Министерство высшего образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Пермский национальный исследовательский политехнический университет» (ПНИПУ)

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

ОТЧЁТ

по лабораторной работе

Тема: «Задача Коммивояжера»

Выполнил

Студент группы РИС-22-2б

Арапов М.В.

Проверил доц. Кафедры ИТАС

Полякова О. А.

Пермь 2023

**Постановка задачи**

1) Создание и визуализация графа

2) Выполнение задачи Коммивояжера

Принцип работы кода заключается в реализации алгоритма коммивояжера с использованием метода ветвей и границ.

Основные функции в коде:

**createTab**: Функция, которая создает новую вкладку в главном окне приложения. Она вызывается при нажатии на кнопку "Create Tab".

**buildGraphs**: Функция, которая строит графы для каждой вкладки в программе. Она вызывается при нажатии на кнопку "Build Graphs".

**calculation**: Функция, которая выполняет вычисления и алгоритмы для каждой вкладки в программе. Она вызывается при нажатии на кнопку "Calculation".

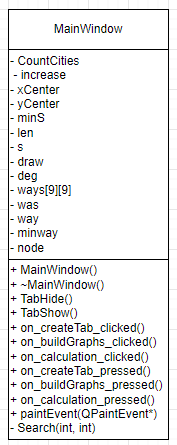
**TabHide**: Функция, которая скрывает текущую вкладку в главном окне приложения. Она вызывается для скрытия вкладки после завершения вычислений.

**Tab Show**: Функция, которая показывает скрытую вкладку в главном окне приложения. Она вызывается для показа вкладки после создания или построения графов.

**paintEvent**: Функция, которая обрабатывает событие перерисовки окна и отображает графику на экране. Она вызывается автоматически при необходимости перерисовки окна.

**Search**: Функция, которая выполняет поиск пути в графе. Она используется для вычисления пути между двумя городами.

**Uml-Диаграмма**



**Код программы**

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <QMessageBox>

#include <cmath>

*struct* **Node**

{

int x,y;

};

*//* *Инициализация* *переменных*

int CountCities, increase,xCenter,yCenter,minS,len,s;

int draw = 0;

double deg=360;

int ways[9][9];

bool \*was;

int \*way, \*minway;

Node \*node;

MainWindow::**MainWindow**(QWidget \*parent) *//* *конструктор* *с* *параметром*

: QMainWindow(*parent*)

, ui(*new* Ui::MainWindow)

{

ui->setupUi(*this*);

ui->buildGraphs -> hide();

ui->calculation -> hide();

TabHide();

QPalette palette;

*this* -> setPalette(palette);

}

MainWindow::~***MainWindow***() *//* *деструктор*

{

*delete* ui;

}

void MainWindow::**TabHide**(){ *//* *Скрытие* *таблицы* *смежности*

QString temp;

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

{

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

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

}

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

{

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

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

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

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

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

{

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

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

}

}

*for* (int i = 9; i > 1; i--)

{

*for* (int e = i - 1; e > 0; e--)

{

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

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

}

}

}

void MainWindow::**TabShow**() *//* *Создание* *таблицы* *смежности*

{

QString temp;

*for* (int i = 1; i < CountCities + 1; i++){

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

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

}

*for* (int i = 1; i < CountCities + 1; i++)

{

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

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

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

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

*for* (int e = i + 1; e < CountCities + 1; e++)

{

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

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

}

}

*for* (int i = CountCities; i > 1; i--)

{

*for* (int e = i - 1; e > 0; e--)

{

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

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

}

}

}

void MainWindow::**on\_createTab\_clicked**() *//* *Приватный* *слот* *клика* *на* *кнопку* *создания* *таблицы*

{

*//* *Изменение* *стиля* *кнопки* *создания* *таблицы* *смежности*

ui -> createTab -> setStyleSheet("color: rgb(255,255,255); background-color: rgb(100,100,100); font-size:20px; border-radius: 5px;");

ui -> calculation -> hide();

ui -> buildGraphs -> hide();

*if* (ui -> CountLabel -> text().toInt() < 3||ui -> CountLabel -> text().toInt() > 9) { *//* *Проверка* *корректности* *введенных* *данных* *размера* *таблицы*

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

TabHide();

draw = 0;

*this* -> repaint();

ui -> statusBar -> setStyleSheet("color:red;");

statusBar() -> showMessage("The number of cities must be a number and have a value of at least 3 and at most 9!"); *//* *вывод* *ошибки* *в* *statusbar*

}

*else* { *//* *создание* *таблицы* *смежности*

TabHide();

statusBar() -> clearMessage();

ui -> buildGraphs -> show();

CountCities=ui -> CountLabel -> text().toInt();

TabShow();

node = *new* Node[CountCities];

xCenter = 820;

yCenter = 280;

draw = 1; *//* *Присвоение* *нового* *значение* *переменной,* *draw* *для* *изменения* *графического* *представления*

*this* -> repaint();

}

}

void MainWindow::**on\_buildGraphs\_clicked**() *//* *Приватный* *слот* *клика* *на* *кнопку* *создания* *ребер* *графов*

{

ui -> buildGraphs -> setStyleSheet("color: rgb(255,255,255); background-color: rgb(100,100,100); font-size:20px; border-radius: 5px;");

ui -> calculation->hide();

QString searcher, buffer;

*for* (int i = 0;i < CountCities;i++) *//* *Вложенный* *цикл* *проверки* *путей* *графа*

{

*for* (int e = 0;e < CountCities;e++)

{

*if* (i == e) { *//* *Присвоение* *элементам* *диагонали* *значение* *0*

buffer = "0";

}

*else* { *//* *проверка* *значение* *элемента* *таблицы* *смежности*

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

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

}

ways[i][e] = buffer.toInt(); *//* *Присвоение* *элементам* *вложенного* *массива* *значений* *таблицы* *смежности*

}

}

ui -> calculation -> show();

draw = 2; *//* *Присвоение* *нового* *значение* *переменной,* *draw* *для* *изменения* *графического* *представления*

*this* -> repaint();

}

void MainWindow::**on\_calculation\_clicked**() *//* *Приватный* *слот* *клика* *на* *кнопку* *вычисления* *кратчайшего* *пути*

{

ui -> calculation -> setStyleSheet("color: rgb(255,255,255); background-color: rgb(100,100,100); font-size:20px; border-radius: 5px;");

ui -> calculation -> hide();

QString buffer;

bool notway = 0;

len = 0;

s = 0;

minS = 10000;

was = *new* bool [CountCities];

way = *new* int [CountCities];

minway = *new* int [CountCities];

*for* (int i = 0; i < CountCities; i++)

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

*for* (int i = 0; i < CountCities; i++)

Search(0,i);

*for* (int i = 0; i < CountCities; i++)

*if* (minway[i] == -1) { notway = 1; }

*if* (notway || minS == 9999) {

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 < CountCities; i++)

{buffer += "->" + QString::number(minway[i] + 1);}

buffer += "->" + QString::number(minway[0] + 1);

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

draw = 3;

*this* -> repaint();

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

}

}

void MainWindow::**on\_createTab\_pressed**() *//* *Приватный* *слот* *нажатия* *на* *кнопку* *создания* *таблицы*

{

ui->createTab->setStyleSheet("color: rgb(30,30,30);background-color: rgb(100,100,100);font-size:20px;border-radius: 5px; border; border: 2px solid rgb(30,30,30)");

}

void MainWindow::**on\_buildGraphs\_pressed**() *//* *Приватный* *слот* *нажатия* *на* *кнопку* *создания* *ребер* *графа*

{

ui->buildGraphs->setStyleSheet("color: rgb(30,30,30);background-color: rgb(100,100,100);font-size:20px;border-radius: 5px; border; border: 2px solid rgb(30,30,30)");

}

void MainWindow::**on\_calculation\_pressed**() *//* *Приватный* *слот* *нажатия* *на* *кнопку* *вычисления* *кратчайшего* *пути*

{

ui->calculation->setStyleSheet("color: rgb(30,30,30);background-color: rgb(100,100,100);font-size:20px;border-radius: 5px;border; border: 2px solid rgb(30,30,30)");

}

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

{

*if* (draw) { *//* *Отрисовка* *элементов* *графа*

int indent;

increase = 360/CountCities;

deg = 0;

QPainter paint;

QPen pen;

QBrush brush;

QString str;

QFont font;

pen.setColor(Qt::*black*);

pen.setWidth(7);

pen.setStyle(Qt::*SolidLine*);

font.setPointSize(20);

brush.setStyle(Qt::*SolidPattern*);

brush.setColor(Qt::*cyan*);

paint.begin(*this*);

paint.setPen(pen);

paint.setFont(font);

paint.setBrush(brush);

*for* (int i = 0;i < CountCities; i++)

{

indent = -10;

*if* (deg > 180) indent = 90;

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

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

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

paint.drawEllipse(node[i].x, node[i].y, 70, 70);

pen.setColor(Qt::*black*);

paint.setPen(pen);

paint.drawText(node[i].x + 50, node[i].y + indent,str);

pen.setColor(Qt::*black*);

paint.setPen(pen);

deg += increase;

}

*if* (draw > 1) { *//* *Отрисовка* *ребер* *графа*

*for* (int i = 0; i < CountCities; i++)

{

*for* (int e = 0; e < CountCities; e++)

{

*if* (ways[i][e] < 1000 && ways[i][e] > 0)

{

pen.setStyle(Qt::*SolidLine*);

pen.setWidth(2);

paint.setPen(pen);

paint.drawLine(node[i].x + 35, node[i].y + 35, node[e].x + 35, node[e].y + 35);

}

}

}

}

*if* (draw == 3) *//* *Отрисовка* *минимального* *пути*

{

pen.setStyle(Qt::*DashLine*);

pen.setWidth(4);

pen.setColor(Qt::*blue*);

paint.setPen(pen);

*for* (int i = 0; i < CountCities - 1; i++)

{

paint.drawLine(node[minway[i]].x + 35, node[minway[i]].y + 35, node[minway[i + 1]].x + 35, node[minway[i + 1]].y + 35);

paint.drawLine(node[minway[0]].x + 35, node[minway[0]].y + 35, node[minway[CountCities - 1]].x + 35, node[minway[CountCities - 1]].y + 35);

paint.drawEllipse(node[minway[i]].x + 17,node[minway[i]].y + 17, 35, 35);

paint.drawEllipse(node[minway[i + 1]].x + 17, node[minway[i + 1]].y + 17, 35, 35);

}

}

paint.end();

}

}

void MainWindow::**Search**(int l, int i){ *//* *Поиск* *минимального* *пути*

len += l;

was[i] = 1;

bool b = 0;

way[s] = i;

s++;

*for* (int e = 0; e < CountCities; e++)

{

*if* (!was[e]) {

*if* (ways[i][e]) Search(ways[i][e], e);

b = 1;

}

}

*if*(!b){

bool minus = 0;

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

*if* (!minus&& len < minS) {

minS = len;

*for* (int z = 0; z < CountCities; z++)

{

minway[z] = way[z];

}

}

}

len -= l;

s--;

was[i] = 0;

way[s] = -1;

}

**Вывод**

