Министерство образования и науки Российской Федерации  
Федеральное‌ ‌государственное‌ ‌бюджетное‌ ‌образовательное‌ ‌учреждение‌

высшего‌ ‌образования‌

**«Пермский национальный исследовательский  
политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**О Т Ч Ё Т**

**по творческой работе**

Тема: Реализация калькулятора и программное решение задачи коммивояжёра средствами Qt.

Вариант 14

Выполнил:

студент группы РИС-20-2б

Вичугов Алексей Дмитриевич

Пермь, 2021

**Введение**

Средствами фреймворка Qt реализовано приложение, выполняющее функции калькулятора, вычисляющего временные интервалы и производящего различные арифметические операции с временными величинами. Также средствами Qt реализовано приложение, позволяющее решить задачу коммивояжёра с дополнительным условием: запрещено повторно посещать ранее посещённые города и двигаться по одной дороге более одного раза.

**Реализация калькулятора временных величин**

Функции, которые должен производить калькулятор согласно варианту: Ввод данных, ввод только в часах, минутах или секундах, нахождение величины временного интервала, суммы, разности, преобразование интервала в часы, минуты, секунды.

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 Klass::on\_clear\_clicked(){}
2. Арифметические операции с введёнными значениями производятся при нажатии на кнопку «Вычислить», в зависимости от выбранной операции над значениями производятся разные действия. Для того, чтобы временной интервал приводился в правильной форме, используется операция преобразования типов. Код функции, производящей вычисления:

void Klass::**on\_go\_clicked**()

{

if(ui->consta->text().toDouble()==0) ui->consta->setText("1");

switch (op) {

case 0: r=(ui->hours->text()).toDouble()\*3600+(ui->minuts->text()).toDouble()\*60+(ui->secunds->text()).toDouble()+(ui->hours\_2->text()).toDouble()\*3600+(ui->minuts\_2->text()).toDouble()\*60+(ui->secunds\_2->text()).toDouble();break;

case 1: r=((ui->hours->text()).toDouble()\*3600+(ui->minuts->text()).toDouble()\*60+(ui->secunds->text()).toDouble())-((ui->hours\_2->text()).toDouble()\*3600+(ui->minuts\_2->text()).toDouble()\*60+(ui->secunds\_2->text()).toDouble());break;

case 2: r=((ui->hours->text()).toDouble()\*3600+(ui->minuts->text()).toDouble()\*60+(ui->secunds->text()).toDouble())\*(ui->consta->text()).toDouble();break;

case 3: r=((ui->hours->text()).toDouble()\*3600+(ui->minuts->text()).toDouble()\*60+(ui->secunds->text()).toDouble())/(ui->consta->text()).toDouble();break;

}

h=static\_cast<int>(r)/3600;

r-=h\*3600;

m=static\_cast<int>(r)/60;

r-=m\*60;

s=r;

result=QString::number(h,'g',15)+':'+QString::number(m,'g',15)+':'+QString::number(s,'g',15);

ui->resval->setText(result);

}

1. Дизайн создан с применением таблиц стилей. При нажатии на кнопку вызывается функция, меняющая внешний вид кнопки:

void Klass::**SetButtonColor**(){

QPushButton \*button = (QPushButton \*)sender();

button->setStyleSheet("background-color: rgb(0,200,200);border-radius: 10px;border: 1px solid black;font-size: 30px;color: white;");

if(button==ui->clear) ui->clear->setStyleSheet("background-color: rgb(0,200,200);border-radius: 10px;border: 1px solid black;font-size: 18px;color: white;");

if(button==ui->inh) ui->inh->setStyleSheet("background-color: rgb(0,180,130);border-radius: 10px;border: 1px solid black;color: white;font-size: 16px;");

if(button==ui->inm) ui->inm->setStyleSheet("background-color: rgb(0,180,130);border-radius: 10px;border: 1px solid black;color: white;font-size: 16px;");

if(button==ui->ins) ui->ins->setStyleSheet("background-color: rgb(0,180,130);border-radius: 10px;border: 1px solid black;color: white;font-size: 16px;");

}

1. Для выбора арифметической операции используются четыре функции, меняющие значение переменной, от которой зависит, какие действия над введёнными величинами будут производиться:

void Klass::**on\_plus\_clicked**()

{

ui->plus->setStyleSheet("background-color: rgb(0,220,220);border-radius: 10px;border: 1px solid black;font-size: 30px;color: white;");

ui->znak->setText("+");

op=0;

ShowLines();

}

void Klass::**on\_minus\_clicked**()

{

ui->minus->setStyleSheet("background-color: rgb(0,220,220);border-radius: 10px;border: 1px solid black;font-size: 30px;color: white;");

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

op=1;

ShowLines();

}

void Klass::**on\_umnojit\_clicked**()

{

ui->umnojit->setStyleSheet("background-color: rgb(0,220,220);border-radius: 10px;border: 1px solid black;font-size: 30px;color: white;");

ui->znak->setText("\*");

op=2;

ShowLines();

}

void Klass::**on\_delit\_clicked**()

{

ui->delit->setStyleSheet("background-color: rgb(0,220,220);border-radius: 10px;border: 1px solid black;font-size: 30px;color: white;");

ui->znak->setText("/");

op=3;

ShowLines();

}

1. Для перехода на следующее поле ввода вызывается функция nextline(), которая вычисляет, какая строка должна стать активной, и вызывает функцию StandartColor(), настраивающую внешний вид для каждого поля ввода. Функции nextline() и StandartColor():

void Klass::**nextline**(){

if (op<2){

if (stroka<6) stroka++;

else stroka=1;

}

else{

if (stroka<3) stroka++;

else if (stroka==3) stroka=8;

else stroka=1;

}

StandartColor();

}

void Klass::**StandartColor**(){

ui->hours->setStyleSheet("background-color: rgb(0,200,230);border-radius: 10px;border: 1px solid black;color: white;");

ui->minuts->setStyleSheet("background-color: rgb(0,200,230);border-radius: 10px;border: 1px solid black;color: white;");

ui->secunds->setStyleSheet("background-color: rgb(0,200,230);border-radius: 10px;border: 1px solid black;color: white;");

ui->hours\_2->setStyleSheet("background-color: rgb(0,200,230);border-radius: 10px;border: 1px solid black;color: white;");

ui->minuts\_2->setStyleSheet("background-color: rgb(0,200,230);border-radius: 10px;border: 1px solid black;color: white;");

ui->secunds\_2->setStyleSheet("background-color: rgb(0,200,230);border-radius: 10px;border: 1px solid black;color: white;");

ui->consta->setStyleSheet("background-color: rgb(0,200,230);border-radius: 10px;border: 1px solid black;color: white;");

switch (stroka) {

case 1:ui->hours->setStyleSheet("background-color: rgb(200,00,100);border-radius: 10px;border: 1px solid black;color: white;");ui->hours->setFocus();break;

case 2: ui->minuts->setStyleSheet("background-color: rgb(200,00,100);border-radius: 10px;border: 1px solid black;color: white;");ui->minuts->setFocus();break;

case 3: ui->secunds->setStyleSheet("background-color: rgb(200,00,100);border-radius: 10px;border: 1px solid black;color: white;");ui->secunds->setFocus();break;

case 4: ui->hours\_2->setStyleSheet("background-color: rgb(200,00,100);border-radius: 10px;border: 1px solid black;color: white;");ui->hours\_2->setFocus();break;

case 5: ui->minuts\_2->setStyleSheet("background-color: rgb(200,00,100);border-radius: 10px;border: 1px solid black;color: white;");ui->minuts\_2->setFocus();break;

case 6: ui->secunds\_2->setStyleSheet("background-color: rgb(200,00,100);border-radius: 10px;border: 1px solid black;color: white;");ui->secunds\_2->setFocus();break;

case 8: ui->consta->setStyleSheet("background-color: rgb(200,00,100);border-radius: 10px;border: 1px solid black;color: white;");ui->consta->setFocus();break;

}

}

**Полный код программы**

Калькулятор реализован с использованием нескольких файлов:

1. Код заголовочного файла, содержащего описание класса:  
   #ifndef KLASS\_H

#define KLASS\_H

#include <QMainWindow>

QT\_BEGIN\_NAMESPACE

namespace **Ui** { class **Klass**; }

QT\_END\_NAMESPACE

class **Klass** : public QMainWindow

{

Q\_OBJECT

public:

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

~***Klass***();

private slots:

void **number**();

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

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

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

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

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

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

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

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

void **nextline**();

void **StandartColor**();

void **SetButtonColor**();

void **ShowLines**();

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

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

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

private:

Ui::Klass \*ui;

};

#endif // KLASS\_H

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

#include "klass.h"

#include <QApplication>

#include <QLabel>

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

{

QApplication a(*argc*, argv);

Klass w;

w.show();

return a.exec();

}

1. Код файла-источника, содержащего определения компонентных функций класса Klass:

#include "klass.h"

#include "ui\_klass.h"

int op=0;

int stroka=1;

QString result;

double h,m,s,r;

Klass::**Klass**(QWidget \*parent)

: QMainWindow(parent)

, ui(new Ui::Klass)

{

ui->setupUi(this);

ui->CText->hide();

ui->consta->hide();

ui->hours->setFocus();

StandartColor();

connect(ui->b0,SIGNAL(clicked()),this,SLOT(number()));

connect(ui->b1,SIGNAL(clicked()),this,SLOT(number()));

connect(ui->b2,SIGNAL(clicked()),this,SLOT(number()));

connect(ui->b3,SIGNAL(clicked()),this,SLOT(number()));

connect(ui->b4,SIGNAL(clicked()),this,SLOT(number()));

connect(ui->b5,SIGNAL(clicked()),this,SLOT(number()));

connect(ui->b6,SIGNAL(clicked()),this,SLOT(number()));

connect(ui->b7,SIGNAL(clicked()),this,SLOT(number()));

connect(ui->b8,SIGNAL(clicked()),this,SLOT(number()));

connect(ui->b9,SIGNAL(clicked()),this,SLOT(number()));

connect(ui->b0,SIGNAL(pressed()),this,SLOT(SetButtonColor()));

connect(ui->b1,SIGNAL(pressed()),this,SLOT(SetButtonColor()));

connect(ui->b2,SIGNAL(pressed()),this,SLOT(SetButtonColor()));

connect(ui->b3,SIGNAL(pressed()),this,SLOT(SetButtonColor()));

connect(ui->b4,SIGNAL(pressed()),this,SLOT(SetButtonColor()));

connect(ui->b5,SIGNAL(pressed()),this,SLOT(SetButtonColor()));

connect(ui->b6,SIGNAL(pressed()),this,SLOT(SetButtonColor()));

connect(ui->b7,SIGNAL(pressed()),this,SLOT(SetButtonColor()));

connect(ui->b8,SIGNAL(pressed()),this,SLOT(SetButtonColor()));

connect(ui->b9,SIGNAL(pressed()),this,SLOT(SetButtonColor()));

connect(ui->plus,SIGNAL(pressed()),this,SLOT(SetButtonColor()));

connect(ui->minus,SIGNAL(pressed()),this,SLOT(SetButtonColor()));

connect(ui->umnojit,SIGNAL(pressed()),this,SLOT(SetButtonColor()));

connect(ui->delit,SIGNAL(pressed()),this,SLOT(SetButtonColor()));

connect(ui->bpoint,SIGNAL(pressed()),this,SLOT(SetButtonColor()));

connect(ui->doble,SIGNAL(pressed()),this,SLOT(SetButtonColor()));

connect(ui->clear,SIGNAL(pressed()),this,SLOT(SetButtonColor()));

connect(ui->inh,SIGNAL(pressed()),this,SLOT(SetButtonColor()));

connect(ui->inm,SIGNAL(pressed()),this,SLOT(SetButtonColor()));

connect(ui->ins,SIGNAL(pressed()),this,SLOT(SetButtonColor()));

connect(ui->hours,SIGNAL(returnPressed()),this,SLOT(nextline()));

connect(ui->minuts,SIGNAL(returnPressed()),this,SLOT(nextline()));

connect(ui->secunds,SIGNAL(returnPressed()),this,SLOT(nextline()));

connect(ui->hours\_2,SIGNAL(returnPressed()),this,SLOT(nextline()));

connect(ui->minuts\_2,SIGNAL(returnPressed()),this,SLOT(nextline()));

connect(ui->secunds\_2,SIGNAL(returnPressed()),this,SLOT(nextline()));

}

Klass::~***Klass***()

{

delete ui;

}

void Klass::**number**()

{

QPushButton \*button = (QPushButton \*)sender();

button->setStyleSheet("background-color: rgb(0,220,220);border-radius: 10px;border: 1px solid black;font-size: 30px;color: white;");

QString buf;

double dbuf;

switch(stroka){

case 1:

dbuf = (ui->hours->text() + button->text()).toDouble();

buf = QString::number(dbuf,'g',15);

ui->hours->setText(buf);

break;

case 2:

dbuf = (ui->minuts->text() + button->text()).toDouble();

buf = QString::number(dbuf,'g',15);

ui->minuts->setText(buf);

break;

case 3:

dbuf = (ui->secunds->text() + button->text()).toDouble();

buf = QString::number(dbuf,'g',15);

ui->secunds->setText(buf);

break;

case 4:

dbuf = (ui->hours\_2->text() + button->text()).toDouble();

buf = QString::number(dbuf,'g',15);

ui->hours\_2->setText(buf);

break;

case 5:

dbuf = (ui->minuts\_2->text() + button->text()).toDouble();

buf = QString::number(dbuf,'g',15);

ui->minuts\_2->setText(buf);

break;

case 6:

dbuf = (ui->secunds\_2->text() + button->text()).toDouble();

buf = QString::number(dbuf,'g',15);

ui->secunds\_2->setText(buf);

break;

case 8:

dbuf = (ui->consta->text() + button->text()).toDouble();

buf = QString::number(dbuf,'g',15);

ui->consta->setText(buf);

break;

}

}

void Klass::**on\_clear\_clicked**()

{

ui->clear->setStyleSheet("background-color: rgb(0,220,220);border-radius: 10px;border: 1px solid black;font-size: 18px;color: white;");

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

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

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

ui->hours\_2->setText("");

ui->minuts\_2->setText("");

ui->secunds\_2->setText("");

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

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

h=0;

m=0;

s=0;

r=0;

stroka=1;

ui->hours->setFocus();

StandartColor();

}

void Klass::**on\_go\_clicked**()

{

if(ui->consta->text().toDouble()==0) ui->consta->setText("1");

switch (op) {

case 0: r=(ui->hours->text()).toDouble()\*3600+(ui->minuts->text()).toDouble()\*60+(ui->secunds->text()).toDouble()+(ui->hours\_2->text()).toDouble()\*3600+(ui->minuts\_2->text()).toDouble()\*60+(ui->secunds\_2->text()).toDouble();break;

case 1: r=((ui->hours->text()).toDouble()\*3600+(ui->minuts->text()).toDouble()\*60+(ui->secunds->text()).toDouble())-((ui->hours\_2->text()).toDouble()\*3600+(ui->minuts\_2->text()).toDouble()\*60+(ui->secunds\_2->text()).toDouble());break;

case 2: r=((ui->hours->text()).toDouble()\*3600+(ui->minuts->text()).toDouble()\*60+(ui->secunds->text()).toDouble())\*(ui->consta->text()).toDouble();break;

case 3: r=((ui->hours->text()).toDouble()\*3600+(ui->minuts->text()).toDouble()\*60+(ui->secunds->text()).toDouble())/(ui->consta->text()).toDouble();break;

}

h=static\_cast<int>(r)/3600;

r-=h\*3600;

m=static\_cast<int>(r)/60;

r-=m\*60;

s=r;

result=QString::number(h,'g',15)+':'+QString::number(m,'g',15)+':'+QString::number(s,'g',15);

ui->resval->setText(result);

}

void Klass::**ShowLines**(){

switch (op) {

case 0: ui->CText->hide();ui->consta->hide();ui->hours\_2->show();ui->minuts\_2->show();ui->secunds\_2->show();ui->Htext2->show();ui->Mtext2->show();ui->Stext2->show();break;

case 1: ui->CText->hide();ui->consta->hide();ui->hours\_2->show();ui->minuts\_2->show();ui->secunds\_2->show();ui->Htext2->show();ui->Mtext2->show();ui->Stext2->show();break;

case 2: ui->CText->show();ui->consta->show();ui->hours\_2->hide();ui->minuts\_2->hide();ui->secunds\_2->hide();ui->Htext2->hide();ui->Mtext2->hide();ui->Stext2->hide();break;

case 3: ui->CText->show();ui->consta->show();;ui->hours\_2->hide();ui->minuts\_2->hide();ui->secunds\_2->hide();ui->Htext2->hide();ui->Mtext2->hide();ui->Stext2->hide();break;

}

}

void Klass::**on\_plus\_clicked**()

{

ui->plus->setStyleSheet("background-color: rgb(0,220,220);border-radius: 10px;border: 1px solid black;font-size: 30px;color: white;");

ui->znak->setText("+");

op=0;

ShowLines();

}

void Klass::**on\_minus\_clicked**()

{

ui->minus->setStyleSheet("background-color: rgb(0,220,220);border-radius: 10px;border: 1px solid black;font-size: 30px;color: white;");

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

op=1;

ShowLines();

}

void Klass::**on\_umnojit\_clicked**()

{

ui->umnojit->setStyleSheet("background-color: rgb(0,220,220);border-radius: 10px;border: 1px solid black;font-size: 30px;color: white;");

ui->znak->setText("\*");

op=2;

ShowLines();

}

void Klass::**on\_delit\_clicked**()

{

ui->delit->setStyleSheet("background-color: rgb(0,220,220);border-radius: 10px;border: 1px solid black;font-size: 30px;color: white;");

ui->znak->setText("/");

op=3;

ShowLines();

}

void Klass::**on\_doble\_clicked**()

{

ui->doble->setStyleSheet("background-color: rgb(0,220,220);border-radius: 10px;border: 1px solid black;font-size: 30px;color: white;");

nextline();

}

void Klass::**on\_ins\_clicked**()

{

ui->ins->setStyleSheet("background-color: rgb(0,200,150);border-radius: 10px;border: 1px solid black;color: white;font-size: 16px;");

r=h\*3600+m\*60+s;

result=QString::number(r,'g',15);

ui->resval->setText(result);

}

void Klass::**on\_inm\_clicked**()

{

ui->inm->setStyleSheet("background-color: rgb(0,200,150);border-radius: 10px;border: 1px solid black;color: white;font-size: 16px;");

r=h\*60+m+s/60;

result=QString::number(r,'g',15);

ui->resval->setText(result);

}

void Klass::**on\_inh\_clicked**()

{

ui->inh->setStyleSheet("background-color: rgb(0,200,150);border-radius: 10px;border: 1px solid black;color: white;font-size: 16px;");

r=h+m/60+s/3600;

result=QString::number(r,'g',10);

ui->resval->setText(result);

}

void Klass::**StandartColor**(){

ui->hours->setStyleSheet("background-color: rgb(0,200,230);border-radius: 10px;border: 1px solid black;color: white;");

ui->minuts->setStyleSheet("background-color: rgb(0,200,230);border-radius: 10px;border: 1px solid black;color: white;");

ui->secunds->setStyleSheet("background-color: rgb(0,200,230);border-radius: 10px;border: 1px solid black;color: white;");

ui->hours\_2->setStyleSheet("background-color: rgb(0,200,230);border-radius: 10px;border: 1px solid black;color: white;");

ui->minuts\_2->setStyleSheet("background-color: rgb(0,200,230);border-radius: 10px;border: 1px solid black;color: white;");

ui->secunds\_2->setStyleSheet("background-color: rgb(0,200,230);border-radius: 10px;border: 1px solid black;color: white;");

ui->consta->setStyleSheet("background-color: rgb(0,200,230);border-radius: 10px;border: 1px solid black;color: white;");

switch (stroka) {

case 1:ui->hours->setStyleSheet("background-color: rgb(200,00,100);border-radius: 10px;border: 1px solid black;color: white;");ui->hours->setFocus();break;

case 2: ui->minuts->setStyleSheet("background-color: rgb(200,00,100);border-radius: 10px;border: 1px solid black;color: white;");ui->minuts->setFocus();break;

case 3: ui->secunds->setStyleSheet("background-color: rgb(200,00,100);border-radius: 10px;border: 1px solid black;color: white;");ui->secunds->setFocus();break;

case 4: ui->hours\_2->setStyleSheet("background-color: rgb(200,00,100);border-radius: 10px;border: 1px solid black;color: white;");ui->hours\_2->setFocus();break;

case 5: ui->minuts\_2->setStyleSheet("background-color: rgb(200,00,100);border-radius: 10px;border: 1px solid black;color: white;");ui->minuts\_2->setFocus();break;

case 6: ui->secunds\_2->setStyleSheet("background-color: rgb(200,00,100);border-radius: 10px;border: 1px solid black;color: white;");ui->secunds\_2->setFocus();break;

case 8: ui->consta->setStyleSheet("background-color: rgb(200,00,100);border-radius: 10px;border: 1px solid black;color: white;");ui->consta->setFocus();break;

}

}

void Klass::**nextline**(){

if (op<2){

if (stroka<6) stroka++;

else stroka=1;

}

else{

if (stroka<3) stroka++;

else if (stroka==3) stroka=8;

else stroka=1;

}

StandartColor();

}

void Klass::**SetButtonColor**(){

QPushButton \*button = (QPushButton \*)sender();

button->setStyleSheet("background-color: rgb(0,200,200);border-radius: 10px;border: 1px solid black;font-size: 30px;color: white;");

if(button==ui->clear) ui->clear->setStyleSheet("background-color: rgb(0,200,200);border-radius: 10px;border: 1px solid black;font-size: 18px;color: white;");

if(button==ui->inh) ui->inh->setStyleSheet("background-color: rgb(0,180,130);border-radius: 10px;border: 1px solid black;color: white;font-size: 16px;");

if(button==ui->inm) ui->inm->setStyleSheet("background-color: rgb(0,180,130);border-radius: 10px;border: 1px solid black;color: white;font-size: 16px;");

if(button==ui->ins) ui->ins->setStyleSheet("background-color: rgb(0,180,130);border-radius: 10px;border: 1px solid black;color: white;font-size: 16px;");

}

void Klass::**on\_bpoint\_clicked**()

{

ui->bpoint->setStyleSheet("background-color: rgb(0,220,220);border-radius: 10px;border: 1px solid black;font-size: 30px;color: white;");

QString s;

switch (stroka) {

case 1: if (!ui->hours->text().contains('.')){s= ui->hours->text();s+='.';ui->hours->setText(s);} break;

case 2: if (!ui->minuts->text().contains('.')){s= ui->minuts->text();s+='.';ui->minuts->setText(s);}break;

case 3: if (!ui->secunds->text().contains('.')){s= ui->secunds->text();s+='.';ui->secunds->setText(s);}break;

case 4: if (!ui->hours\_2->text().contains('.')){s= ui->hours\_2->text();s+='.';ui->hours\_2->setText(s);}break;

case 5: if (!ui->minuts\_2->text().contains('.')){s= ui->minuts\_2->text();s+='.';ui->minuts\_2->setText(s);}break;

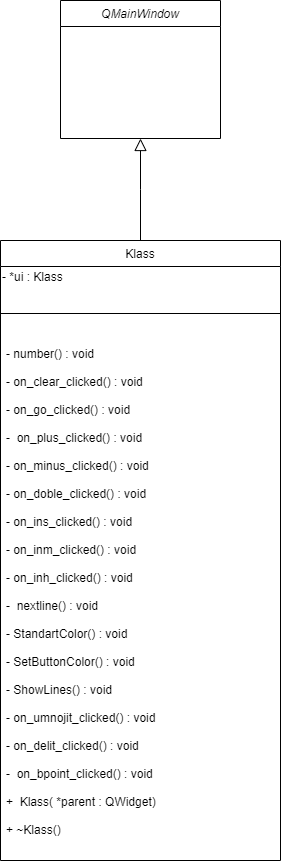
case 6: if (!ui->secunds\_2->text().contains('.')){s= ui->secunds\_2->text();s+='.';ui->secunds\_2->setText(s);}break;

case 8: if (!ui->consta->text().contains('.')){s= ui->consta->text();s+='.';ui->consta->setText(s);}break;

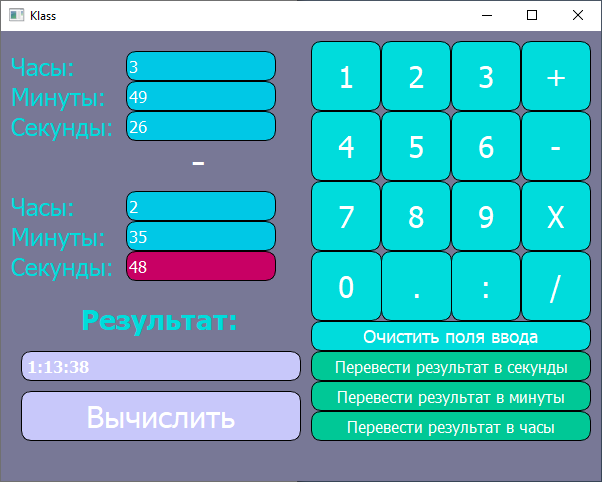
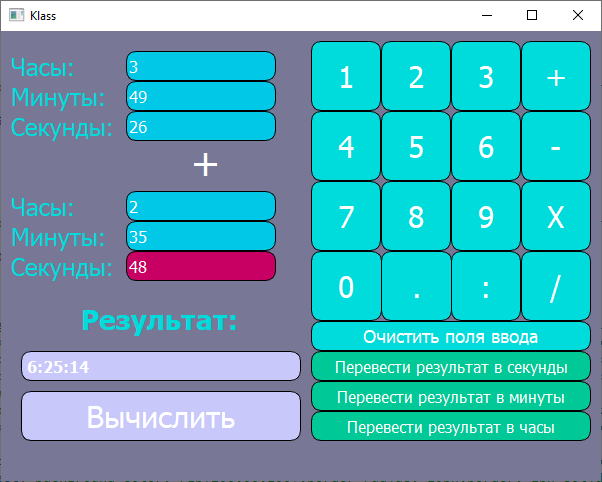
}

}

**Диаграмма классов**



**Результат работы**

Далее представлен внешний вид калькулятора: 

**Решение задачи коммивояжёра**

Разработано решение задачи коммивояжёра. Пользователь указывает количество городов, которые необходимо посетить коммивояжёру и расстояния между городами. В результате, если существует маршрут, соответствующий условию задачи, он изображается визуально и выводиться в виде текста.

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

void Vouage::**on\_gotab\_clicked**()

{

ui->gotab->setStyleSheet("color: rgb(0,250,0);background-color: rgb(150,150,150);border: 1px solid rgb(0,250,0);border-radius: 10px;");

ui->go->hide();

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

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

ui->build->hide();

Hider();

ris=0;

this->repaint();

QMessageBox::warning(this,"Предупреждение","Количество городов должно являться числом и иметь значение не менее 3 и не более 9! ");

}

else{

Hider();

ui->build->show();

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

Shower();

uzel = new Uzel[r];

xcenter=800;

ycenter=300;

ris=1;

this->repaint();

}

}

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

void Vouage::**on\_build\_clicked**()

{

ui->build->setStyleSheet("color: rgb(0,250,0);background-color: rgb(150,150,150);border: 1px solid rgb(0,250,0);border-radius: 10px;");

ui->go->hide();

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

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

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

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

ways[e][i]=ways[i][e];

}

}

ui->go->show();

ris=2;

this->repaint();

}

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

void Vouage::**on\_go\_clicked**()

{

ui->go->setStyleSheet("color: rgb(0,250,0);background-color: rgb(150,150,150);border: 1px solid rgb(0,250,0);border-radius: 10px;");

ui->go->hide();

QString searcher,buffer;

bool notway=0;

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++) {minway[i]=-1;way[i]=-1;}

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

for (int i=0;i<r;i++) if (minway[i]==-1) notway=1;

if (notway||mins==10000){

QMessageBox::warning(this,"Предупреждение","Не существует маршрута, позволяющего посетить все города, или невозможно посетить их лишь однажды. ");

}

else{

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

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

buffer+= " (или обратный ему).\n Длина пути: " +QString::number(mins);

ris=3;

this->repaint();

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

}

delete[] was;

delete[] way;

}

1. Функция, находящая кратчайший путь и его длину:

void Vouage::**Poisk**(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 (ways[i][e]) Poisk(ways[i][e],e);

b=1;

}

}

if(!b){

bool minus=0;

for (int k=0;k<r;k++) if (way[k]==-1) minus=1;

if (!minus&& leng<mins){

mins=leng;

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

minway[z]=way[z];

}

}

}

leng-=l;

s--;

was[i]=0;

way[s]=-1;

}

1. Функция, отображающая схему городов и путей между ними:

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

{

if (ris){

int otstup;

gradus/=r;

prirost=gradus;

gradus=0;

QPainter paint;

QPen pen;

QBrush brush;

QString stroka;

QFont font;

pen.setColor(Qt::blue);

pen.setWidth(7);

pen.setStyle(Qt::SolidLine);

font.setPointSize(20);

brush.setStyle(Qt::Dense3Pattern);

brush.setColor(Qt::white);

paint.begin(this);

paint.setPen(pen);

paint.setFont(font);

paint.setBrush(brush);

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

otstup=-10;

if (gradus>180) otstup=130;

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

uzel[i].x=xcenter+(200\*cos(gradus/57));

uzel[i].y=ycenter+(200\*-sin(gradus/57));

paint.drawEllipse(uzel[i].x,uzel[i].y,100,100);

paint.drawText(uzel[i].x+48,uzel[i].y+otstup,stroka);

gradus+=prirost;

}

if (ris>1){

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

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

if (ways[i][e]){

pen.setStyle(Qt::DashLine);

pen.setWidth(3);

paint.setPen(pen);

paint.drawLine(uzel[i].x+50,uzel[i].y+50,uzel[e].x+50,uzel[e].y+50);

}

}

}

}

if (ris==3){

pen.setStyle(Qt::SolidLine);

pen.setWidth(5);

pen.setColor(Qt::green);

paint.setPen(pen);

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

paint.drawLine(uzel[minway[i-1]].x+50,uzel[minway[i-1]].y+50,uzel[minway[i]].x+50,uzel[minway[i]].y+50);

}

}

paint.end();

}

}

**Полный код программы**

1. Код заголовочного файла:

#ifndef VOUAGE\_H

#define VOUAGE\_H

#include <QMainWindow>

#include <QPainter>

QT\_BEGIN\_NAMESPACE

namespace **Ui** { class **Vouage**; }

QT\_END\_NAMESPACE

class **Vouage** : public QMainWindow

{

Q\_OBJECT

public:

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

~***Vouage***();

private slots:

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

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

void **Poisk**(int l, int i);

void **Hider**();

void **Shower**();

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

void **on\_gotab\_pressed**();

void **on\_build\_pressed**();

void **on\_go\_pressed**();

private:

Ui::Vouage \*ui;

protected:

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

};

#endif // VOUAGE\_H

1. Код файла, содержащего функцию main():

#include "vouage.h"

#include <QApplication>

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

{

QApplication a(*argc*, argv);

Vouage w;

w.show();

return a.exec();

}

1. Код файла-источника, содержащего основной класс:

#include "vouage.h"

#include "ui\_vouage.h"

#include <cmath>

#include <QMessageBox>

bool notall=0;

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

double gradus=360;

int ways[10][10];

bool \*was;

int \*way, \*minway;

struct **Uzel**{

int x,y;

};

Uzel \*uzel;

Vouage::**Vouage**(QWidget \*parent)

: QMainWindow(parent)

, ui(new Ui::Vouage)

{

ui->setupUi(this);

ui->build->hide();

ui->go->hide();

Hider();

QPixmap back("C:/3.jpg");

back = back.scaled(this->size(), Qt::IgnoreAspectRatio);

QPalette palette;

palette.setBrush(QPalette::Background, back);

//palette.setColor(QPalette::Window,QColor(200,200,200));

this->setPalette(palette);

}

void Vouage::**Hider**(){

QString sear;

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

sear="clh" + QString::number(i);

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

}

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

sear="clv" + QString::number(i);

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

sear="d" + QString::number(i);

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

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

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

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

}

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

sear="l" + QString::number(i) + QString::number(e);

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

}

}

}

void Vouage::**Shower**(){

QString sear;

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

sear="clh" + QString::number(i);

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

}

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

sear="clv" + QString::number(i);

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

sear="d" + QString::number(i);

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

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

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

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

}

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

sear="l" + QString::number(i) + QString::number(e);

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

}

}

}

Vouage::~***Vouage***()

{

delete ui;

}

void Vouage::**Poisk**(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 (ways[i][e]) Poisk(ways[i][e],e);

b=1;

}

}

if(!b){

bool minus=0;

for (int k=0;k<r;k++) if (way[k]==-1) minus=1;

if (!minus&& leng<mins){

mins=leng;

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

minway[z]=way[z];

}

}

}

leng-=l;

s--;

was[i]=0;

way[s]=-1;

}

void Vouage::**on\_go\_clicked**()

{

ui->go->setStyleSheet("color: rgb(0,250,0);background-color: rgb(150,150,150);border: 1px solid rgb(0,250,0);border-radius: 10px;");

ui->go->hide();

QString searcher,buffer;

bool notway=0;

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++) {minway[i]=-1;way[i]=-1;}

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

for (int i=0;i<r;i++) if (minway[i]==-1) notway=1;

if (notway||mins==10000){

QMessageBox::warning(this,"Предупреждение","Не существует маршрута, позволяющего посетить все города, или невозможно посетить их лишь однажды. ");

}

else{

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

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

buffer+= " (или обратный ему).\n Длина пути: " +QString::number(mins);

ris=3;

this->repaint();

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

}

delete[] was;

delete[] way;

}

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

{

if (ris){

int otstup;

gradus/=r;

prirost=gradus;

gradus=0;

QPainter paint;

QPen pen;

QBrush brush;

QString stroka;

QFont font;

pen.setColor(Qt::blue);

pen.setWidth(7);

pen.setStyle(Qt::SolidLine);

font.setPointSize(20);

brush.setStyle(Qt::Dense3Pattern);

brush.setColor(Qt::white);

paint.begin(this);

paint.setPen(pen);

paint.setFont(font);

paint.setBrush(brush);

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

otstup=-10;

if (gradus>180) otstup=130;

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

uzel[i].x=xcenter+(200\*cos(gradus/57));

uzel[i].y=ycenter+(200\*-sin(gradus/57));

paint.drawEllipse(uzel[i].x,uzel[i].y,100,100);

paint.drawText(uzel[i].x+48,uzel[i].y+otstup,stroka);

gradus+=prirost;

}

if (ris>1){

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

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

if (ways[i][e]){

pen.setStyle(Qt::DashLine);

pen.setWidth(3);

paint.setPen(pen);

paint.drawLine(uzel[i].x+50,uzel[i].y+50,uzel[e].x+50,uzel[e].y+50);

}

}

}

}

if (ris==3){

pen.setStyle(Qt::SolidLine);

pen.setWidth(5);

pen.setColor(Qt::green);

paint.setPen(pen);

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

paint.drawLine(uzel[minway[i-1]].x+50,uzel[minway[i-1]].y+50,uzel[minway[i]].x+50,uzel[minway[i]].y+50);

}

}

paint.end();

}

}

void Vouage::**on\_gotab\_clicked**()

{

ui->gotab->setStyleSheet("color: rgb(0,250,0);background-color: rgb(150,150,150);border: 1px solid rgb(0,250,0);border-radius: 10px;");

ui->go->hide();

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

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

ui->build->hide();

Hider();

ris=0;

this->repaint();

QMessageBox::warning(this,"Предупреждение","Количество городов должно являться числом и иметь значение не менее 3 и не более 9! ");

}

else{

Hider();

ui->build->show();

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

Shower();

uzel = new Uzel[r];

xcenter=800;

ycenter=300;

ris=1;

this->repaint();

}

}

void Vouage::**on\_build\_clicked**()

{

ui->build->setStyleSheet("color: rgb(0,250,0);background-color: rgb(150,150,150);border: 1px solid rgb(0,250,0);border-radius: 10px;");

ui->go->hide();

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

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

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

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

ways[e][i]=ways[i][e];

}

}

ui->go->show();

ris=2;

this->repaint();

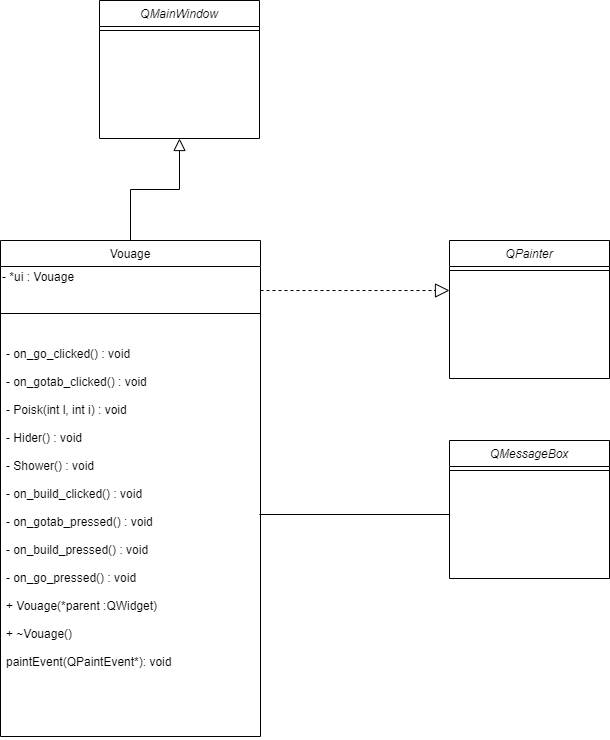
}

void Vouage::**on\_gotab\_pressed**(){ui->gotab->setStyleSheet("color: rgb(250,0,0);background-color: rgb(150,150,150);border: 1px solid rgb(250,0,0);border-radius: 10px;");}

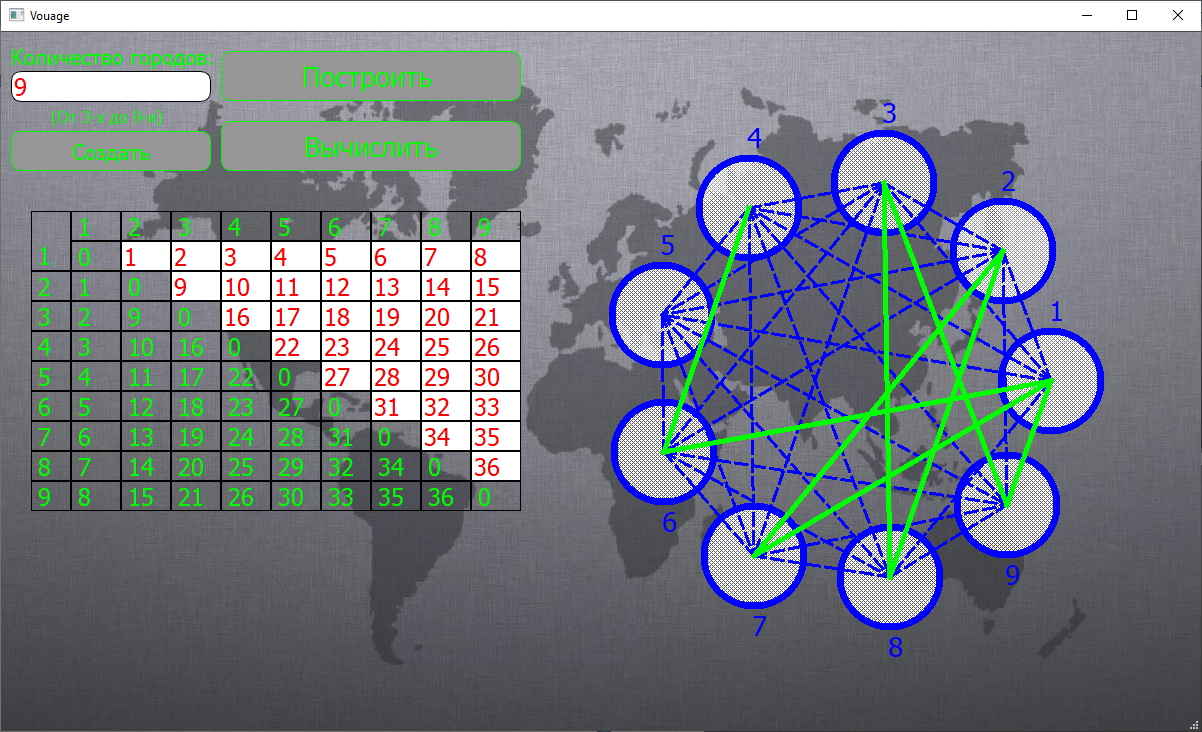
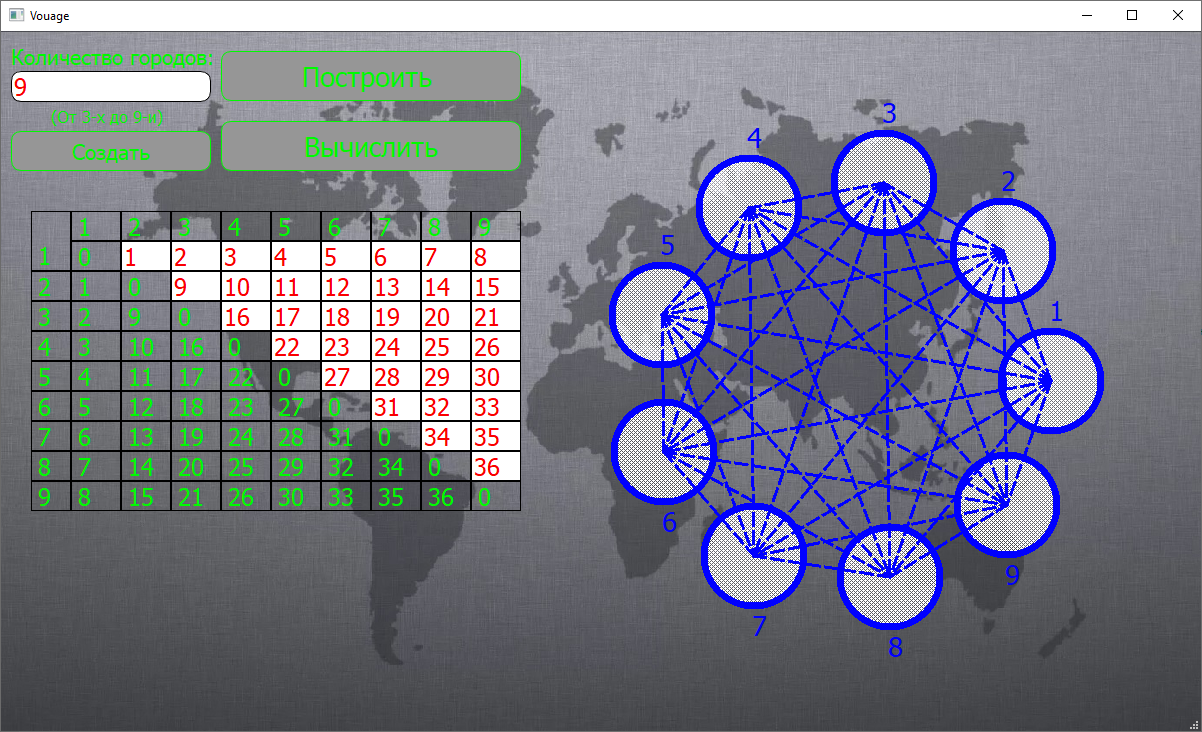
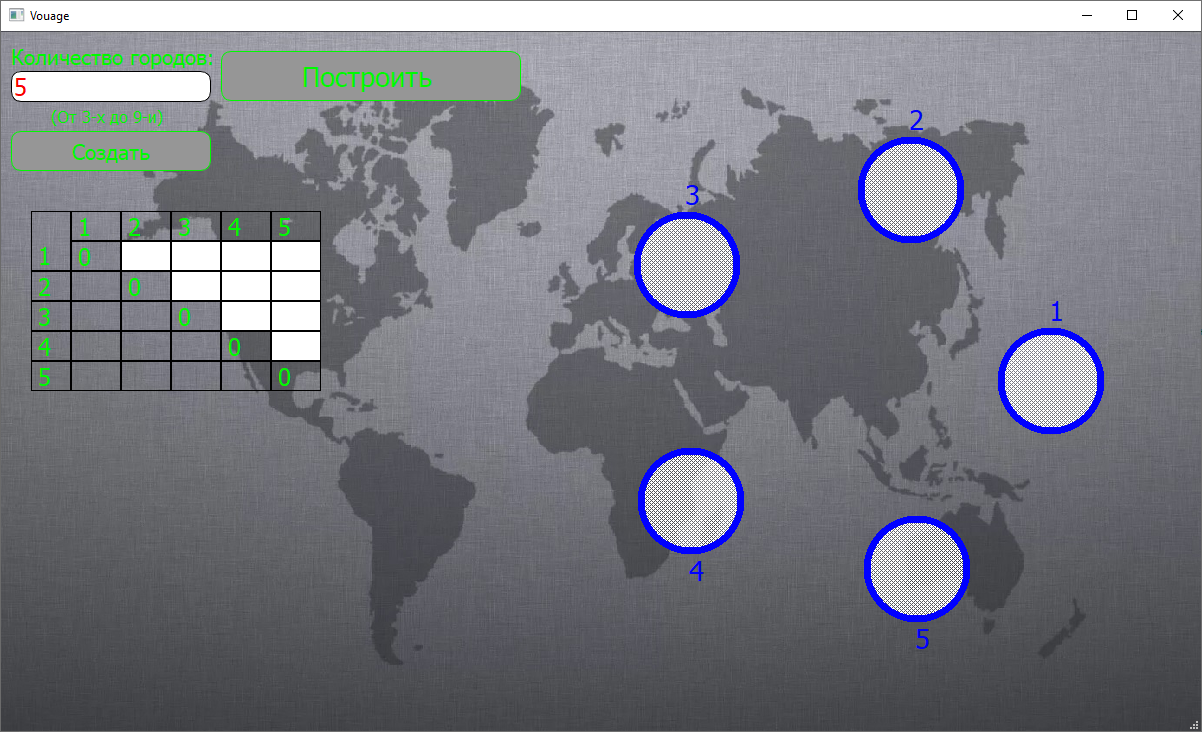
void Vouage::**on\_build\_pressed**(){ui->build->setStyleSheet("color: rgb(250,0,0);background-color: rgb(150,150,150);border: 1px solid rgb(250,0,0);border-radius: 10px;");}

void Vouage::**on\_go\_pressed**(){ui->go->setStyleSheet("color: rgb(250,0,0);background-color: rgb(150,150,150);border: 1px solid rgb(250,0,0);border-radius: 10px;");}

**Диаграмма классов**



**Результат работы**

Внешний вид окна программы: 

**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.