Федеральное государственное бюджетное образовательное учреждение высшего образования

«Пермский национальный исследовательский политехнический университет»

Электротехнический факультет

Кафедра ИТАС

Отчёт по творческим работам

Решение задачи коммивояжёра

Выполнил: студент группы

Рис-20-1б Ахматнурова О.Ф

Проверил: доцент кафедры ИТАС

Полякова О.А.

Пермь, 2021

Содержание

[1 Творческая работа №2. Решение задачи коммивояжёра 3](#_Toc72693711)

[1.1 Постановка задачи 3](#_Toc72693712)

[1.2 Анализ задачи 3](#_Toc72693713)

[1.3 Результат работы программы 4](#_Toc72693714)

[1.4 Достоинства программы 5](#_Toc72693715)

1. Творческая работа №2. Решение задачи коммивояжёра
   1. Постановка задачи

Создать и реализовать алгоритм решение задачи коммивояжера. По заданию граф является не направленным с уже указанными длинами дуг.

* 1. Анализ задачи

Вся работа программы завязана на кнопке выполнения операций, но не в прям смысле этого слова, здесь же подразумевается кнопка как виджет, и соответствующее с указанным виджетом действие.

Данная задача решается методом ветвей и границ. Этот алгоритмический метод позволяет найти оптимальное решение различных задач оптимизации, особенно дискретной и комбинаторной оптимизации.

Так же существует класс myPainter, который отвечает за различные части визуализации графа.

void myPainter::**draw**(QPainter \*painter, QPaintEvent \*event) //функция дроу в которой происходит обрисовка элементов

{

painter->fillRect(event->rect(),back);

painter->setBrush(front);

painter->setPen(textPen);

painter->setFont(glFont);

textPen.setWidth(4);

extern QVector<TOP> graph1; //вектор объектов топ которые представляют собой веришны

extern QVector<EDGE> graph2; //вектор объектов эдж которые представляют собой ребра

for (int i=0;i<graph1.size();i++)

{

draw\_top(graph1[i].x, //координаты веришны

graph1[i].y,

QString::number(i+1),

*painter*);

}

for (int i=0;i<graph2.size();i++)

{

draw\_edge(graph2[i].x1, //координаты ребра

graph2[i].y1,

graph2[i].x2,

graph2[i].y2,

graph2[i].type,

*painter*,

QString::number(graph2[i].mass));

}

}

В программе предусмотрен вывод ошибки при невозможности расчет графа.

void MainWindow::**Fcheck**()

{

try

{

F();

}

catch(...)

{

QMessageBox::warning(this, "Внимание","невозможно вычислить");

}

}

Так же конечно же все кнопки меняют свой цвет при наведении на них, и так же сам граф при построении.

void MainWindow::**Fswitch**()

{

QPushButton\* button = qobject\_cast<QPushButton\*>(*sender()*);

int buttonID = G->id(*button*);

for (int i=0;i<4;i++)

{

G->button(i)->setStyleSheet("QPushButton { background-color: rgb(237,207,209); }\n");

}

if (buttonID!=4) G->button(buttonID)->setStyleSheet("QPushButton { background-color: rgb(199,112,118); }\n");

extern QString status;

extern QString text\_for\_arrows;

extern QString statusBook[6];

if (buttonID==1) text\_for\_arrows=ent2->text();

status=statusBook[buttonID];

}

* 1. Результат работы программы

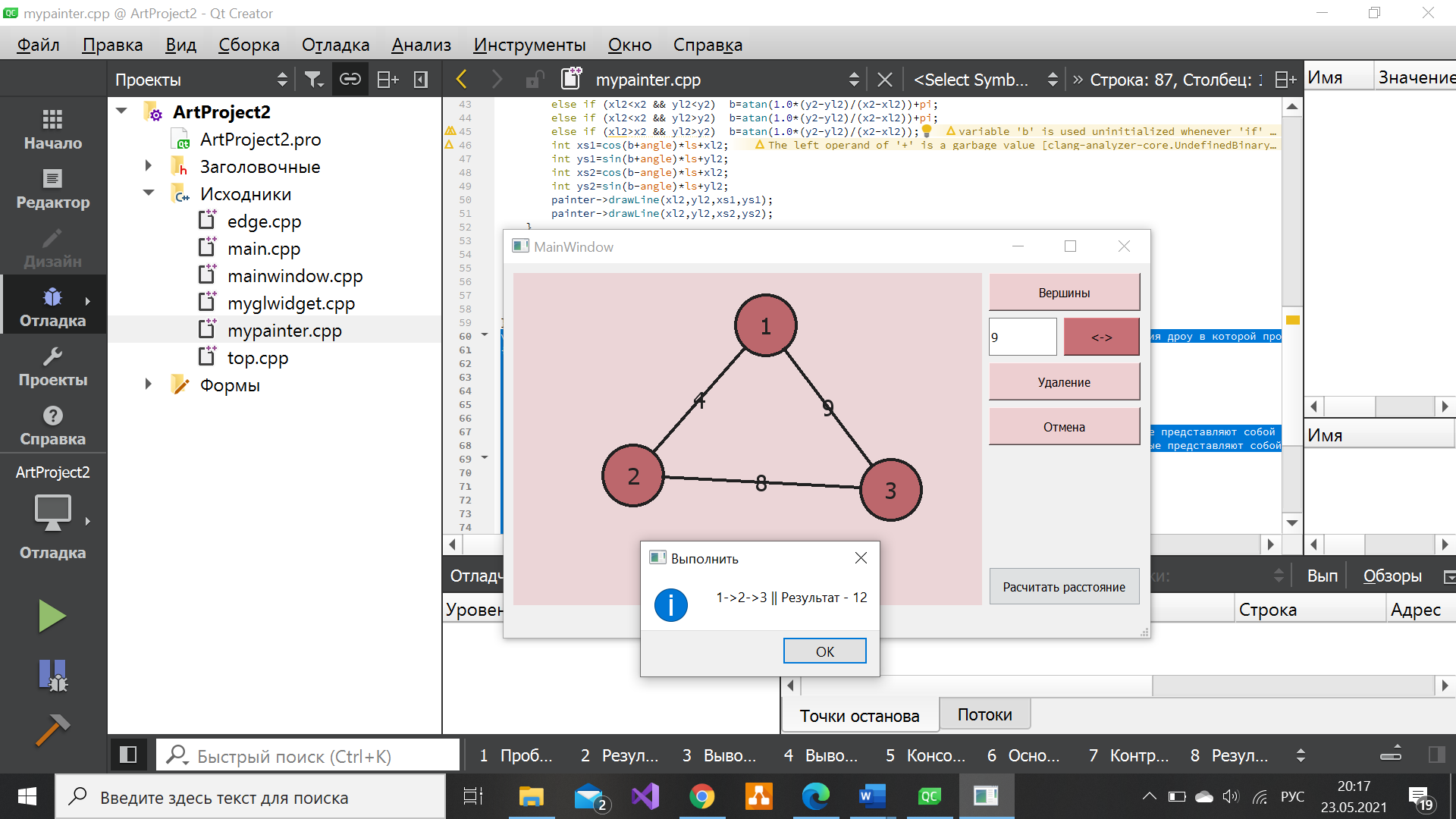


Рисунок 6 – Результат работы программы

* 1. Достоинства программы

К достоинствам проекта можно отнести простоту и понятность интерфейса, так же особенно нужно выделить универсальность программы. Пользователь сам выбирает необходимое количество городов (вершин), так же их длину и программа рассчитывает расстояние вне зависимости от того, какого количество введено.