Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

**«Пермский национальный** **исследовательский политехнический университет»**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

направление подготовки: 09.03.01– «Информатика и вычислительная техника»

**Лабораторная работа**

**по дисциплине**

**«Теория алгоритмов и структуры данных»**

**на тему**

**«Задача Коммивояжера и АРМ Диспетчера трамвайного депо»**

Выполнил студент гр. ИВТ-23-1б

Южаков Федор Алексеевич

Проверил:

Доцент каф. ИТАС

Ольга Андреевна Полякова

(оценка) (подпись)

(дата)

г. Пермь, 2024

**Цель и задачи работы**

Целью данной работы реализация двух программ: программы для решения задачи коммивояжера, визуализации графа и программы для автоматизации рабочего места диспетчера трамвайного депо.

**UML диаграмма**

На рисунке 1 изображена диаграмма классов программы по решению и визуализации задачи Коммивояжера. На рисунке 2 изображена диаграмма классов программы по автоматизации рабочего места диспетчера трамвайного депо.

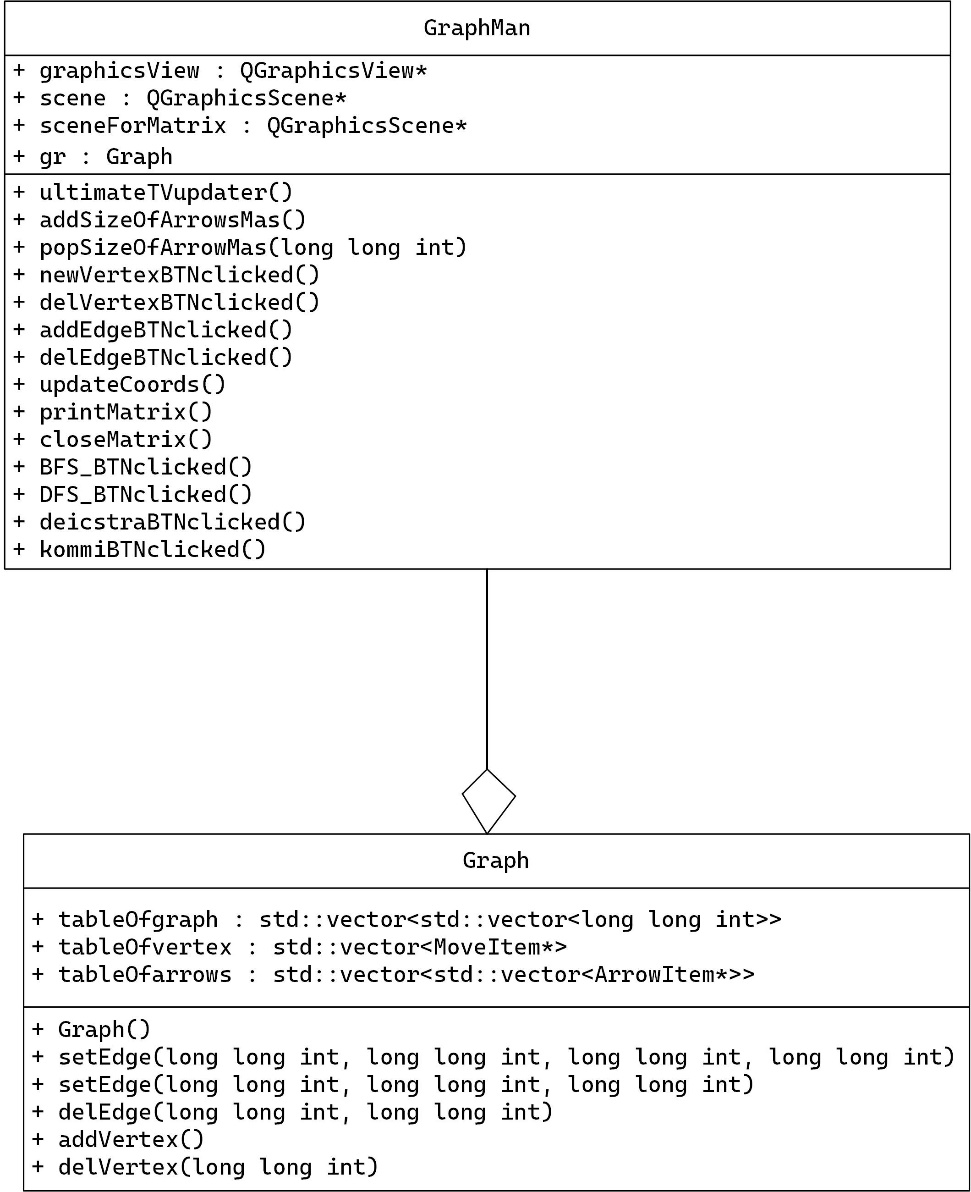


Рисунок 1

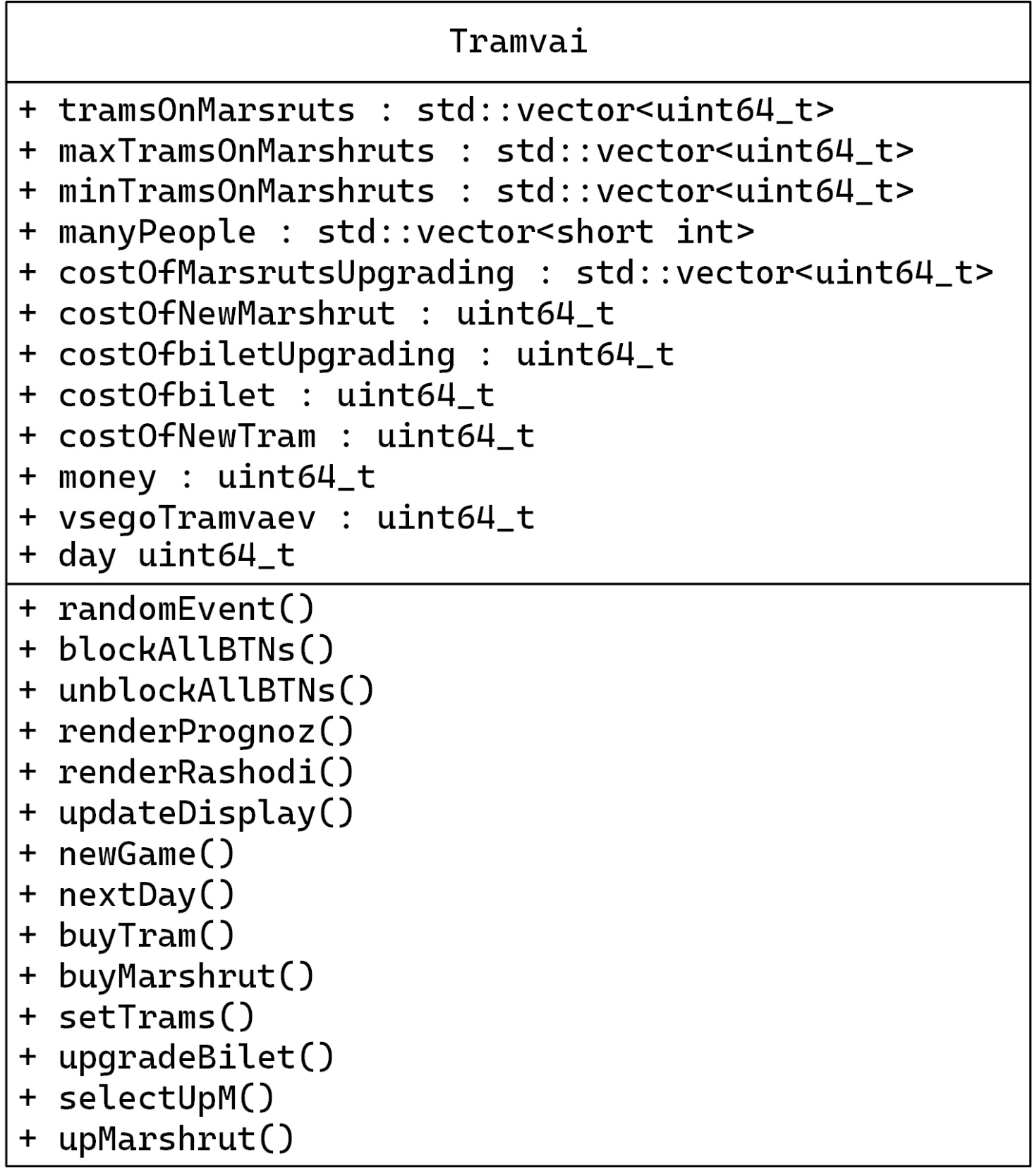


Рисунок 2

**Код программы по решению и визуализации задачи Коммивояжера**

**Файл ArrowItem.h**

#ifndef ARROWITEM\_H

#define ARROWITEM\_H

#include <QObject>

#include <QGraphicsItem>

#include <QPainter>

#include <QGraphicsSceneMouseEvent>

#include <QDebug>

#include <QCursor>

class ArrowItem : public QObject, public QGraphicsItem

{

Q\_OBJECT

public:

explicit ArrowItem(QObject \*parent = 0);

~ArrowItem();

long long int weight; // вес ребра

QPointF start; // координаты старта ребра

QPointF end; // координаты конца ребра

signals:

private:

QRectF boundingRect() const;

void paint(QPainter \*painter, const QStyleOptionGraphicsItem \*option, QWidget \*widget); // рисование ребра

public slots:

};

#endif // ARROWITEM\_H

**Файл ArrowItem.cpp**

#include "ArrowItem.h"

#include <cmath>

ArrowItem::ArrowItem(QObject \*parent) :

QObject(parent), QGraphicsItem() {}

ArrowItem::~ArrowItem() {}

QRectF ArrowItem::boundingRect() const // прямоугольник ограничение стрелки - нужен для правильного расширения и сужения сцены

{

return QRectF(start.x(),start.y(), (start.x()<end.x() ? abs(end.x()-start.x()) : -abs(end.x()-start.x())), (start.y()<end.y() ? -abs(end.y()-end.y()) : abs(end.y()-end.y())));

}

void ArrowItem::paint(QPainter \*painter, const QStyleOptionGraphicsItem \*option, QWidget \*widget)

{

qreal arrowSize = 10; // размер башки стрелочки

painter->setPen(Qt::black);

painter->setBrush(Qt::black);

painter->setFont(QFont("Montserrat Alternates", 15));

QLineF line(end, start);

double angle = std::atan2(-line.dy(), line.dx()); // угол между стрелкой и ОХ

// отодвигаем конец и начало стрелки от центра круга с радиусом 20

line.setP2(QPointF(start.x()-20\*cos(angle), start.y()+20\*sin(angle)));

line.setP1(QPointF(end.x()+20\*cos(angle), end.y()-20\*sin(angle)));

// точки для рисования башки стрелки

QPointF arrowP1 = line.p1() + QPointF(sin(angle + M\_PI / 3) \* arrowSize, cos(angle + M\_PI / 3) \* arrowSize);

QPointF arrowP2 = line.p1() + QPointF(sin(angle + M\_PI - M\_PI / 3) \* arrowSize, cos(angle + M\_PI - M\_PI / 3) \* arrowSize);

QPolygonF arrowHead;

arrowHead.clear();

arrowHead << line.p1() << arrowP1 << arrowP2;

painter->drawLine(line); // бошка стрелки

if(start != end) painter->drawPolygon(arrowHead); // обраотка петель - стрелка на петли не рисуется

painter->setPen(Qt::blue);

if(start != end) painter->drawText(QPointF(end.x()+50\*cos(angle), end.y()-50\*sin(angle)), QString::number(weight)); // текст отодвинут

else painter->drawText(QPointF(end.x(), end.y()-30), QString::number(weight)); // вес ребра

Q\_UNUSED(option);

Q\_UNUSED(widget);

}

**Файл moveitem.h**

#ifndef MOVEITEM\_H

#define MOVEITEM\_H

#include <QObject>

#include <QGraphicsItem>

#include <QPainter>

#include <QGraphicsSceneMouseEvent>

#include <QDebug>

#include <QCursor>

class MoveItem : public QObject, public QGraphicsItem

{

Q\_OBJECT

public:

explicit MoveItem(QObject \*parent = 0);

~MoveItem();

long long int nomer;

signals:

private:

QRectF boundingRect() const;

void paint(QPainter \*painter, const QStyleOptionGraphicsItem \*option, QWidget \*widget); // рисование вершины

void mouseMoveEvent(QGraphicsSceneMouseEvent \*event); // событие передвижения вершины

void mousePressEvent(QGraphicsSceneMouseEvent \*event); // событие нажатия мыши (смена курсора на руку)

void mouseReleaseEvent(QGraphicsSceneMouseEvent \*event); // событие отпускания мыши (смена курсора на стандартный)

public slots:

signals:

void coordinatesChanged(); // сигнал о смене координат вершины

};

#endif // MOVEITEM\_H

**Файл moveitem.cpp**

#include "moveitem.h"

MoveItem::MoveItem(QObject \*parent) :

QObject(parent), QGraphicsItem() {}

MoveItem::~MoveItem() {}

QRectF MoveItem::boundingRect() const

{

return QRectF (-20,-20,40,40);

}

void MoveItem::paint(QPainter \*painter, const QStyleOptionGraphicsItem \*option, QWidget \*widget)

{

painter->setPen(Qt::black);

painter->setBrush(Qt::white);

painter->setFont(QFont("Montserrat Alternates"));

painter->drawEllipse(-20,-20,40,40);

QString text(QString::number(nomer)); painter->drawText(-5, 5, text);

Q\_UNUSED(option);

Q\_UNUSED(widget);

}

void MoveItem::mouseMoveEvent(QGraphicsSceneMouseEvent \*event)

{

/\* Устанавливаем позицию графического элемента

\* в графической сцене, транслировав координаты

\* курсора внутри графического элемента

\* в координатную систему графической сцены

\* \*/

emit coordinatesChanged();

this->setPos(mapToScene(event->pos()));

}

void MoveItem::mousePressEvent(QGraphicsSceneMouseEvent \*event)

{

/\* При нажатии мышью на графический элемент

\* заменяем курсор на руку, которая держит этот элемента

\* \*/

this->setCursor(QCursor(Qt::ClosedHandCursor));\

Q\_UNUSED(event);

}

void MoveItem::mouseReleaseEvent(QGraphicsSceneMouseEvent \*event)

{

/\* При отпускании мышью элемента

\* заменяем на обычный курсор стрелку

\* \*/

this->setCursor(QCursor(Qt::ArrowCursor));

Q\_UNUSED(event);

}

**Файл graphman.h**

#ifndef GRAPHMAN\_H

#define GRAPHMAN\_H

#include <QMainWindow>

#include <QGraphicsView>

#include <QGraphicsScene>

#include <QGraphicsItemGroup>

#include "moveitem.h"

#include "ArrowItem.h"

QT\_BEGIN\_NAMESPACE

namespace Ui { class GraphMan; }

QT\_END\_NAMESPACE

class Graph // класс графа

{

public:

std::vector<std::vector<long long int>> tableOfgraph; // таблица смежности

std::vector<MoveItem\*> tableOfvertex; // вектор с указателями на графический объект - вершину

std::vector<std::vector<ArrowItem\*>> tableOfarrows; // таблица с указателями на стрелки (ребра графа)

Graph(); // конструктор

void setEdge(long long int, long long int, long long int, long long int); //метод установки/смены ребра сразу в две стороны

void setEdge(long long int, long long int, long long int); // метод установки/смены ребра в одну сторону

void delEdge(long long int, long long int); //метод удаления ребра в одну сторону

void addVertex(); // метод добавления вершины

void delVertex(long long int);

};

class GraphMan : public QMainWindow

{

Q\_OBJECT

public:

GraphMan(QWidget \*parent = nullptr);

~GraphMan();

QGraphicsView \*graphicsView; // телевизор

QGraphicsScene \*scene; // сцена для отображения графа

QGraphicsScene \*sceneForMatrix; //сцена для отображения таблицы смежности

Graph gr; // граф

void ultimateTVupdater(); // обновитель телевизора

void addSizeOfArrowsMas(); // увеличить размер таблицы со стрелками

void popSizeOfArrowMas(long long int); // убрать из таблицы со стрелками строку и столбец по индексу

public slots:

void newVertexBTNclicked(); // добавление вершины

void delVertexBTNclicked(); // удаление вершины

void addEdgeBTNclicked(); // добавление/изменение ребра

void delEdgeBTNclicked(); // удаление ребра

void updateCoords(); // обновление координат стрелок

void printMatrix(); // вывод таблицы смежности вместо графа

void closeMatrix(); // выход из режима таблицы смежности в режим графа

void BFS\_BTNclicked(); // вывод результатов обхода в ширину

void DFS\_BTNclicked(); // вывод результатов обхода в глубину

void deicstraBTNclicked(); // вывод кратчайшего пути от одной вершины до другой

void kommiBTNclicked(); // решение и вывод ответа на задачу Коммивояжёра

private:

Ui::GraphMan \*ui;

};

#endif // GRAPHMAN\_H

**Файл graphman.cpp**

#include "graphman.h"

#include "ui\_graphman.h"

#include "moveitem.h"

#include "ArrowItem.h"

#include <QPainter>

#include <QGraphicsView>

#include <QGraphicsScene>

#include <QRegExpValidator>

#include <vector>

#include <queue>

#include <algorithm>

static int randomBetween(int low, int high) // рандом между двумя числами

{

return ((rand()+rand()) % ((high + 1) - low) + low);

}

Graph::Graph() // конструктор графа

{

std::vector<long long int> tmp(1, 0);

tableOfgraph.push\_back(tmp);

tableOfvertex.push\_back(nullptr);

std::vector<ArrowItem\*> tmp2(1, nullptr);

tableOfarrows.push\_back(tmp2);

}

void Graph::setEdge(long long int vertexStart, long long int vertexFinish, long long int directWeight, long long int reverseWeight) // установка ребра от и до прямой вес и обратный

{

if (vertexStart == 0 || vertexFinish == 0) return;

if (vertexFinish > static\_cast<long long int>(tableOfgraph.size()) || vertexStart > static\_cast<long long int>(tableOfgraph.size())) return;

tableOfgraph[vertexStart][vertexFinish] = directWeight;

tableOfgraph[vertexFinish][vertexStart] = reverseWeight;

}

void Graph::setEdge(long long int vertexStart, long long int vertexFinish, long long int directWeight) // установка ребра от и до

{

if (vertexStart == 0 || vertexFinish == 0) return;

if (vertexFinish > static\_cast<long long int>(tableOfgraph.size()) || vertexStart > static\_cast<long long int>(tableOfgraph.size())) return;

tableOfgraph[vertexStart][vertexFinish] = directWeight;

}

void Graph::delEdge(long long int vertexStart, long long int vertexFinish) // удаление ребра от и до

{

if (vertexStart == 0 || vertexFinish == 0) return;

if (vertexFinish > static\_cast<long long int>(tableOfgraph.size()) || vertexStart > static\_cast<long long int>(tableOfgraph.size())) return;

setEdge(vertexStart, vertexFinish, -1);

}

void Graph::addVertex() // добавление вершины

{

std::vector<long long int> tmp(tableOfgraph.size(), -1);

tmp[0] = tableOfgraph.size();

tableOfgraph.push\_back(tmp);

for(unsigned long long int i = 0; i < tableOfgraph.size(); i++) // правильно увеличиваем таблицу смежности

{

if(i==0) tableOfgraph[i].push\_back(tableOfgraph.size()-1);

else tableOfgraph[i].push\_back(-1);

}

}

void Graph::delVertex(long long int vertexNumber) // удаляем вершину

{

if (tableOfgraph.size() < 2) return; // если нечего удалять

tableOfgraph.erase(std::next(tableOfgraph.begin(), vertexNumber)); // убираем строку вершины из таблицы смежности

for(unsigned long long int i = 0; i < tableOfgraph.size(); i++) tableOfgraph[i].erase(std::next(tableOfgraph[i].begin(), vertexNumber)); // убираем столб вершины из таблицы смежности

for(unsigned long long int i = vertexNumber; i < tableOfgraph.size(); i++) tableOfgraph[i][0] = i; // перенумеровываем вершины в таблице смежности и в векторе вершин

for(unsigned long long int i = vertexNumber; i < tableOfgraph[0].size(); i++) tableOfgraph[0][i] = i;

for(unsigned long long int i = vertexNumber; i < tableOfvertex.size(); i++) tableOfvertex[i]->nomer=i;

}

GraphMan::GraphMan(QWidget \*parent)

: QMainWindow(parent)

, ui(new Ui::GraphMan)

{

ui->setupUi(this);

// коннекты кнопок

connect(ui->addVertexBTN, &QPushButton::clicked, this, &GraphMan::newVertexBTNclicked);

connect(ui->delVertexBTN, &QPushButton::clicked, this, &GraphMan::delVertexBTNclicked);

connect(ui->addEdgeBTN, &QPushButton::clicked, this, &GraphMan::addEdgeBTNclicked);

connect(ui->delEdgeBTN, &QPushButton::clicked, this, &GraphMan::delEdgeBTNclicked);

connect(ui->kommiBTN, &QPushButton::clicked, this, &GraphMan::kommiBTNclicked);

connect(ui->tableBTN, &QPushButton::clicked, this, &GraphMan::printMatrix);

connect(ui->graphBTN, &QPushButton::clicked, this, &GraphMan::closeMatrix);

connect(ui->BFS\_BTN, &QPushButton::clicked, this, &GraphMan::BFS\_BTNclicked);

connect(ui->DFS\_BTN, &QPushButton::clicked, this, &GraphMan::DFS\_BTNclicked);

connect(ui->deicstraBTN, &QPushButton::clicked, this, &GraphMan::deicstraBTNclicked);

graphicsView = ui->graphTV; //коннекты телевизора и сцены

scene = new QGraphicsScene;

sceneForMatrix = new QGraphicsScene;

graphicsView -> setScene(scene);

ui->graphTV->setRenderHint(QPainter::Antialiasing); // Настраиваем рендер

ui->graphTV->setCacheMode(QGraphicsView::CacheBackground); // Кэш фона

ui->graphTV->setViewportUpdateMode(QGraphicsView::FullViewportUpdate); // режим обнавления телека - полный

QRegExp re("^[0-9]\*[1-9][0-9]\*$"); // целые положительные числа

QRegExpValidator\*v = new QRegExpValidator(re);

// устанавливаем ограничения и текст-подсказку на все поля ввода

ui->addEdgeStart->setPlaceholderText("Начало");

ui->addEdgeStart->setValidator(v);

ui->addEdgeFinish->setPlaceholderText("Конец");

ui->addEdgeFinish->setValidator(v);

ui->addEdgeWeight->setPlaceholderText("Прям. вес");

ui->addEdgeWeight->setValidator(v);

ui->addEdgeReverseWeight->setPlaceholderText("Обрат. вес");

ui->addEdgeReverseWeight->setValidator(v);

ui->delEdgeStart->setPlaceholderText("Начало");

ui->delEdgeStart->setValidator(v);

ui->delEdgeFinish->setPlaceholderText("Конец");

ui->delEdgeFinish->setValidator(v);

ui->delVertexInput->setPlaceholderText("Вершина №");

ui->delVertexInput->setValidator(v);

ui->BFS\_start->setPlaceholderText("Вершина начала");

ui->BFS\_start->setValidator(v);

ui->DFS\_start->setPlaceholderText("Вершина начала");

ui->DFS\_start->setValidator(v);

ui->deicstraStart->setPlaceholderText("Вершина начала");

ui->deicstraStart->setValidator(v);

ui->deicstraFinish->setPlaceholderText("Вершина конца");

ui->deicstraFinish->setValidator(v);

ui->kommiStart->setPlaceholderText("Вершина начала");

ui->kommiStart->setValidator(v);

ui->graphBTN->setHidden(true);

srand(time(NULL)); // обновление сида рандомного генератора

}

void GraphMan::ultimateTVupdater() // хороший такой обновитель дисплея

{

if (ui->graphTV->scene() == sceneForMatrix) printMatrix(); // если на экране таблица, то выводим ее

else

{

QRectF rect = scene->itemsBoundingRect(); // иначе увеличиваем/уменьшаем сцену, чтобы увидеть вершины за экраном и обновляем дисплей

rect.setHeight(rect.height()+1); // чинит некоторые визуальные баги, но, к сожалению не все

scene->setSceneRect(rect);

ui->graphTV->update();

}

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

}

void GraphMan::newVertexBTNclicked() // создание новой вершины - модифицирует все таблицы и векторы, какие нужно

{

MoveItem \*item = new MoveItem(); // Создаём графический элемент

item->nomer = gr.tableOfgraph.size();

item->setPos(randomBetween(0, 800), randomBetween(0,600)); // Устанавливаем случайную позицию элемента

scene->addItem(item); // Добавляем элемент на графическую сцену

gr.tableOfvertex.push\_back(item);

gr.addVertex();

addSizeOfArrowsMas();

ultimateTVupdater();

connect(item, &MoveItem::coordinatesChanged, this, &GraphMan::updateCoords); // для обновления стрелок, каждый круг испускает сигнал при движении, на который реагируем обновлением координат стрелок

}

void GraphMan::delVertexBTNclicked() // слот, выполняющийся при нажатии на кнопку УДАЛИТЬ ВЕРШИНУ

{

if(gr.tableOfvertex.size() < 2) return; // если вершин нет то выход

long long int vertexNumber = ui->delVertexInput->text().toLongLong(); // получаем номер вершины, которую нужно удалить

if (vertexNumber > static\_cast<int64\_t>(gr.tableOfvertex.size())-1 || vertexNumber < 1) return; //если номер неадекватен, то выходим

scene->removeItem(gr.tableOfvertex[vertexNumber]); // убираем вершину со сцены

gr.tableOfvertex.erase(std::next(gr.tableOfvertex.begin(), vertexNumber)); // убираем вершину из вектора

gr.delVertex(vertexNumber); // убираем из таблицы смежности и вектора вершин, перенумеровываем

popSizeOfArrowMas(vertexNumber); // убираем лишние стрелки

ui->delVertexInput->setText(""); // чистим поле ввода

ultimateTVupdater(); // обновляем дисплей

}

void GraphMan::addEdgeBTNclicked() // добавление/изменение ребра, разное в зависимости от того, какие веса введены, а какие нет

{

long long int start = ui->addEdgeStart->text().toLongLong(), finish = ui->addEdgeFinish->text().toLongLong();

if ((start > static\_cast<int64\_t>(gr.tableOfgraph.size())-1)||(finish > static\_cast<int64\_t>(gr.tableOfgraph.size())-1)||(finish < 1)||(start < 1)) return;

if ((ui->addEdgeWeight->text() == "") && (ui->addEdgeReverseWeight->text() == "")) return;

if ((ui->addEdgeWeight->text() != "") && (ui->addEdgeReverseWeight->text() != "")) // если ввели и прямой и обратный вес

{

long long int weight = ui->addEdgeWeight->text().toLongLong();

long long int reverseWeight = ui->addEdgeReverseWeight->text().toLongLong();

gr.setEdge(start, finish, weight, reverseWeight); // модифицируем таблицу смежности

if(gr.tableOfarrows[start][finish] == nullptr) //если нет ребра

{

ArrowItem\* arrow = new ArrowItem();

arrow->weight = weight;

arrow->start = gr.tableOfvertex[start]->pos();

arrow->end = gr.tableOfvertex[finish]->pos();

scene->addItem(arrow);

gr.tableOfarrows[start][finish] = arrow;

}

else // если есть стрелки

{

gr.tableOfarrows[start][finish]->weight = weight;

}

// повторяем для обратного веса

if(gr.tableOfarrows[finish][start] == nullptr)

{

ArrowItem\* arrow = new ArrowItem();

arrow->weight = reverseWeight;

arrow->start = gr.tableOfvertex[finish]->pos();

arrow->end = gr.tableOfvertex[start]->pos();

scene->addItem(arrow);

gr.tableOfarrows[finish][start] = arrow;

}

else

{

gr.tableOfarrows[finish][start]->weight = reverseWeight;

}

}

else if (ui->addEdgeWeight->text() != "") // если есть прямой вес, то все также, но только с прямым

{

long long int weight = ui->addEdgeWeight->text().toLongLong();

gr.setEdge(start, finish, weight);

if(gr.tableOfarrows[start][finish] == nullptr)

{

ArrowItem\* arrow = new ArrowItem();

arrow->weight = weight;

arrow->start = gr.tableOfvertex[start]->pos();

arrow->end = gr.tableOfvertex[finish]->pos();

scene->addItem(arrow);

gr.tableOfarrows[start][finish] = arrow;

}

else

{

gr.tableOfarrows[start][finish]->weight = weight;

}

}

else if (ui->addEdgeReverseWeight->text() != "") // с только обратным

{

long long int reverseWeight = ui->addEdgeReverseWeight->text().toLongLong();

gr.setEdge(finish, start, reverseWeight);

if(gr.tableOfarrows[finish][start] == nullptr)

{

ArrowItem\* arrow = new ArrowItem();

arrow->weight = reverseWeight;

arrow->start = gr.tableOfvertex[finish]->pos();

arrow->end = gr.tableOfvertex[start]->pos();

scene->addItem(arrow);

gr.tableOfarrows[finish][start] = arrow;

}

else

{

gr.tableOfarrows[finish][start]->weight = reverseWeight;

}

}

// чистим поля ввода

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

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

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

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

ultimateTVupdater();

}

void GraphMan::delEdgeBTNclicked() // удаление ребра

{

// проверки на дурака

long long int start = ui->delEdgeStart->text().toLongLong(), finish = ui->delEdgeFinish->text().toLongLong();

if ((start > static\_cast<int64\_t>(gr.tableOfgraph.size())-1) || (finish > static\_cast<int64\_t>(gr.tableOfgraph.size())-1)) return;

if (start == 0 || finish == 0) return;

if (gr.tableOfgraph[start][finish] == -1) return;

gr.delEdge(start, finish); // модифицируем таблицу смежности

scene->removeItem(gr.tableOfarrows[start][finish]); // убираем стрелку

gr.tableOfarrows[start][finish] = nullptr;

ui->delEdgeStart->setText(""); // чистим поля ввода

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

ultimateTVupdater();

}

void GraphMan::addSizeOfArrowsMas() // расширение таблицы со стрелками

{

std::vector<ArrowItem\*> tmp(gr.tableOfarrows.size(), nullptr);

gr.tableOfarrows.push\_back(tmp);

for (uint64\_t i = 0; i < gr.tableOfarrows.size(); i++) gr.tableOfarrows[i].push\_back(nullptr);

}

void GraphMan::popSizeOfArrowMas(long long int deletedVertex) // вычеркивание строки и столбца по индексу из таблицы со стрелками

{

for (uint64\_t i = 1; i < gr.tableOfarrows[deletedVertex].size(); i++) // идем по строке

{

if (gr.tableOfarrows[deletedVertex][i] != nullptr) // удаляем стрелки, если они есть

{

scene->removeItem(gr.tableOfarrows[deletedVertex][i]);

gr.tableOfarrows[deletedVertex][i]=nullptr;

}

}

gr.tableOfarrows.erase(std::next(gr.tableOfarrows.begin(), deletedVertex)); // стираем строку

for (uint64\_t i = 1; i < gr.tableOfarrows.size(); i++) // идем по столбцу

{

if (gr.tableOfarrows[i][deletedVertex] != nullptr)

{

scene->removeItem(gr.tableOfarrows[i][deletedVertex]); // если есть стрелка, удаляем

}

gr.tableOfarrows[i].erase(std::next(gr.tableOfarrows[i].begin(), deletedVertex)); // удаляем элемент из строки

}

}

void GraphMan::updateCoords() // обновляет координаты стрелок

{

for (uint64\_t i = 1; i < gr.tableOfarrows.size(); i++) // идем по таблице стрелок и меняем координаты начала и конца в соответствии с текущими координатами вершин

{

for(uint64\_t j = 1; j < gr.tableOfarrows[i].size(); j++)

{

if (gr.tableOfarrows[i][j] != nullptr)

{

gr.tableOfarrows[i][j]->start = gr.tableOfvertex[i]->pos();

gr.tableOfarrows[i][j]->end = gr.tableOfvertex[j]->pos();

}

}

}

ultimateTVupdater();

}

void GraphMan::printMatrix() // печатает таблицу смежности и сменяет сцену на сцену с таблицей

{

ui->graphBTN->setHidden(false); ui->tableBTN->setHidden(true); // меняем кнопки переключения режимов местами

QGraphicsScene\* del = sceneForMatrix; // чистим сцену

sceneForMatrix = new QGraphicsScene;

delete del;

unsigned long long int x = 0, y = 30;

for (uint64\_t i = 0; i < gr.tableOfgraph.size(); i++) // добавляем объекты на сцену, меняем цвет в соответствии с моим желанием, если таблица смежности и таблица стрелок не совпадают, то красим в красный (помогло при отладке)

{

x = 0;

for(uint64\_t j = 0; j < gr.tableOfgraph[i].size(); j++)

{

QColor bruh;

if ((gr.tableOfgraph[i][j] != -1 && gr.tableOfarrows[i][j] != nullptr)&&i!=0&&j!=0) bruh = QColor(27,235,16);

else if ((gr.tableOfgraph[i][j] == -1 && gr.tableOfarrows[i][j] == nullptr) && i!=0 && j!=0) bruh = QColor(201,250,255);

else if ((i == 0 || j == 0) && gr.tableOfarrows[i][j] == nullptr) bruh = QColor(233,255,44);

else bruh = QColor(255,0,0);

sceneForMatrix->addPolygon(QPolygonF(QRectF(x, y, 100, 30)), QPen(Qt::black), QBrush(bruh));

QGraphicsTextItem\* text = sceneForMatrix->addText(QString::number(gr.tableOfgraph[i][j]), QFont("Montserrat Alternates", 15, 2));

text->setPos(x+3, y-2);

x+=100;

}

y+=30;

}

ui->graphTV->setScene(sceneForMatrix); // меняем сцену и обновляем дисплей

ui->graphTV->update();

}

void GraphMan::closeMatrix() // выходим из режима матрицы

{

ui->graphBTN->setHidden(true); ui->tableBTN->setHidden(false);

ui->graphTV->setScene(scene);

ultimateTVupdater();

}

QString BFS(const std::vector<std::vector<long long int>>& adjacency\_matrix, long long int start) // обход в ширину

{

std::vector<bool> visited(adjacency\_matrix.size(), false);

std::queue<long long int> queue;

std::string path = "";

visited[start] = true;

queue.push(start);

while (!queue.empty())

{

long long int vertex = queue.front();

queue.pop();

// Добавляем вершину к пути

path += std::to\_string(vertex) + "->";

// Перебираем всех соседей

for (unsigned long long int i = 1; i < adjacency\_matrix[vertex].size(); ++i)

{

if ((adjacency\_matrix[vertex][i] !=-1 || adjacency\_matrix[i][vertex] !=-1) && !visited[i])

{

visited[i] = true;

queue.push(i);

}

}

}

// Удаляем последнюю стрелку "->" и выводим путь

if (!path.empty())

{

path.pop\_back();

path.pop\_back();

}

return QString(QString::fromStdString(path));

}

void GraphMan::BFS\_BTNclicked() // производит обход в ширину и выводит в поле вывода

{

int64\_t start = ui->BFS\_start->text().toLongLong();

if(start > static\_cast<int64\_t>(gr.tableOfvertex.size())-1 || start < 1) return;

ui->BFS\_start->setText("");

ui->output->setText(BFS(gr.tableOfgraph, start));

}

void DFS(const std::vector<std::vector<long long int>>& graph, long long int start, std::vector<bool>& visited, std::string& path) // основная часть функции рекурсивный обход в глубину

{

visited[start] = true;

path += std::to\_string(start) + "->";

for (unsigned long long int i = 1; i < graph[start].size(); i++)

{

if ((graph[start][i] !=-1 || graph[i][start] !=-1) && !visited[i])

{

DFS(graph, i, visited, path);

}

}

}

QString graphTraversal(const std::vector<std::vector<long long int>>& adjacencyMatrix, long long int start) // более удобная обертка для обхода в глубину, возвращающая путь

{

std::vector<bool> visited(adjacencyMatrix.size(), false);

std::string path;

DFS(adjacencyMatrix, start, visited, path);

// Удаляем последнюю стрелку "->" и возвращаем путь

if (!path.empty())

{

path.pop\_back();

path.pop\_back();

}

return QString(QString::fromStdString(path));

} void GraphMan::DFS\_BTNclicked() // производит обход в глубину и выводит в поле для вывода

{

int64\_t start = ui->DFS\_start->text().toLongLong();

if(start > static\_cast<int64\_t>(gr.tableOfvertex.size())-1 || start < 1) return;

ui->DFS\_start->setText("");

ui->output->setText(graphTraversal(gr.tableOfgraph, start));

}

// Функция реализует алгоритм Дейкстры для поиска кратчайшего пути

QString dijkstra(const std::vector<std::vector<long long int>>& graph, long long int start, long long int end)

{

// Инициализируем вектор расстояний как бесконечности

std::vector<long long int> dist(graph.size(), INT64\_MAX);

// Вектор для хранения предыдущих вершин в пути

std::vector<long long int> prev(graph.size(), -1);

// Очередь с приоритетами для выбора следующей вершины с минимальным расстоянием

std::priority\_queue<std::pair<long long int, long long int>, std::vector<std::pair<long long int, long long int>>, std::greater<std::pair<long long int, long long int>>> pq;

// Начальная вершина имеет расстояние 0 от себя

dist[start] = 0;

pq.push({0, start});

// Основной цикл алгоритма

while (!pq.empty())

{

// Получаем вершину с минимальным расстоянием

long long int distance = pq.top().first;

long long int current = pq.top().second;

pq.pop();

// Если текущее расстояние больше уже найденного, пропускаем вершину

if (distance > dist[current]) continue;

// Перебираем всех соседей текущей вершины

for (long long int i = 1; i < static\_cast<int64\_t>(graph[current].size()); i++)

{

long long int nextDistance = graph[current][i];

// Если сосед существует и обновление расстояния возможно

if (nextDistance >= 0 && dist[i] > dist[current] + nextDistance)

{

// Обновляем расстояние до соседа

dist[i] = dist[current] + nextDistance;

// Запоминаем текущую вершину как предыдущую для соседа

prev[i] = current;

// Добавляем соседа в очередь с приоритетами

pq.push({dist[i], i});

}

}

}

// Если путь до конечной вершины не найден

if (dist[end] == INT64\_MAX) return "Путь не найден";

// Восстанавливаем путь от конечной вершины до начальной

std::vector<long long int> path;

for (long long int at = end; at != -1; at = prev[at])

{

path.push\_back(at);

}

reverse(path.begin(), path.end());

// Строим строку с путем

std::string pathStr;

for (auto v : path)

{

pathStr += std::to\_string(v) + "->";

}

// Удаляем последнюю стрелку "->"

pathStr.pop\_back();

pathStr.pop\_back();

// Добавляем вес пути в конец строки

pathStr += " " + std::to\_string(dist[end]);

return QString(QString::fromStdString(pathStr));

}

void GraphMan::deicstraBTNclicked() // находит кратчайший путь от и до по алгоритму Дейкстры, выводит в поле для вывода

{

int64\_t start = ui->deicstraStart->text().toLongLong();

if(start > static\_cast<int64\_t>(gr.tableOfvertex.size())-1 || start < 1) return;

int64\_t finish = ui->deicstraFinish->text().toLongLong();

if(finish > static\_cast<int64\_t>(gr.tableOfvertex.size())-1 || finish < 1) return;

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

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

ui->output->setText(dijkstra(gr.tableOfgraph, start, finish));

}

// рекурсивная функция ветвей и границ

void branchAndBound(const std::vector<std::vector<long long>>& adjacencyMatrix, std::vector<long long>& path, std::vector<bool>& visited, long long& min\_path\_cost, std::vector<long long>& min\_path, long long& path\_cost, long long v, long long level, long long start)

{

long long n = adjacencyMatrix.size();

if (level == n && adjacencyMatrix[v][start] != -1)

{

// Проверка на наличие пути обратно в начальную вершину

long long current\_path\_cost = path\_cost + adjacencyMatrix[v][start];

if (current\_path\_cost < min\_path\_cost)

{

min\_path\_cost = current\_path\_cost;

min\_path = path;

min\_path.push\_back(start);

}

return;

}

for (long long i = 0; i < n; ++i)

{

if (!visited[i] && adjacencyMatrix[v][i] != -1)

{

visited[i] = true;

path[level] = i;

path\_cost += adjacencyMatrix[v][i];

if (path\_cost < min\_path\_cost)

{

branchAndBound(adjacencyMatrix, path, visited, min\_path\_cost, min\_path, path\_cost, i, level + 1, start);

}

// Возврат к предыдущему состоянию

path\_cost -= adjacencyMatrix[v][i];

visited[i] = false;

}

}

}

// обвес рекурсивной функции

QString findShortestPath(const std::vector<std::vector<long long>>& adjacencyMatrix, long long start)

{

long long n = adjacencyMatrix.size();

// Инициализация пути и минимального расстояния

std::vector<long long> min\_path;

long long min\_path\_cost = INT64\_MAX;

std::vector<bool> visited(n, false);

std::vector<long long> path(n);

long long path\_cost = 0;

// Начало поиска с начальной вершины

visited[start] = true;

path[0] = start;

branchAndBound(adjacencyMatrix, path, visited, min\_path\_cost, min\_path, path\_cost, start, 1, start);

// Формирование строки с кратчайшим путём

std::string result;

if (min\_path\_cost == INT64\_MAX)

{

result = "Задача не имеет решения";

}

else

{

for (unsigned long long i = 0; i < min\_path.size(); ++i)

{

if (i > 0)

{

result += "->";

}

result += std::to\_string(min\_path[i]+1); // поправка на индекс

}

result += " (" + std::to\_string(min\_path\_cost) + ")";

}

return QString(QString::fromStdString(result));

}

void GraphMan::kommiBTNclicked() // находит кратчайший путь коммивояжера и выводит его

{

int64\_t start = ui->kommiStart->text().toLongLong();

if(start > static\_cast<int64\_t>(gr.tableOfvertex.size())-1 || start < 1) return;

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

std::vector<std::vector<long long int>> newMatrix = gr.tableOfgraph; //копия матрицы смежности

for(uint64\_t i = 0; i < gr.tableOfgraph.size(); i++) // убираем петли

for(uint64\_t j = 0; j < gr.tableOfgraph.size(); j++)

if(i == j) newMatrix[i][j] = -1;

newMatrix.erase(std::next(newMatrix.begin(), 0)); //убираем шапку таблицы

for (uint64\_t i = 0; i < newMatrix.size(); i++) newMatrix[i].erase(std::next(newMatrix[i].begin(), 0)); // убираем нулевой столбец

ui->output->setText(findShortestPath(newMatrix, start-1)); // -1 поправка на другую индексацию в таблице без шапки

}

GraphMan::~GraphMan()

{

delete ui;

}

**Файл main.cpp**

#include "graphman.h"

#include <QApplication>

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

{

QApplication a(argc, argv);

GraphMan w;

w.show();

return a.exec();

}

**Файл graphman.ui**

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

<ui version="4.0">

<class>GraphMan</class>

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

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>1258</width>

<height>660</height>

</rect>

</property>

<property name="sizePolicy">

<sizepolicy hsizetype="Fixed" vsizetype="Fixed">

<horstretch>0</horstretch>

<verstretch>0</verstretch>

</sizepolicy>

</property>

<property name="minimumSize">

<size>

<width>1258</width>

<height>660</height>

</size>

</property>

<property name="maximumSize">

<size>

<width>1258</width>

<height>660</height>

</size>

</property>

<property name="windowTitle">

<string>Графы</string>

</property>

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

<widget class="QGraphicsView" name="graphTV">

<property name="geometry">

<rect>

<x>370</x>

<y>10</y>

<width>881</width>

<height>641</height>

</rect>

</property>

<property name="sizePolicy">

<sizepolicy hsizetype="Expanding" vsizetype="Expanding">

<horstretch>0</horstretch>

<verstretch>0</verstretch>

</sizepolicy>

</property>

</widget>

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

<property name="geometry">

<rect>

<x>10</x>

<y>60</y>

<width>351</width>

<height>41</height>

</rect>

</property>

<property name="font">

<font>

<family>Montserrat Alternates</family>

<pointsize>12</pointsize>

</font>

</property>

<property name="cursor">

<cursorShape>PointingHandCursor</cursorShape>

</property>

<property name="text">

<string>Добавить вершину</string>

</property>

</widget>

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

<property name="geometry">

<rect>

<x>10</x>

<y>110</y>

<width>241</width>

<height>41</height>

</rect>

</property>

<property name="font">

<font>

<family>Montserrat Alternates</family>

<pointsize>12</pointsize>

</font>

</property>

<property name="cursor">

<cursorShape>PointingHandCursor</cursorShape>

</property>

<property name="text">

<string>Удалить вершину</string>

</property>

</widget>

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

<property name="geometry">

<rect>

<x>10</x>

<y>160</y>

<width>351</width>

<height>41</height>

</rect>

</property>

<property name="font">

<font>

<family>Montserrat Alternates</family>

<pointsize>12</pointsize>

</font>

</property>

<property name="cursor">

<cursorShape>PointingHandCursor</cursorShape>

</property>

<property name="text">

<string>Добавить/изменить ребро</string>

</property>

</widget>

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

<property name="geometry">

<rect>

<x>10</x>

<y>260</y>

<width>351</width>

<height>41</height>

</rect>

</property>

<property name="font">

<font>

<family>Montserrat Alternates</family>

<pointsize>12</pointsize>

</font>

</property>

<property name="cursor">

<cursorShape>PointingHandCursor</cursorShape>

</property>

<property name="text">

<string>Удалить ребро</string>

</property>

</widget>

<widget class="QLineEdit" name="addEdgeStart">

<property name="geometry">

<rect>

<x>10</x>

<y>210</y>

<width>71</width>

<height>41</height>

</rect>

</property>

<property name="font">

<font>

<family>Montserrat Alternates</family>

</font>

</property>

<property name="text">

<string/>

</property>

<property name="maxLength">

<number>18</number>

</property>

</widget>

<widget class="QLineEdit" name="addEdgeFinish">

<property name="geometry">

<rect>

<x>90</x>

<y>210</y>

<width>71</width>

<height>41</height>

</rect>

</property>

<property name="font">

<font>

<family>Montserrat Alternates</family>

</font>

</property>

<property name="maxLength">

<number>18</number>

</property>

</widget>

<widget class="QLineEdit" name="addEdgeWeight">

<property name="geometry">

<rect>

<x>170</x>

<y>210</y>

<width>91</width>

<height>41</height>

</rect>

</property>

<property name="font">

<font>

<family>Montserrat Alternates</family>

</font>

</property>

<property name="maxLength">

<number>18</number>

</property>

</widget>

<widget class="QLineEdit" name="addEdgeReverseWeight">

<property name="geometry">

<rect>

<x>270</x>

<y>210</y>

<width>91</width>

<height>41</height>

</rect>

</property>

<property name="font">

<font>

<family>Montserrat Alternates</family>

</font>

</property>

<property name="maxLength">

<number>18</number>

</property>

</widget>

<widget class="QLineEdit" name="delEdgeStart">

<property name="geometry">

<rect>

<x>10</x>

<y>310</y>

<width>171</width>

<height>41</height>

</rect>

</property>

<property name="font">

<font>

<family>Montserrat Alternates</family>

</font>

</property>

<property name="maxLength">

<number>18</number>

</property>

</widget>

<widget class="QLineEdit" name="delEdgeFinish">

<property name="geometry">

<rect>

<x>190</x>

<y>310</y>

<width>171</width>

<height>41</height>

</rect>

</property>

<property name="font">

<font>

<family>Montserrat Alternates</family>

</font>

</property>

<property name="maxLength">

<number>18</number>

</property>

</widget>

<widget class="QLineEdit" name="delVertexInput">

<property name="geometry">

<rect>

<x>260</x>

<y>110</y>

<width>101</width>

<height>41</height>

</rect>

</property>

<property name="font">

<font>

<family>Montserrat Alternates</family>

</font>

</property>

<property name="text">

<string/>

</property>

<property name="maxLength">

<number>18</number>

</property>

</widget>

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

<property name="geometry">

<rect>

<x>10</x>

<y>560</y>

<width>171</width>

<height>41</height>

</rect>

</property>

<property name="font">

<font>

<family>Montserrat Alternates</family>

<pointsize>12</pointsize>

</font>

</property>

<property name="cursor">

<cursorShape>PointingHandCursor</cursorShape>

</property>

<property name="text">

<string>Коммивояжёр</string>

</property>

</widget>

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

<property name="geometry">

<rect>

<x>10</x>

<y>10</y>

<width>351</width>

<height>41</height>

</rect>

</property>

<property name="font">

<font>

<family>Montserrat Alternates</family>

<pointsize>12</pointsize>

</font>

</property>

<property name="cursor">

<cursorShape>PointingHandCursor</cursorShape>

</property>

<property name="text">

<string>Режим таблицы смежности</string>

</property>

</widget>

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

<property name="geometry">

<rect>

<x>10</x>

<y>10</y>

<width>351</width>

<height>41</height>

</rect>

</property>

<property name="font">

<font>

<family>Montserrat Alternates</family>

<pointsize>12</pointsize>

</font>

</property>

<property name="cursor">

<cursorShape>PointingHandCursor</cursorShape>

</property>

<property name="text">

<string>Режим графа</string>

</property>

</widget>

<widget class="QPushButton" name="BFS\_BTN">

<property name="geometry">

<rect>

<x>10</x>

<y>360</y>

<width>171</width>

<height>41</height>

</rect>

</property>

<property name="font">

<font>

<family>Montserrat Alternates</family>

<pointsize>12</pointsize>

</font>

</property>

<property name="cursor">

<cursorShape>PointingHandCursor</cursorShape>

</property>

<property name="text">

<string>Обход в ширину</string>

</property>

</widget>

<widget class="QLineEdit" name="BFS\_start">

<property name="geometry">

<rect>

<x>190</x>

<y>360</y>

<width>171</width>

<height>41</height>

</rect>

</property>

<property name="font">

<font>

<family>Montserrat Alternates</family>

</font>

</property>

<property name="maxLength">

<number>18</number>

</property>

</widget>

<widget class="QPushButton" name="DFS\_BTN">

<property name="geometry">

<rect>

<x>10</x>

<y>410</y>

<width>171</width>

<height>41</height>

</rect>

</property>

<property name="font">

<font>

<family>Montserrat Alternates</family>

<pointsize>12</pointsize>

</font>

</property>

<property name="cursor">

<cursorShape>PointingHandCursor</cursorShape>

</property>

<property name="text">

<string>Обход в глубину</string>

</property>

</widget>

<widget class="QLineEdit" name="DFS\_start">

<property name="geometry">

<rect>

<x>190</x>

<y>410</y>

<width>171</width>

<height>41</height>

</rect>

</property>

<property name="font">

<font>

<family>Montserrat Alternates</family>

</font>

</property>

<property name="maxLength">

<number>18</number>

</property>

</widget>

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

<property name="geometry">

<rect>

<x>10</x>

<y>460</y>

<width>351</width>

<height>41</height>

</rect>

</property>

<property name="font">

<font>

<family>Montserrat Alternates</family>

<pointsize>12</pointsize>

</font>

</property>

<property name="cursor">

<cursorShape>PointingHandCursor</cursorShape>

</property>

<property name="text">

<string>Алгоритм Дейкстры</string>

</property>

</widget>

<widget class="QLineEdit" name="deicstraStart">

<property name="geometry">

<rect>

<x>10</x>

<y>510</y>

<width>171</width>

<height>41</height>

</rect>

</property>

<property name="font">

<font>

<family>Montserrat Alternates</family>

</font>

</property>

<property name="maxLength">

<number>18</number>

</property>

</widget>

<widget class="QLineEdit" name="deicstraFinish">

<property name="geometry">

<rect>

<x>190</x>

<y>510</y>

<width>171</width>

<height>41</height>

</rect>

</property>

<property name="font">

<font>

<family>Montserrat Alternates</family>

</font>

</property>

<property name="maxLength">

<number>18</number>

</property>

</widget>

<widget class="QTextBrowser" name="output">

<property name="geometry">

<rect>

<x>10</x>

<y>610</y>

<width>351</width>

<height>41</height>

</rect>

</property>

<property name="font">

<font>

<family>Montserrat Alternates</family>

<pointsize>12</pointsize>

</font>

</property>

</widget>

<widget class="QLineEdit" name="kommiStart">

<property name="geometry">

<rect>

<x>190</x>

<y>560</y>

<width>171</width>

<height>41</height>

</rect>

</property>

<property name="font">

<font>

<family>Montserrat Alternates</family>

</font>

</property>

<property name="maxLength">

<number>18</number>

</property>

</widget>

</widget>

</widget>

<resources/>

<connections/>

</ui>

**Файл graphman.pro**

QT += core gui

greaterThan(QT\_MAJOR\_VERSION, 4): QT += widgets

CONFIG += c++11

# You can make your code fail to compile if it uses deprecated APIs.

# In order to do so, uncomment the following line.

#DEFINES += QT\_DISABLE\_DEPRECATED\_BEFORE=0x060000 # disables all the APIs deprecated before Qt 6.0.0

SOURCES += \

ArrowItem.cpp \

main.cpp \

graphman.cpp \

moveitem.cpp

HEADERS += \

ArrowItem.h \

graphman.h \

moveitem.h

FORMS += \

graphman.ui

# Default rules for deployment.

qnx: target.path = /tmp/$${TARGET}/bin

else: unix:!android: target.path = /opt/$${TARGET}/bin

!isEmpty(target.path): INSTALLS += target

**Код программы АРМ диспетчера трамвайного депо**

**Файл tramvai.h**

#ifndef TRAMVAI\_H

#define TRAMVAI\_H

#include <QMainWindow>

QT\_BEGIN\_NAMESPACE

namespace Ui { class Tramvai; }

QT\_END\_NAMESPACE

class Tramvai : public QMainWindow

{

Q\_OBJECT

public:

Tramvai(QWidget \*parent = nullptr);

~Tramvai();

std::vector<uint64\_t> tramsOnMarsruts; // индексы - номера маршрутов

std::vector<uint64\_t> maxTramsOnMarshruts; // по индексу мршрута - максимальное кол-во трамваев

std::vector<uint64\_t> minTramsOnMarshruts; // по индексу маршрута - минимально кол-во трамваев

std::vector<short int> manyPeople; // по индексам - сколько людей -1 мало 0 средне 1 много

std::vector<uint64\_t> costOfMarsrutsUpgrading; // цена улучшения маршрута

uint64\_t costOfNewMarshrut; // цена нового маршрута

uint64\_t costOfbiletUpgrading; // стоимость повышения цены билета

uint64\_t costOfbilet; // цена одного билета

uint64\_t costOfNewTram; // цена нового трамвая

uint64\_t money; // ваше состояние

uint64\_t vsegoTramvaev; // всего трамваев

uint64\_t day; // текущий день

void randomEvent(); // рандомное распределение людей

void blockAllBTNs(); // блокировка всех кнопок

void unblockAllBTNs(); // разблокировка всех кнопок

void renderPrognoz(); // вывод распределения трамваев и прогнозов на экран

void renderRashodi(); // вывод сводки расходы/доходы

void updateDisplay(); // обновление всех ценников и прочих полей с текстом

public slots:

void newGame(); // начало игры

void nextDay(); // переход к следующему дню

void buyTram(); // купить трамвай

void buyMarshrut(); // купить маршрут

void setTrams(); // установить количество трамваев на маршрут

void upgradeBilet(); // Увеличить стоимость билета

void selectUpM(); // автомотически показывает цену улучшения выбранного маршрута

void upMarshrut(); // улучшает маршрут

private:

Ui::Tramvai \*ui;

};

#endif // TRAMVAI\_H

**Файл tramvai.cpp**

#include "tramvai.h"

#include "ui\_tramvai.h"

#include <QRegExpValidator>

struct otchet // вспомогательная структоура для хранения отчета за день

{

uint64\_t plusTram = 0; // доход

uint64\_t minusElectricity = 0; // расход на электричество

uint64\_t minusRepair = 0; // расход на ремонт

uint64\_t repairs = 0; // количество ремонтов

uint64\_t minusCrash = 0; // расходы на аварии

uint64\_t crashes = 0; // количество аварий

uint64\_t repairPath = 0; // расходы на ремонт трамвайных путей

void clear() // метод отчистки

{

plusTram = 0;

minusElectricity = 0;

minusRepair = 0;

repairs = 0;

minusCrash = 0;

crashes = 0;

repairPath = 0;

}

};

otchet svodka; // для хранения отчета

QString prognoz = ""; // для хранения прогноза на день

static bool randomByPercents(int percent) // рандомно по проценту

{

if ((rand() % (100 - 0 + 1) + 0) <= percent) return true;

else return false;

}

static int randomBetween(int low, int high) // рандом между двумя числами

{

return ((rand()\*rand()) % ((high + 1) - low) + low);

}

Tramvai::Tramvai(QWidget \*parent)

: QMainWindow(parent)

, ui(new Ui::Tramvai)

{

ui->setupUi(this);

// коннекты слотов и кнопок

connect(ui->newGameBTN, &QPushButton::clicked, this, &Tramvai::newGame);

connect(ui->buyTramBTN, &QPushButton::clicked, this, &Tramvai::buyTram);

connect(ui->newMarshrutBTN, &QPushButton::clicked, this, &Tramvai::buyMarshrut);

connect(ui->setTramvaiBTN, &QPushButton::clicked, this, &Tramvai::setTrams);

connect(ui->ubgradeBiletsBTN, &QPushButton::clicked, this, &Tramvai::upgradeBilet);

connect(ui->upMarshrutInput, &QLineEdit::textChanged, this, &Tramvai::selectUpM);

connect(ui->upMarshrutBTN, &QPushButton::clicked, this, &Tramvai::upMarshrut);

connect(ui->newDayBTN, &QPushButton::clicked, this, &Tramvai::nextDay);

// установка текста подсказки в поля ввода

ui->setTramKolvoInput->setPlaceholderText("Кол-во трамваев");

ui->marshrutInput->setPlaceholderText("Маршрут");

ui->upMarshrutInput->setPlaceholderText("Маршрут");

// сначала доступна только кнопка НАЧАТЬ

blockAllBTNs();

ui->newGameBTN->setEnabled(true);

// устанавливаем ограничение ввода

QRegExp re("^[0-9]\*[1-9][0-9]\*$");

QRegExpValidator\*v = new QRegExpValidator(re);

ui->marshrutInput->setValidator(v);

ui->setTramKolvoInput->setValidator(v);

ui->upMarshrutInput->setValidator(v);

}

void Tramvai::blockAllBTNs() // блокировка всех кнопок

{

ui->buyTramBTN->setEnabled(false);

ui->newDayBTN->setEnabled(false);

ui->newGameBTN->setEnabled(false);

ui->newMarshrutBTN->setEnabled(false);

ui->setTramvaiBTN->setEnabled(false);

ui->ubgradeBiletsBTN->setEnabled(false);

ui->upMarshrutBTN->setEnabled(false);

}

void Tramvai::unblockAllBTNs() // разблокировка всех кнопок

{

ui->buyTramBTN->setEnabled(true);

ui->newDayBTN->setEnabled(true);

ui->newGameBTN->setEnabled(true);

ui->newMarshrutBTN->setEnabled(true);

ui->setTramvaiBTN->setEnabled(true);

ui->ubgradeBiletsBTN->setEnabled(true);

ui->upMarshrutBTN->setEnabled(true);

}

void Tramvai::renderPrognoz() // вывод прогнозов и распределения трамваев по маршрутам

{

prognoz = "";

prognoz.push\_back("День №" + QString::number(day) + "\nВсего трамваев: " + QString::number(vsegoTramvaev) + "\n\n");

for (uint64\_t i = 0; i < tramsOnMarsruts.size(); i++)

{

prognoz.push\_back("Маршрут №"+QString::number(i)+ ": " + "\nТрамваи: " + QString::number(tramsOnMarsruts[i]) + "\nМинимально: " + QString::number(minTramsOnMarshruts[i]) + "\nМаксимально: " + QString::number(maxTramsOnMarshruts[i]) + "\nЛюди: " + (manyPeople[i] == 1 ? "много" : (manyPeople[i] == -1 ? "мало" : "средне")) + "\n\n");

}

uint64\_t nonUsed = vsegoTramvaev;

for(uint64\_t i = 0; i < tramsOnMarsruts.size(); i++)

{

nonUsed -= tramsOnMarsruts[i];

}

prognoz.push\_back("Неиспользовано трамваев: " + QString::number(nonUsed));

ui->marshrutiANDtrams->setText(prognoz);

}

void Tramvai::renderRashodi() // вывод сводки за предыдущий день

{

QString vivod = 0;

// конструирование строки: расходы доходы аварии ремонты

vivod.push\_back("Трамваями заработано: " + QString::number(svodka.plusTram) + " ₽" + "\n");

vivod.push\_back("Расходы на электричество: " + QString::number(svodka.minusElectricity) + " ₽" + "\n");

vivod.push\_back("Расходы на ремонт трамваев: " + QString::number(svodka.minusRepair) + " ₽" + "\n");

vivod.push\_back("Трамваев отремонтировано: " + QString::number(svodka.repairs) + "\n");

vivod.push\_back("Расходы на аварии: " + QString::number(svodka.minusCrash) + " ₽" + "\n");

vivod.push\_back("Всего аварий: " + QString::number(svodka.crashes) + "\n");

if (svodka.repairPath != 0) vivod.push\_back("На ремонт путей потрачено: " + QString::number(svodka.repairPath) + " ₽" + "\n");

// Считаем деньги

uint64\_t vsegoVMinus = svodka.minusElectricity + svodka.minusRepair + svodka.minusCrash + svodka.repairPath; // все расходы за день

vivod.push\_back("---------------------------------------------\nИТОГО расходов: " + QString::number(vsegoVMinus) + " ₽" + "\n");

if (vsegoVMinus > svodka.plusTram) // ушли в минус

{

if (money < vsegoVMinus - svodka.plusTram) // не смогли покрыть расходы

{

vivod.push\_back("\nВЫ БАНКРОТ!!!!!!!! НО вы можете начать заного.");

money = 0;

blockAllBTNs();

ui->newGameBTN->setEnabled(true);

}

else

{

money -= (vsegoVMinus - svodka.plusTram); // смогли покрыть расходы

vivod.push\_back("\nВаше состояние изменилось на: -" + QString::number(vsegoVMinus - svodka.plusTram) + " .");

}

}

else if (vsegoVMinus < svodka.plusTram) // получили прибыль

{

money += (svodka.plusTram - vsegoVMinus);

vivod.push\_back("\nВаше состояние изменилось на: +" + QString::number(svodka.plusTram - vsegoVMinus) + " ₽.");

}

else vivod.push\_back("\nВаше состояние НЕ изменилось."); // расходы и доходы равны

ui->rashodi->setText(vivod);

}

void Tramvai::updateDisplay() // обновление всех полей с текстом

{

ui->biletUPcost->setText(QString::number(costOfbiletUpgrading)+" ₽");

ui->money->setText(QString::number(money)+" ₽");

ui->newMarshrutCost->setText(QString::number(costOfNewMarshrut)+" ₽");

ui->tramCost->setText(QString::number(costOfNewTram)+" ₽");

ui->upMarshrutCost->setText("Введите маршрут");

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

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

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

renderPrognoz();

}

void Tramvai::newGame() // начало игры

{

// чистим векторы от прошлой игры

tramsOnMarsruts.clear();

maxTramsOnMarshruts.clear();

minTramsOnMarshruts.clear();

costOfMarsrutsUpgrading.clear();

manyPeople.clear();

// устанавливаем начальные значения

tramsOnMarsruts.push\_back(1);

maxTramsOnMarshruts.push\_back(3);

minTramsOnMarshruts.push\_back(1);

costOfMarsrutsUpgrading.push\_back(10000000);

manyPeople.push\_back(0);

costOfNewMarshrut = 100000000;

costOfbiletUpgrading = 100000000;

costOfbilet = 35;

costOfNewTram = 1000000;

money = 100000000;

vsegoTramvaev = 1;

day = 1;

// стираем остатки от предыдущей игры

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

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

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

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

ui->rashodi->setText("Проведите хоть один день, чтобы узнать сводку за прошлый день");

unblockAllBTNs(); // разблокируем все кнопки

updateDisplay(); // обновляем дисплей

srand(time(NULL)); // меняем сид рандома

randomEvent(); // начинаем игру

}

void Tramvai::randomEvent() // распределяет людей по трамваям

{

for (uint64\_t i = 0; i < manyPeople.size(); i++)

{

if (randomByPercents(20)) manyPeople[i] = 1; // много людей

else if (randomByPercents(20)) manyPeople[i] = -1; // мало людей

else manyPeople[i] = 0; // обычное кол-во людей

}

renderPrognoz();

}

void Tramvai::nextDay() // переход к следующему дню

{

for (uint64\_t i = 0; i < tramsOnMarsruts.size(); i++) // проверка на выполненность условий

{

if (tramsOnMarsruts[i] < minTramsOnMarshruts[i] || tramsOnMarsruts[i] > maxTramsOnMarshruts[i])

{

updateDisplay();

ui->marshrutiANDtrams->setText(prognoz+"\nНе все условия выполнены!!!\nЕсли вы не можете выполнить условия, нажмите НАЧАТЬ ЗАНОГО.");

return;

}

}

// подсчет трамваев на маршрутах (они приносят деньги, тратят электричество и могут попадать в аварии)

uint64\_t vsegoTramOnMarsh = 0;

for (uint64\_t i = 0; i < tramsOnMarsruts.size(); i++)

{

vsegoTramOnMarsh += tramsOnMarsruts[i];

}

svodka.clear(); // удаляем старый отчет

// подсчитываем выручку

for(uint64\_t i = 0; i < tramsOnMarsruts.size(); i++)

{

if (manyPeople[i] == 0) svodka.plusTram += tramsOnMarsruts[i] \* costOfbilet \* randomBetween(1000, 1500); // зароботок трамваев там где средее кол-во людей

else if (manyPeople[i] == 1) svodka.plusTram += tramsOnMarsruts[i] \* costOfbilet \* randomBetween(1500, 3000); // зароботок трамваев там где большое кол-во людей

else if (manyPeople[i] == -1) svodka.plusTram += tramsOnMarsruts[i] \* costOfbilet \* randomBetween(500, 1000); // зароботок трамваев там где малое кол-во людей

}

svodka.minusElectricity = vsegoTramOnMarsh \* randomBetween(1000, 2000); // подсчет потраченного электричества

// подсчет денег на ремонт и лечение сбитых людей и моральный ущерб

for (uint64\_t i = 0; i < vsegoTramOnMarsh; i++)

{

if (randomByPercents(4))

{

svodka.repairs++;

svodka.minusRepair += randomBetween(1000, 1000000); // трамвай могли просто поцарапать, а могли и сломать почти полностью

}

if(randomByPercents(1)&&randomByPercents(1))

{

svodka.crashes++;

svodka.minusCrash += randomBetween(100, 5000000); // человека могло сбить, а могло и просто зажать дверью его сумку, оттуда и разные суммы

}

}

if (day % 100 == 0) // каждый 100 - й день ремонтируем трамвайные пути

{

for(uint64\_t i = 0; i < tramsOnMarsruts.size(); i++)

{

svodka.repairPath += 1000000 \* maxTramsOnMarshruts[i]; // чем выше уровень маршрута, тем дороже его ремонтировать

}

}

renderRashodi(); // выводим сводку

day++; // наступает новый день

randomEvent();

updateDisplay();

}

void Tramvai::buyTram() // купить трамвай

{

if (costOfNewTram <= money) // должно хватать денег

{

vsegoTramvaev++;

money -= costOfNewTram;

costOfNewTram += costOfNewTram / 10; // типа инфляция

}

updateDisplay();

}

void Tramvai::buyMarshrut() // покупка маршрута

{

if (costOfNewMarshrut <= money) // должно хватать денег

{

tramsOnMarsruts.push\_back(0);

minTramsOnMarshruts.push\_back(1);

maxTramsOnMarshruts.push\_back(3);

costOfMarsrutsUpgrading.push\_back(10000000);

manyPeople.push\_back(rand() % (1 + 1 + 1) -1);

money -= costOfNewMarshrut;

costOfNewMarshrut += costOfNewMarshrut / 10; // типа инфляция

updateDisplay();

}

}

void Tramvai::setTrams() // устанавить трамваи на маршрут

{

uint64\_t freeTrams = vsegoTramvaev;

for(uint64\_t i = 0; i < tramsOnMarsruts.size(); i++)

{

freeTrams -= tramsOnMarsruts[i];

}

if ((ui->setTramKolvoInput->text().toULongLong() <= freeTrams) && (ui->marshrutInput->text().toULongLong() < tramsOnMarsruts.size()))

{ // нельзя установить больше трамваев, чем свободно

tramsOnMarsruts[ui->marshrutInput->text().toULongLong()] = ui->setTramKolvoInput->text().toULongLong();

updateDisplay();

}

}

void Tramvai::upgradeBilet() // повысить цену на билет

{

if (costOfbiletUpgrading <= money) // на процедуру должно хватать денег

{

costOfbilet += 5;

money -= costOfbiletUpgrading;

costOfbiletUpgrading += costOfbiletUpgrading / 10 \* 3; // чтобы часто не повышали цены

updateDisplay();

}

}

void Tramvai::selectUpM() // выводит информацию о стоимости улучшения выбранного маршрута

{

if (ui->upMarshrutInput->text() == "")

{

ui->upMarshrutCost->setText("Введите маршрут");

return;

}

uint64\_t selectedMar = ui->upMarshrutInput->text().toULongLong();

if (selectedMar < tramsOnMarsruts.size()) ui->upMarshrutCost->setText(QString::number(costOfMarsrutsUpgrading[selectedMar]) + " ₽");

else

{

ui->upMarshrutCost->setText("Введите маршрут");

return;

}

}

void Tramvai::upMarshrut() // Увеличивает уровень (макс и мин колво трамваев на маршруте) маршрута

{

if (ui->upMarshrutInput->text() == "") // только если ввели корректны маршрут

{

ui->upMarshrutCost->setText("Введите маршрут");

return;

}

uint64\_t selectedMar = ui->upMarshrutInput->text().toULongLong();

if (selectedMar < tramsOnMarsruts.size())

{

if (costOfMarsrutsUpgrading[selectedMar] <= money) // только если хватает денег

{

money -= costOfMarsrutsUpgrading[selectedMar];

costOfMarsrutsUpgrading[selectedMar] += costOfMarsrutsUpgrading[selectedMar] / 10; // типо инфляция

maxTramsOnMarshruts[selectedMar]++; // повышаем макс число трамваев на маршруте

minTramsOnMarshruts[selectedMar] = maxTramsOnMarshruts[selectedMar] -3; // повышаем мин колво трамваев на маршруте

if (minTramsOnMarshruts[selectedMar] == 0) minTramsOnMarshruts[selectedMar] = 1; // если трамваев мало то 1 это минимум

updateDisplay();

}

}

else

{

ui->upMarshrutCost->setText("Введите маршрут");

return;

}

}

Tramvai::~Tramvai()

{

delete ui;

}

**Файл main.cpp**

#include "tramvai.h"

#include <QApplication>

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

{

QApplication a(argc, argv);

Tramvai w;

w.show();

return a.exec();

}

**Файл tramvai.ui**

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

<ui version="4.0">

<class>Tramvai</class>

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

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>1328</width>

<height>765</height>

</rect>

</property>

<property name="windowTitle">

<string>Tramvai</string>

</property>

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

<widget class="QTextBrowser" name="marshrutiANDtrams">

<property name="geometry">

<rect>

<x>10</x>

<y>70</y>

<width>471</width>

<height>391</height>

</rect>

</property>

<property name="font">

<font>

<family>Montserrat Alternates</family>

<pointsize>10</pointsize>

</font>

</property>

</widget>

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

<property name="geometry">

<rect>

<x>490</x>

<y>190</y>

<width>411</width>

<height>51</height>

</rect>

</property>

<property name="font">

<font>

<family>Montserrat Alternates</family>

<pointsize>12</pointsize>

</font>

</property>

<property name="cursor">

<cursorShape>PointingHandCursor</cursorShape>

</property>

<property name="text">

<string>Установить кол-во трамваев на маршрут</string>

</property>

</widget>

<widget class="QLineEdit" name="marshrutInput">

<property name="geometry">

<rect>

<x>1120</x>

<y>190</y>

<width>201</width>

<height>51</height>

</rect>

</property>

<property name="font">

<font>

<family>Montserrat Alternates</family>

<pointsize>14</pointsize>

</font>

</property>

</widget>

<widget class="QLineEdit" name="setTramKolvoInput">

<property name="geometry">

<rect>

<x>910</x>

<y>190</y>

<width>201</width>

<height>51</height>

</rect>

</property>

<property name="font">

<font>

<family>Montserrat Alternates</family>

<pointsize>14</pointsize>

</font>

</property>

</widget>

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

<property name="geometry">

<rect>

<x>490</x>

<y>130</y>

<width>411</width>

<height>51</height>

</rect>

</property>

<property name="font">

<font>

<family>Montserrat Alternates</family>

<pointsize>12</pointsize>

</font>

</property>

<property name="cursor">

<cursorShape>PointingHandCursor</cursorShape>

</property>

<property name="text">

<string>Улучшить маршрут</string>

</property>

</widget>

<widget class="QLineEdit" name="upMarshrutInput">

<property name="geometry">

<rect>

<x>910</x>

<y>130</y>

<width>131</width>

<height>51</height>

</rect>

</property>

<property name="font">

<font>

<family>Montserrat Alternates</family>

<pointsize>14</pointsize>

</font>

</property>

</widget>

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

<property name="geometry">

<rect>

<x>490</x>

<y>70</y>

<width>411</width>

<height>51</height>

</rect>

</property>

<property name="font">

<font>

<family>Montserrat Alternates</family>

<pointsize>12</pointsize>

</font>

</property>

<property name="cursor">

<cursorShape>PointingHandCursor</cursorShape>

</property>

<property name="text">

<string>Купить трамвай</string>

</property>

</widget>

<widget class="QTextBrowser" name="tramCost">

<property name="geometry">

<rect>

<x>910</x>

<y>70</y>

<width>411</width>

<height>51</height>

</rect>

</property>

<property name="font">

<font>

<family>Montserrat Alternates</family>

<pointsize>14</pointsize>

</font>

</property>

</widget>

<widget class="QTextBrowser" name="money">

<property name="geometry">

<rect>

<x>490</x>

<y>10</y>

<width>831</width>

<height>51</height>

</rect>

</property>

<property name="font">

<font>

<family>Montserrat Alternates</family>

<pointsize>14</pointsize>

</font>

</property>

</widget>

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

<property name="geometry">

<rect>

<x>490</x>

<y>250</y>

<width>411</width>

<height>51</height>

</rect>

</property>

<property name="font">

<font>

<family>Montserrat Alternates</family>

<pointsize>12</pointsize>

</font>

</property>

<property name="cursor">

<cursorShape>PointingHandCursor</cursorShape>

</property>

<property name="text">

<string>Повысить цены на билет</string>

</property>

</widget>

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

<property name="geometry">

<rect>

<x>10</x>

<y>10</y>

<width>271</width>

<height>51</height>

</rect>

</property>

<property name="font">

<font>

<family>Montserrat Alternates</family>

<pointsize>14</pointsize>

</font>

</property>

<property name="text">

<string>Маршруты и прогнозы:</string>

</property>

</widget>

<widget class="QLabel" name="label\_2">

<property name="geometry">

<rect>

<x>400</x>

<y>10</y>

<width>81</width>

<height>51</height>

</rect>

</property>

<property name="font">

<font>

<family>Montserrat Alternates</family>

<pointsize>14</pointsize>

</font>

</property>

<property name="text">

<string>Деньги:</string>

</property>

</widget>

<widget class="QTextBrowser" name="upMarshrutCost">

<property name="geometry">

<rect>

<x>1050</x>

<y>130</y>

<width>271</width>

<height>51</height>

</rect>

</property>

<property name="font">

<font>

<family>Montserrat Alternates</family>

<pointsize>14</pointsize>

</font>

</property>

</widget>

<widget class="QTextBrowser" name="biletUPcost">

<property name="geometry">

<rect>

<x>910</x>

<y>250</y>

<width>411</width>

<height>51</height>

</rect>

</property>

<property name="font">

<font>

<family>Montserrat Alternates</family>

<pointsize>14</pointsize>

</font>

</property>

</widget>

<widget class="QLabel" name="label\_3">

<property name="geometry">

<rect>

<x>490</x>

<y>370</y>

<width>831</width>

<height>341</height>

</rect>

</property>

<property name="font">

<font>

<family>Montserrat Alternates</family>

<pointsize>12</pointsize>

</font>

</property>

<property name="text">

<string> Инструкция:

Цель - заработать как можно больше денег и пережить хотя бы 100 дней.

Если на маршруте повышено количество людей, зароботок с маршрута увеличен.

Если людей среднее количество - зароботок базовый.

Если людей мало - зароботок снижен.

Каждый день может случится поломка трамвая или сбитие человека, эти события

уносят некоторое кол-во денег, если денег не хватит, вы банкрот, нужно начать сначала.

Чем больше трамваев, тем больше поломок и аварий.

Каждый новый трамвай дороже предыдущего.

Повышение цен на билеты - дорогая прцедура.

Улучшение маршрута позволяет пустить больше трамваев,

но и минимальное количество трамваев на маршруте также растет.

Открытие нового маршрута - очень дорогое событие.

Каждый трамвай за день потребляет какое-то кол-во электричества.

На каждый сотый день вас ждет ремонт трам. путей, зависящий от кол-ва маршрутов.</string>

</property>

</widget>

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

<property name="geometry">

<rect>

<x>490</x>

<y>310</y>

<width>411</width>

<height>51</height>

</rect>

</property>

<property name="font">

<font>

<family>Montserrat Alternates</family>

<pointsize>12</pointsize>

</font>

</property>

<property name="cursor">

<cursorShape>PointingHandCursor</cursorShape>

</property>

<property name="text">

<string>Новый маршрут</string>

</property>

</widget>

<widget class="QTextBrowser" name="newMarshrutCost">

<property name="geometry">

<rect>

<x>910</x>

<y>310</y>

<width>411</width>

<height>51</height>

</rect>

</property>

<property name="font">

<font>

<family>Montserrat Alternates</family>

<pointsize>14</pointsize>

</font>

</property>

</widget>

<widget class="QLabel" name="label\_4">

<property name="geometry">

<rect>

<x>10</x>

<y>470</y>

<width>341</width>

<height>31</height>

</rect>

</property>

<property name="font">

<font>

<family>Montserrat Alternates</family>

<pointsize>14</pointsize>

</font>

</property>

<property name="text">

<string>Сводка за предыдущий день:</string>

</property>

</widget>

<widget class="QTextBrowser" name="rashodi">

<property name="geometry">

<rect>

<x>10</x>

<y>510</y>

<width>471</width>

<height>251</height>

</rect>

</property>

<property name="font">

<font>

<family>Montserrat Alternates</family>

<pointsize>12</pointsize>

</font>

</property>

</widget>

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

<property name="geometry">

<rect>

<x>490</x>

<y>710</y>

<width>411</width>

<height>51</height>

</rect>

</property>

<property name="font">

<font>

<family>Montserrat Alternates</family>

<pointsize>12</pointsize>

</font>

</property>

<property name="cursor">

<cursorShape>PointingHandCursor</cursorShape>

</property>

<property name="text">

<string>Начать новый день</string>

</property>

</widget>

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

<property name="geometry">

<rect>

<x>910</x>

<y>710</y>

<width>411</width>

<height>51</height>

</rect>

</property>

<property name="font">

<font>

<family>Montserrat Alternates</family>

<pointsize>12</pointsize>

</font>

</property>

<property name="cursor">

<cursorShape>PointingHandCursor</cursorShape>

</property>

<property name="text">

<string>Начать заного</string>

</property>

</widget>

</widget>

</widget>

<resources/>

<connections/>

</ui>

**Файл tramvai.pro**

QT += core gui

greaterThan(QT\_MAJOR\_VERSION, 4): QT += widgets

CONFIG += c++11

# You can make your code fail to compile if it uses deprecated APIs.

# In order to do so, uncomment the following line.

#DEFINES += QT\_DISABLE\_DEPRECATED\_BEFORE=0x060000 # disables all the APIs deprecated before Qt 6.0.0

SOURCES += \

main.cpp \

tramvai.cpp

HEADERS += \

tramvai.h

FORMS += \

tramvai.ui

# Default rules for deployment.

qnx: target.path = /tmp/$${TARGET}/bin

else: unix:!android: target.path = /opt/$${TARGET}/bin

!isEmpty(target.path): INSTALLS += target

**ВЫВОД**

Были составлены две программы: для решения задачи коммивояжера, визуализации графа и для автоматизации рабочего места диспетчера трамвайного депо. Программы были отлажены и протестированы, о работе программ снято видео и выложено на YouTube.com по адресу <https://youtu.be/zgg8TPQ-Rj0>. Для захвата видео с экрана монитора использовалась бесплатная программа nVidia GeForce Experience. Для обработки видео использовалась бесплатная программа DaVinci Resolve.