 0, y2: 1, stop: 0 rgba(194, 14, 14, 1), stop: 1 rgba(194, 14, 14, 1)); border: 2px solid grey; border-radius: 5px;}");

ui->progressBar\_2->setStyleSheet("\*::chunk {background-color: qlineargradient(x1: 0, y1: 0, x2: 0, y2: 1, stop: 0 rgba(194, 14, 14, 1), stop: 1 rgba(0,0,150,150));}");

ui->progressBar\_3->setStyleSheet("\*{background-color: black; border: 1px solid black; }");

ui->progressBar\_3->setStyleSheet("\*{color: rgb(148, 135, 135); background-color: qlineargradient(x1: 0, y1: 0, x2: 0, y2: 1, stop: 0 rgba(79, 79, 79, 1), stop: 1 rgba(79, 79, 79, 1)); border: 2px solid grey; border-radius: 5px;}");

ui->progressBar\_3->setStyleSheet("\*::chunk {background-color: qlineargradient(x1: 0, y1: 0, x2: 0, y2: 1, stop: 0 rgba(79, 79, 79, 1), stop: 1 rgba(0,0,0,0));}");

ui->progressBar\_3->setTextVisible(false);

ui->endButton->setEnabled(0);

}

secondwindow::~***secondwindow***()

{

delete ui;

}

void secondwindow::**on\_horizontalSlider\_valueChanged**(int value)

{

temperature=value;

ui->tempview->display(value);

}

//повышение передач

void secondwindow::**on\_toolButton\_1\_clicked**()

{

if (peredach<4){

peredach+=1;

ui->label\_peredach->setText('('+QString::number(peredach)+')');

}

}

//понижение передач

void secondwindow::**on\_toolButton\_2\_clicked**()

{

if(peredach>0){

peredach-=1;

ui->label\_peredach->setText('('+QString::number(peredach)+')');

}

}

void secondwindow::**on\_randomButton\_clicked**()

{

sec=rand() % 60;

min=rand() % 60;

hour=rand() % 23;

ui->timeEdit->setTime(QTime(hour,min,sec,0));

ui->spinBox\_road->setValue(rand()%100000);

ui->spinBox\_water->setValue(rand()%1000);

ui->spinBox\_pressure->setValue(rand()%1450);

}

void secondwindow::**TimerSLOT**(){

sec++;

if(sec>=60){

min++;

sec-=60;

}

if (min>=60){

hour++;

min-=60;

}

if(hour==24&&min==60&& sec==60){

sec=0;

min=0;

hour=0;

}

//Отображение времени

ui->timeEdit->setTime(QTime(hour,min,sec,0));

//Заполнение бака c водой

if (peredach==0){

if(bak\_woda<water){

if (water<600){

bak\_woda+=10;

}

else {

bak\_woda+=25;

}

}

else{

bak\_woda=water;

}

ui->progressBar->setValue((bak\_woda\*1000)/water);

}

//Заполнение котла с паром

kof\_water=temperature/100;

if (bak\_woda>0){

if(bak\_pressure<pressure){

bak\_pressure+=kof\_water;

bak\_woda-=kof\_water;

}

else{

bak\_pressure=pressure;

}

}

if (bak\_pressure==pressure){

bak\_pressure-=10;

QMessageBox::warning(this, "Осторожно","Давление слишком большое");

temperature=0;

ui->tempview->display(temperature);

}

//Отображение давления

ui->progressBar\_2->setValue((bak\_pressure\*1000)/pressure);

ui->progressBar->setValue((bak\_woda\*1000)/water);

//Работа с передачами и паром

switch (peredach) {

case 1:

if (bak\_pressure>=2)

{

bak\_pressure-=2; bak\_road+=5;

}

break;

case 2:

if (bak\_pressure>=6){

bak\_pressure-=6; bak\_road+=15;

}

break;

case 3: if(bak\_pressure>=10){

bak\_pressure-=10; bak\_road+=20;

}

break;

case 4:

if(bak\_pressure>=14) {

bak\_pressure-=14; bak\_road+=30;

}

break;

}

ui->progressBar->setValue((bak\_woda\*1000)/water);

ui->progressBar\_3->setValue((bak\_road\*100000)/road);

ui->marker->setGeometry(((bak\_road\*1000)/road),20,25,30);

ui->label\_road->setText(QString::number(bak\_road)+' '+':');

if (bak\_road>=road){

QMessageBox::information(this,"Конец","Вы доехали в В от точки А");

timer->stop();

close();

}

}

void secondwindow::**on\_startButton\_clicked**()

{

if (road<=0&&water<=0&&pressure<=0){

QMessageBox::warning(this, "Ошибка","Некоторые поля остались не заполненными");

}

else{

timer->start(1000);

ui->randomButton->setEnabled(0);

ui->spinBox\_road->setEnabled(0);

ui->spinBox\_water->setEnabled(0);

ui->spinBox\_pressure ->setEnabled(0);

ui->startButton->setEnabled(0);

ui->endButton->setEnabled(1);

}

}

void secondwindow::**on\_endButton\_clicked**()

{

timer->stop();

ui->endButton->setEnabled(0);

ui->startButton->setEnabled(1);

}

void secondwindow::**on\_spinBox\_water\_valueChanged**(int arg1)

{

water=arg1;

}

void secondwindow::**on\_spinBox\_pressure\_valueChanged**(int arg1)

{

pressure=arg1;

}

void secondwindow::**on\_spinBox\_road\_valueChanged**(int arg1)

{

road=arg1;

}

void secondwindow::**on\_spinBox\_temp\_valueChanged**(int arg1)

{

temperature= arg1;

ui->tempview->display(temperature);

}

void secondwindow::**on\_timeEdit\_userTimeChanged**(const QTime &time)

{

hour=time.hour();

min=time.minute();

sec=time.second();

}

1. Код файла-источника, содержащего функцию main():

#include "mainwindow.h"

#include "secondwindow.h"

#include <QApplication>

#include <QDialog>

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

{

QApplication a(*argc*, *argv*);

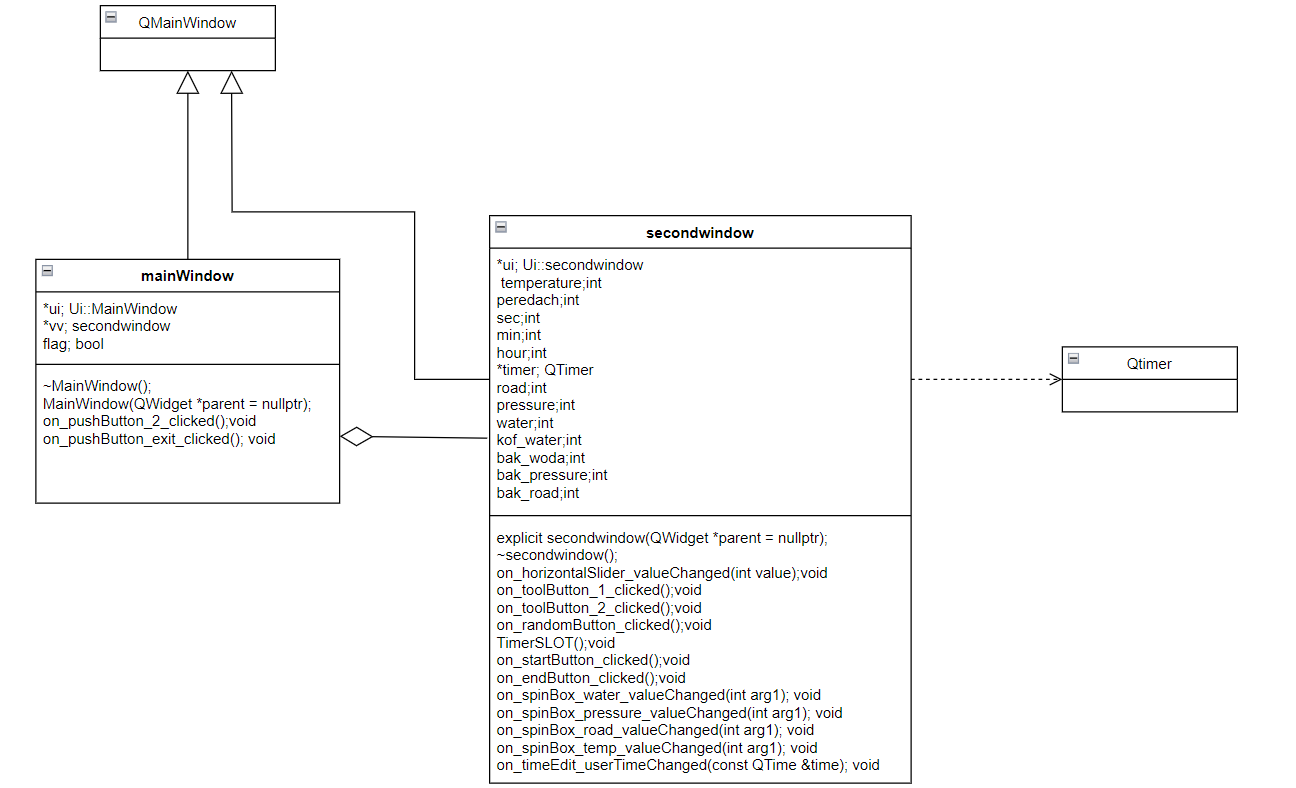
MainWindow w;

w.show();

return a.exec();

}

## Диаграмма классов



## Результат работы

Далее представлен внешний вид проекта АРМ машиниста паровоза:

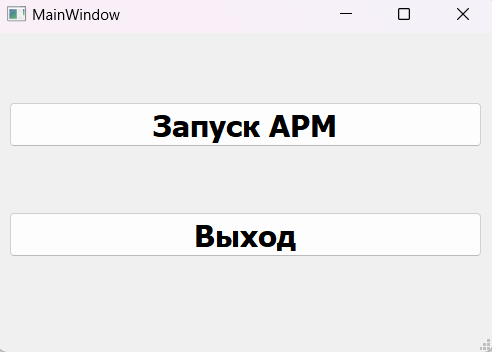


Рисунок 1 －Основное окно проэкта

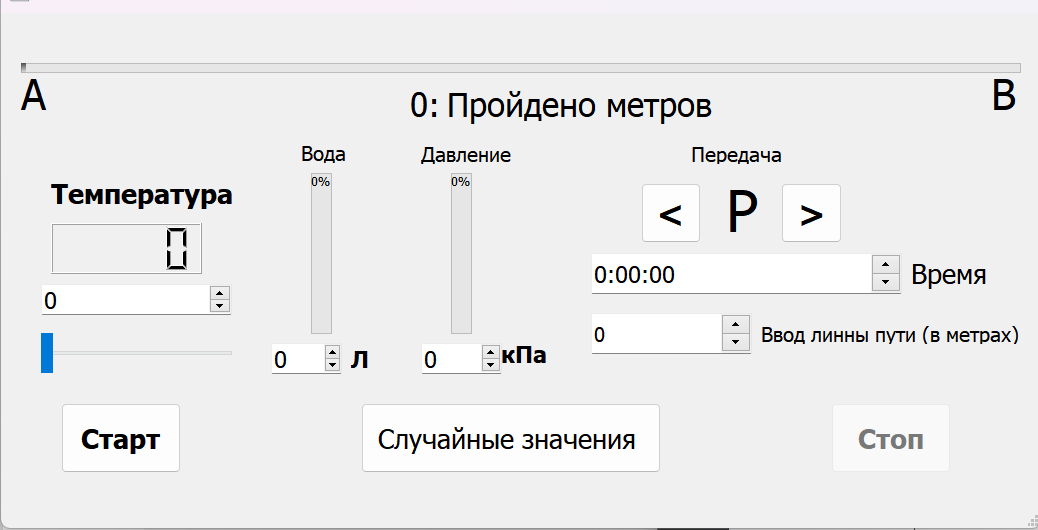


Рисунок 2 －Окно с АРМом

# Решение задачи коммивояжёра

Разработано решение задачи коммивояжёра. Пользователь указывает количество городов, которые необходимо посетить коммивояжёру и расстояния между городами.

1. Слот, проверяющий корректность ввода, и вызывающий отображение таблицы смежности:

void MainWindow::**on\_maketableButton\_clicked**()

{

if (ui->vvodvertex->text().toInt()<3||ui->vvodvertex->text().toInt()>9){

ui->vvodvertex->setText("");

Hide();

ris=0;

this->repaint();

QMessageBox::warning(this,"Ошибка","Количество вершин должно быть не менее 3 и не более 9! ");

}

else{

Hide();

ui->buildButton->show();

r=ui->vvodvertex->text().toInt();

Show();

node = new Node[r];

xcenter=650;

ycenter=300;

ris=1;

this->repaint();

ui->buildButton->setEnabled(1);

}

}

1. Слот, заносящий в массив путей значения из таблицы смежности, и разрешающий отображение путей на схеме:

void MainWindow::**on\_buildButton\_clicked**()

{

QString searcher,buffer;

for (int i=1;i<r;i++){

for (int e=i+1;e<r+1;e++){

searcher="p"+QString::number(i) + QString::number(e);

buffer=this->findChild <QLineEdit\*> (searcher)->text();

if (!buffer.length()) buffer='0';

searcher="n"+QString::number(e) + QString::number(i);

this->findChild<QLabel\*>(searcher)->setText(buffer);

}

}

for (int i=0;i<r-1;i++){

for (int e=i+1;e<r;e++){

searcher="n"+QString::number(e+1) + QString::number(i+1);

buffer=this->findChild<QLabel\*>(searcher)->text();

if (!buffer.length()) buffer='0';

matrix[i][e]=buffer.toInt();

matrix[e][i]=matrix[i][e];

}

}

ui->roadButton->setEnabled(1);

ris=2;

this->repaint();

}

1. Слот, проверяющий существование пути, соответствующего условию задачи, запускающий алгоритм поиска и разрешающий отображение кратчайшего пути на схеме:

void MainWindow::**on\_roadButton\_clicked**(){

QString searcher,buffer;

leng=0;

s=0;

mins=10000;

was = new bool [r];

way = new int [r];

minway = new int [r];

for (int i=0;i<r;i++) Perebor(0,i);

buffer="Кратчайший путь: " + QString::number(minway[0]+1);

for (int i=1;i<r;i++) buffer += "->" + QString::number(minway[i]+1);

if (mins==10000){

QMessageBox::warning(this,"Предупреждение","Не существует маршрута, позволяющего посетить все города, или невозможно посетить их без повторного посещения пройденных дорог и городов. ");

}

else{

buffer+= "\n Длина пути: " +QString::number(mins);

QMessageBox::information(this,"Результат",buffer);

}

delete[] was;

delete[] way;

ui->roadButton->setEnabled(0);

}

1. Функция, находящая кратчайший путь и его длину:

void MainWindow::**Perebor**(int l, int i){

leng+=l;

was[i]=1;

bool b=0;

way[s]=i;

s++;

for (int e=0;e<r;e++){

if (!was[e]){

if (matrix[i][e]) Perebor(matrix[i][e],e);

b=1;

}

}

if(!b){

if (leng<mins){

mins=leng;

for (int z=0;z<r;z++) minway[z]=way[z];

}

}

leng-=l;

s--;

was[i]=0;

}

## 

## Полный код программы

1. Код заголовочного файла mainwindow:

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QPainter>

#include <QPaintEvent>

#include <QVector>

#include <QLabel>

#include <QMessageBox>

#include <QDebug>

#include <cmath>

QT\_BEGIN\_NAMESPACE

namespace **Ui** { class **MainWindow**; }

QT\_END\_NAMESPACE

class **MainWindow** : public QMainWindow

{

Q\_OBJECT

public:

**MainWindow**(QWidget \*parent = nullptr);

~***MainWindow***();

void **Hide**();

void **Show**();

void **Perebor**(int,int);

private slots:

void **on\_maketableButton\_clicked**();

void **on\_buildButton\_clicked**();

void **on\_roadButton\_clicked**();

private:

Ui::MainWindow \*ui;

int matrix[10][10];//Матрица смежности

QString solution;

QPainter paint;

virtual void ***paintEvent***(QPaintEvent \*) ;

};

#endif // MAINWINDOW\_H

1. Код файла-источника, содержащего основной класс:

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <QLineEdit>

#include <QLabel>

#define N 10

double grad=360;

int r,ris=0,prirost,xcenter,ycenter,mins=10000,leng,s;

struct **Node**{

int x,y;

};

bool \*was;

int \*way, \*minway;

Node \*node;

MainWindow::**MainWindow**(QWidget \*parent)

: QMainWindow(*parent*)

, ui(new Ui::MainWindow)

{

ui->setupUi(this);

solution="";

//Ограничение доступа к кнопкам

ui->buildButton->setEnabled(0);

ui->roadButton->setEnabled(0);

//Задание маски блоков ввода

ui->vvodvertex->setInputMask("9");

ui->p12->setInputMask("99");

ui->p13->setInputMask("99");

ui->p14->setInputMask("99");

ui->p15->setInputMask("99");

ui->p16->setInputMask("99");

ui->p17->setInputMask("99");

ui->p18->setInputMask("99");

ui->p19->setInputMask("99");

ui->p23->setInputMask("99");

ui->p24->setInputMask("99");

ui->p25->setInputMask("99");

ui->p26->setInputMask("99");

ui->p27->setInputMask("99");

ui->p28->setInputMask("99");

ui->p29->setInputMask("99");

ui->p34->setInputMask("99");

ui->p35->setInputMask("99");

ui->p36->setInputMask("99");

ui->p37->setInputMask("99");

ui->p37->setInputMask("99");

ui->p38->setInputMask("99");

ui->p39->setInputMask("99");

ui->p45->setInputMask("99");

ui->p46->setInputMask("99");

ui->p47->setInputMask("99");

ui->p48->setInputMask("99");

ui->p49->setInputMask("99");

ui->p56->setInputMask("99");

ui->p57->setInputMask("99");

ui->p58->setInputMask("99");

ui->p59->setInputMask("99");

ui->p67->setInputMask("99");

ui->p68->setInputMask("99");

ui->p69->setInputMask("99");

ui->p78->setInputMask("99");

ui->p79->setInputMask("99");

ui->p89->setInputMask("99");

Hide();

}

MainWindow::~***MainWindow***()

{

delete ui;

}

//Функция скрытия матрицы смежности с основного виджета

void MainWindow::**Hide**(){

QString pol;//Строковая переменная нужная для считывание значений с нужных виджетов

for (int i=1;i<10;i++){

pol="gor" + QString::number(i);

this->findChild<QWidget\*>(pol)->hide();

}

for (int i=1; i<10; i++){

pol="vert"+QString::number(i);

this->findChild<QWidget\*>(pol)->hide();

pol="dig"+QString::number(i);

this->findChild<QWidget\*>(pol)->hide();

for(int e=i+1;e<10;e++ ){

pol="p"+QString::number(i)+QString::number(e);

this->findChild<QWidget\*>(pol)->hide();

}

for(int e=1;e<i;e++){

pol="n"+QString::number(i)+QString::number(e);

this->findChild<QWidget\*>(pol)->hide();

}

}

}

//Функция вывода матрицы смежности на основной виджет

void MainWindow::**Show**(){

QString pol; //Строковая переменная нужная для считывание значений с нужных виджетов

for (int i=1;i<r+1;i++){

pol="gor" + QString::number(i);

this->findChild<QWidget\*>(pol)->show();

}

for (int i=1; i<r+1; i++){

pol="vert"+QString::number(i);

this->findChild<QWidget\*>(pol)->show();

pol="dig"+QString::number(i);

this->findChild<QWidget\*>(pol)->show();

for(int e=i+1;e<r+1;e++ ){

pol="p"+QString::number(i)+QString::number(e);

this->findChild<QWidget\*>(pol)->show();

}

for(int e=1;e<i;e++){

pol="n"+QString::number(i)+QString::number(e);

this->findChild<QWidget\*>(pol)->show();

}

}

}

//Функция отрисовки событий

void MainWindow::***paintEvent***(QPaintEvent \*)

{

if (ris){

int otstup;

grad/=r;

prirost=grad;

grad=0;

//Объявление холста с карандашом

QPainter paint;

QPen pen;

QBrush brush;

QString stroka;

QFont font;

pen.setColor(Qt::black);

pen.setWidth(4);

pen.setStyle(Qt::SolidLine);

font.setPointSize(20);

paint.begin(this);

paint.setPen(pen);

paint.setFont(font);

for (int i=0;i<r;i++){

otstup=-10;

//Отрисока кругов(вершин графа)

if (grad>180) otstup=100;

stroka=QString::number(i+1);

node[i].x=xcenter+(150\*cos(grad/57));

node[i].y=ycenter+(150\*-sin(grad/57));

paint.drawEllipse(node[i].x,node[i].y,50,50);

paint.drawText(node[i].x+25,node[i].y+otstup,stroka);

grad+=prirost;

}

if (ris>1){

//Отрисока линий(ребер графа)

for (int i=0;i<r;i++){

for (int e=i+1;e<r;e++){

if (matrix[i][e]){

pen.setStyle(Qt::SolidLine);

pen.setWidth(3);

paint.setPen(pen);

paint.drawLine(node[i].x+25,node[i].y+25,node[e].x+25,node[e].y+25);

}

}

}

}

paint.end();

}

}

//Слот вывода матрицы на основной виджет

void MainWindow::**on\_maketableButton\_clicked**()

{

if (ui->vvodvertex->text().toInt()<3||ui->vvodvertex->text().toInt()>9){

ui->vvodvertex->setText("");

Hide();

ris=0;

this->repaint();

QMessageBox::warning(this,"Ошибка","Количество вершин должно быть не менее 3 и не более 9! ");

}

else{

Hide();

ui->buildButton->show();

r=ui->vvodvertex->text().toInt();

Show();

node = new Node[r];

xcenter=650;

ycenter=300;

ris=1;

this->repaint();

ui->buildButton->setEnabled(1);

}

}

//Слот считывания матрицы с основного виджета

void MainWindow::**on\_buildButton\_clicked**()

{

QString searcher,buffer;

for (int i=1;i<r;i++){

for (int e=i+1;e<r+1;e++){

searcher="p"+QString::number(i) + QString::number(e);

buffer=this->findChild <QLineEdit\*> (searcher)->text();

if (!buffer.length()) buffer='0';

searcher="n"+QString::number(e) + QString::number(i);

this->findChild<QLabel\*>(searcher)->setText(buffer);

}

}

for (int i=0;i<r-1;i++){

for (int e=i+1;e<r;e++){

searcher="n"+QString::number(e+1) + QString::number(i+1);

buffer=this->findChild<QLabel\*>(searcher)->text();

if (!buffer.length()) buffer='0';

matrix[i][e]=buffer.toInt();

matrix[e][i]=matrix[i][e];

}

}

ui->roadButton->setEnabled(1);

ris=2;

this->repaint();

}

//Слот вычисления кратчайшего пути

void MainWindow::**on\_roadButton\_clicked**(){

QString searcher,buffer;

leng=0;

s=0;

mins=10000;

was = new bool [r];

way = new int [r];

minway = new int [r];

for (int i=0;i<r;i++) Perebor(0,i);

buffer="Кратчайший путь: " + QString::number(minway[0]+1);

for (int i=1;i<r;i++) buffer += "->" + QString::number(minway[i]+1);

if (mins==10000){

QMessageBox::warning(this,"Предупреждение","Не существует маршрута, позволяющего посетить все города, или невозможно посетить их без повторного посещения пройденных дорог и городов. ");

}

else{

buffer+= "\n Длина пути: " +QString::number(mins);

QMessageBox::information(this,"Результат",buffer);

}

delete[] was;

delete[] way;

ui->roadButton->setEnabled(0);

}

//Вычисление маршрута перебором

void MainWindow::**Perebor**(int l, int i){

leng+=l;

was[i]=1;

bool b=0;

way[s]=i;

s++;

for (int e=0;e<r;e++){

if (!was[e]){

if (matrix[i][e]) Perebor(matrix[i][e],e);

b=1;

}

}

if(!b){

if (leng<mins){

mins=leng;

for (int z=0;z<r;z++) minway[z]=way[z];

}

}

leng-=l;

s--;

was[i]=0;

}

1. Код файла, содержащего функцию main():

#include "mainwindow.h"

#include <QApplication>

using namespace std;

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

{

QApplication a(*argc*, *argv*);

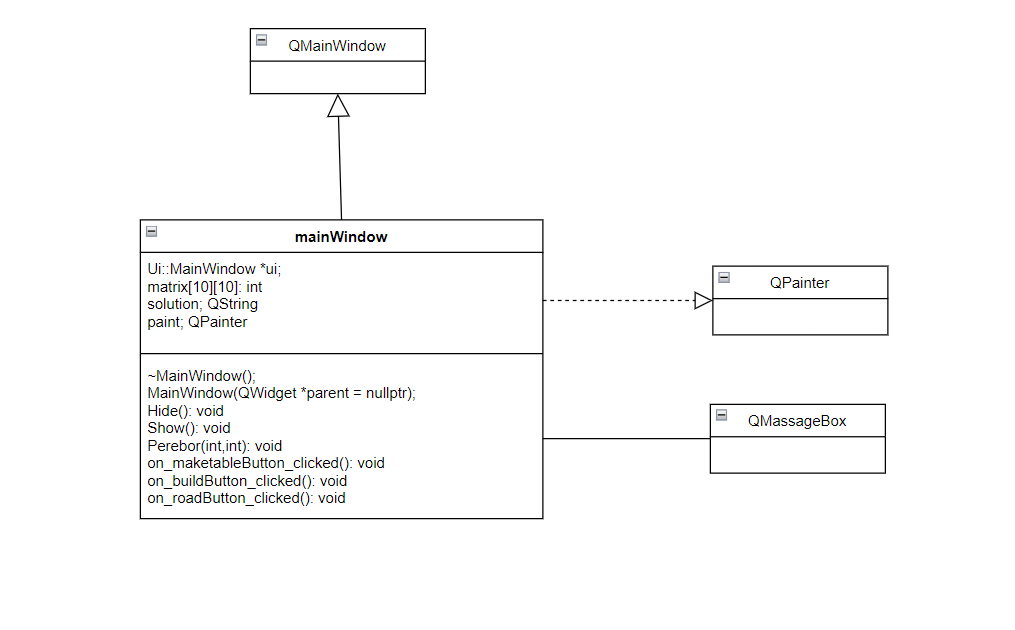
MainWindow w;

w.show();

return a.exec();

}

## Диаграмма классов



## Результат работы

Внешний вид окна программы:

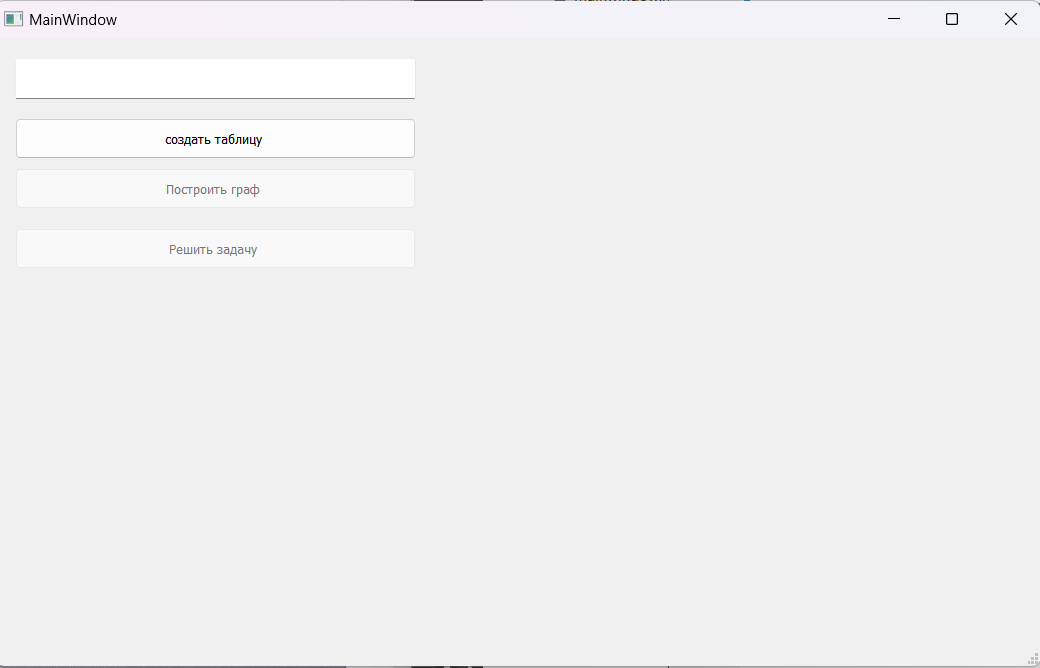


Рисунок 3 - Изначальное окно программы

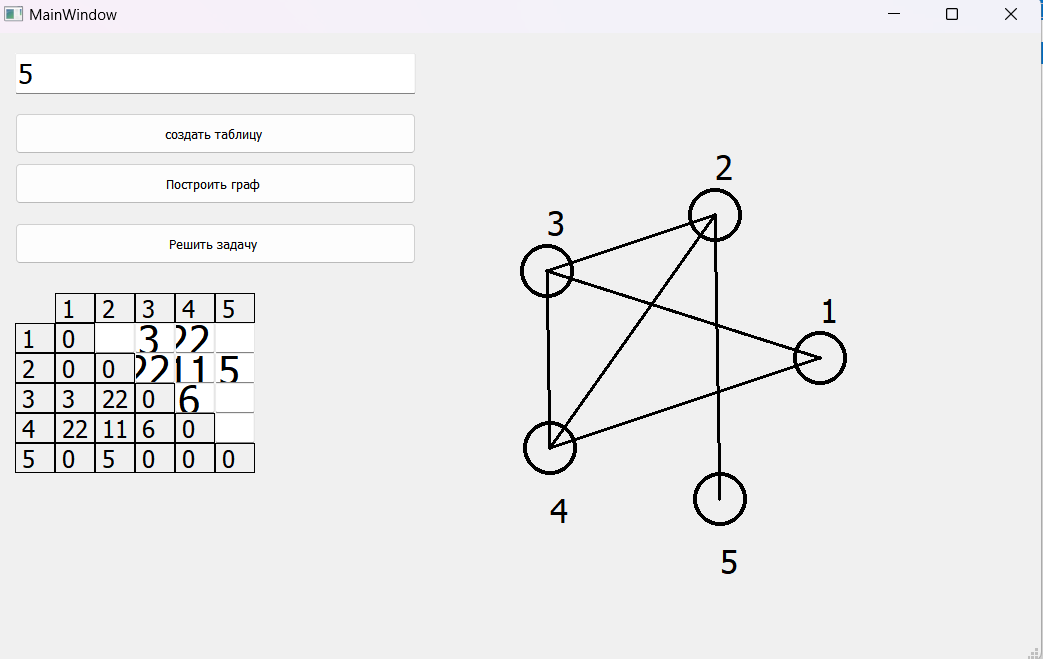


Рисунок 4 - Построение графика и таблицы смежности

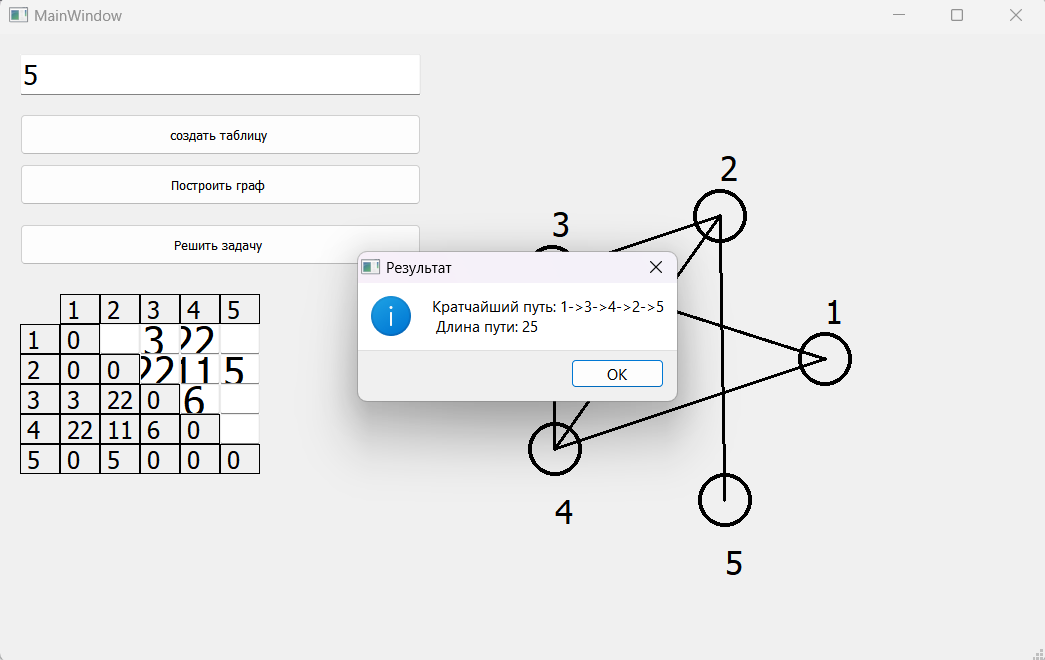


Рисунок 5 - Решение задачи коммивояжера

# Вывод

Средствами фреймворка Qt реализовано приложение, выполняющее функции АРМа машиниста паровоза. Также средствами Qt реализовано приложение, позволяющее решить задачу коммивояжёра.

# Используемое программное обеспечение

1. Обе части творческой работы реализованы с помощью фреймворка Qt.
2. Диаграммы классов составлены в программе draw.io.
3. Видео – демонстрирующее результат творческой работы записано с помощью программы OBS.