Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«Пермский национальный исследовательский политехнический университет»**

Кафедра **«**Информационные технологии и автоматизированные системы**»**

**ОТЧЕТ**

**ПО ТВОРЧЕСКОЙ РАБОТЕ №2**

Дисциплина: «Основы алгоритмизации и программирования»

Семестр 2

Тема: Задача Коммивояжёра

Выполнил работу

Студент группы РИС-22-1Б

Поважный В. Е.

Проверил

Доцент кафедры ИТАС

Полякова О.А.

г. Пермь-2023

**Введение**

Для выполнения лабораторной работы требуется сделать отчет программы. Создание программ – отличный способ практики программирования.

**Постановка задачи**

1. Реализация приложения на основе OpenGL, для решения задачи Коммивояжёра.
2. Изучить метод ветвей и границ.
3. Визуализация графа и решения задачи.

Метод ветвей и границ заключается в том, что нужно построить дерево потенциальных маршрутов. На каждом шаге нужно рассчитывать оценку перспективности маршрута и сравнивать с базовым циклом, при превышении которого маршрут не является перспективным, и отбрасывать его.

**Код программы**

**Graph.h**

#pragma once

#include <GL/glut.h>

#include <iostream>

#include <iomanip>

#include <vector>

#include <string>

using namespace std;

const int maxSize = 20;

extern int WinW;

extern int WinH;

struct VertexCoord

{

int x, y;

};

class Graph

{

private:

vector<int> vertList;

public:

int adjMatrix[maxSize][maxSize];

Graph();

~Graph();

bool IsEmpty();

bool IsFull();

int GetVertPos(const int& vertex);

int GetVertText(int i);

vector<int> GetVertList();

int GetAdjMatrixElem(int i, int j);

int GetAmountVerts();

int GetAmountEdges();

void SetEdgeZero(int i, int j);

void InsertVertex(const int& vertex);

void InsertEdge(const int& vertex1, const int& vertex2, int weigth);

void Print();

void EraseLastVert();

void EraseEdge(const int& vertex1, const int& vertex2);

void DrawGraph();

};

extern Graph graph;

void mouseMove(int x, int y);

void makeGraph();

void reshape(int w, int h);

void display();

int\*\* Change\_Matrix();

int\* Search\_MinElem(int\* line, int n);

void Print\_Matrix(int\*\* matrix);

int\*\* Reduct\_Matrix(int\*\* oldmatrix);

int\*\* High\_Zero(int\*\* oldmatrix);

void Print\_Result();

void setCoords(int i, int n);

bool SalesmanPossible(int\*\* matrix);

void ButtonMouseOn(int x, int y);

void mouseMove(int x, int y);

void mouseClick(int button, int state, int x, int y);

void drawBtnSalesman();

void drawBtnNewGraph();

void drawBtnAddVertex();

void drawBtnDelVertex();

void drawBtnAddEdge();

void drawBtnDelEdge();

void drawCircle(int x, int y, int R, bool r);

void drawText(int nom, int x1, int y1, bool red);

void drawLine(int text, int x0, int y0, int x1, int y1, bool b);

void drawVertex(int n, bool r);

**Graph.cpp**

**#include "Graph.h"**

**using namespace std;**

**int WinW = 900;**

**int WinH = 600;**

**int R;**

**int ver = 0;**

**bool ButtonSalesmanSecond = false;**

**bool\* vertmouse = new bool[maxSize];**

**bool standView = false;**

**bool vertmove = false;**

**bool commiglut = false;**

**bool prohod = false;**

**int ButtonCheck;**

**int Moving\_Vertex;**

**int x\_coord\_mouse;**

**int y\_coord\_mouse;**

**VertexCoord vertcrd[maxSize + 2];**

**Graph graph;**

**vector<pair<int, int>> Way;**

**vector<int> New\_Way;**

**Graph::Graph()**

**{**

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

**{**

**for (int j = 0; j < maxSize; j++)**

**adjMatrix[i][j] = 0;**

**vertmouse[i] = false;**

**}**

**}**

**Graph::~Graph()**

**{**

**}**

**int Graph::GetVertPos(const int& vertex)**

**{**

**for (size\_t i = 0; i < vertList.size(); i++)**

**if (vertList[i] == vertex)**

**return i;**

**return -1;**

**}**

**int Graph::GetVertText(int i)**

**{**

**return vertList[i];**

**}**

**vector<int> Graph::GetVertList()**

**{**

**return vertList;**

**}**

**int Graph::GetAdjMatrixElem(int i, int j)**

**{**

**return adjMatrix[i][j];**

**}**

**int Graph::GetAmountVerts()**

**{**

**return vertList.size();**

**}**

**void Graph::SetEdgeZero(int i, int j)**

**{**

**adjMatrix[i][j] = 0; adjMatrix[j][i] = 0;**

**}**

**bool Graph::IsEmpty()**

**{**

**if (vertList.size() != 0)**

**return false;**

**else**

**return true;**

**}**

**bool Graph::IsFull()**

**{**

**return (vertList.size() == maxSize);**

**}**

**void Graph::InsertVertex(const int& vertex)**

**{**

**if (!IsFull())**

**vertList.push\_back(vertex);**

**else**

**{**

**cout << "Граф уже заполнен. Невозможно добавить новую вершину" << endl;**

**return;**

**}**

**}**

**void Graph::InsertEdge(const int& vertex1, const int& vertex2, int weight)**

**{**

**if (GetVertPos(vertex1) != (-1) && GetVertPos(vertex2) != (-1))**

**{**

**int vertPos1 = GetVertPos(vertex1);**

**int vertPos2 = GetVertPos(vertex2);**

**if (adjMatrix[vertPos1][vertPos2] != 0 || adjMatrix[vertPos2][vertPos1] != 0)**

**{**

**cout << "Ребро между вершинами уже есть" << endl;**

**return;**

**}**

**else**

**adjMatrix[vertPos1][vertPos2] = weight;**

**}**

**else**

**{**

**if (weight < 1)**

**cout << "\nДанная величина веса некорректна\n";**

**else**

**cout << "Обеих вершин (или одной из них) нет в графе" << endl; cout << "Исходная вершина: ";**

**int sourceVertex, targetVertex, edgeWeight;**

**cin >> sourceVertex;**

**cout << "Конечная вершина: ";**

**cin >> targetVertex;**

**cout << "Вес ребра: ";**

**cin >> edgeWeight;**

**graph.InsertEdge(sourceVertex, targetVertex, edgeWeight);**

**}**

**}**

**void Graph::Print()**

**{**

**if (!IsEmpty())**

**{**

**cout << "Матрица смежности графа: " << endl;**

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

**{**

**cout << vertList[i] << " ";**

**for (int j = 0; j < vertList.size(); j++)**

**cout << setw(4) << adjMatrix[i][j];**

**cout << endl;**

**}**

**}**

**else**

**cout << "\nГраф пуст\n" << endl;**

**}**

**void Graph::EraseLastVert()**

**{**

**if (IsEmpty())**

**{**

**cout << "\nГраф пуст\n";**

**return;**

**}**

**int n = vertList.size();**

**for (int j = 0; j < n; j++)**

**{**

**adjMatrix[n - 1][j] = 0;**

**adjMatrix[j][n - 1] = 0;**

**}**

**vertList.pop\_back();**

**}**

**void Graph::EraseEdge(const int& vertex1, const int& vertex2)**

**{**

**if (GetVertPos(vertex1) != (-1) && GetVertPos(vertex2) != (-1))**

**{**

**int vertPos1 = GetVertPos(vertex1);**

**int vertPos2 = GetVertPos(vertex2);**

**if (adjMatrix[vertPos1][vertPos2] != 0 || adjMatrix[vertPos2][vertPos1] != 0)**

**{**

**adjMatrix[vertPos1][vertPos2] = 0;**

**adjMatrix[vertPos2][vertPos1] = 0;**

**}**

**else**

**{**

**cout << "Ребра между данными вершинами нет\n" << endl;**

**return;**

**}**

**}**

**else**

**{**

**cout << "Обеих вершин (или одной из них) нет в графе" << endl;**

**return;**

**}**

**}**

**int Graph::GetAmountEdges()**

**{**

**int numb = 0;**

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

**for (int j = 0; j < vertList.size(); j++)**

**if (adjMatrix[i][j] > 0)**

**numb++;**

**return numb;**

**}**

**void Graph::DrawGraph()**

**{**

**int n = graph.GetAmountVerts();**

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

**if (!standView)**

**setCoords(i, n);**

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

**for (int j = 0; j < n; j++)**

**{**

**int a = adjMatrix[i][j];**

**if (a != 0)**

**drawLine(a, vertcrd[i].x, vertcrd[i].y, vertcrd[j].x, vertcrd[j].y, false);**

**if (a == adjMatrix[j][i] && a != 0)**

**drawLine(a, vertcrd[j].x, vertcrd[j].y, vertcrd[i].x, vertcrd[i].y, false);**

**}**

**drawVertex(n, prohod);**

**glutPostRedisplay();**

**}**

**int ClickOnCircle(int x, int y)**

**{**

**for (int i = 0; i < graph.GetAmountVerts(); i++)**

**if (pow(x - vertcrd[i].x, 2) + pow(y - vertcrd[i].y, 2) <= pow(R, 2))**

**return i;**

**return -1;**

**}**

**void ButtonMouseOn(int x, int y)**

**{**

**if (x > 10 && x < WinW / 7 && y > WinH - WinH / 10 && y < WinH - 20)**

**{**

**ButtonCheck = 1;**

**}**

**else**

**{**

**if (x > 10 && x < WinW / 7 && y > WinH - 2 \* (WinH / 10) && y < WinH - (WinH / 10) - 20)**

**{**

**ButtonCheck = 2;**

**}**

**else**

**{**

**if (x > 10 && x < WinW / 7 && y > WinH - 3 \* (WinH / 10) && y < WinH - 2 \* (WinH / 10) - 20)**

**{**

**ButtonCheck = 3;**

**}**

**else**

**{**

**if (x > 10 && x < WinW / 7 && y > WinH - 4 \* (WinH / 10) && y < WinH - 3 \* (WinH / 10) - 20)**

**{**

**ButtonCheck = 4;**

**}**

**else**

**{**

**if (x > 10 && x < WinW / 7 && y > WinH - 5 \* (WinH / 10) && y < WinH - 4 \* (WinH / 10) - 20)**

**{**

**ButtonCheck = 5;**

**}**

**else**

**{**

**if (x > 10 && x < WinW / 7 && y > WinH - 6 \* (WinH / 10) && y < WinH - 5 \* (WinH / 10) - 20)**

**{**

**ButtonCheck = 6;**

**}**

**else**

**{**

**ButtonCheck = 0;**

**}**

**}**

**}**

**}**

**}**

**}**

**}**

**void mouseMove(int x, int y)**

**{**

**y = WinH - y;**

**x\_coord\_mouse = x;**

**y\_coord\_mouse = y;**

**int i = ClickOnCircle(x, y);**

**if (i != -1)**

**vertmouse[i] = true;**

**else**

**for (int j = 0; j < graph.GetAmountVerts(); j++)**

**vertmouse[j] = false;**

**if (vertmove)**

**{**

**vertcrd[Moving\_Vertex].x = x\_coord\_mouse;**

**vertcrd[Moving\_Vertex].y = y\_coord\_mouse;**

**}**

**ButtonMouseOn(x, y);**

**glutPostRedisplay();**

**}**

**void mouseClick(int button, int state, int x, int y)**

**{**

**int j = ClickOnCircle(x, WinH - y);**

**if (vertmove)**

**if (button == GLUT\_LEFT\_BUTTON && state == GLUT\_DOWN)**

**{**

**vertmove = false;**

**return;**

**}**

**if (j != -1)**

**{**

**standView = true;**

**if (button == GLUT\_LEFT\_BUTTON && state == GLUT\_DOWN)**

**{**

**vertmove = true;**

**Moving\_Vertex = j;**

**return;**

**}**

**}**

**if ((x > 10) && (x < WinW / 7) && (y > 20) && (y < WinH / 10))**

**{**

**if (button == GLUT\_LEFT\_BUTTON && state == GLUT\_DOWN)**

**{**

**New\_Way.clear();**

**Way.clear();**

**cout << "\nЗадача Коммивояжера:\n";**

**int\*\* matrix = Change\_Matrix();**

**bool checker = SalesmanPossible(matrix);**

**if (!checker)**

**{**

**cout << "\nДля данного графа нельзя выполнить задачу коммивояжера\n";**

**return;**

**}**

**int n = graph.GetAmountVerts();**

**while (Way.size() < n)**

**matrix = High\_Zero(matrix);**

**cout << endl;**

**ButtonSalesmanSecond = true;**

**Print\_Result();**

**if (prohod)**

**prohod = false;**

**else**

**prohod = true;**

**}**

**}**

**if ((x > 10) && (x < WinW / 7) && (y > (WinH / 10) + 20) && (y < 2 \* (WinH / 10)))**

**{**

**if (button == GLUT\_LEFT\_BUTTON && state == GLUT\_DOWN)**

**{**

**ButtonSalesmanSecond = false;**

**prohod = false;**

**New\_Way.clear();**

**Way.clear();**

**Graph New;**

**graph = New;**

**makeGraph();**

**return;**

**}**

**if (button == GLUT\_RIGHT\_BUTTON && state == GLUT\_DOWN)**

**{**

**standView = false;**

**return;**

**}**

**}**

**if ((x > 10) && (x < WinW / 7) && (y > 2 \* (WinH / 10) + 20) && (y < 3 \* (WinH / 10)))**

**{**

**if (button == GLUT\_LEFT\_BUTTON && state == GLUT\_DOWN)**

**{**

**ButtonSalesmanSecond = false;**

**prohod = false;**

**int cur = graph.GetAmountVerts();**

**graph.InsertVertex(cur + 1);**

**vertcrd[cur].x = WinW / 2;**

**vertcrd[cur].y = WinH / 2;**

**return;**

**}**

**}**

**if ((x > 10) && (x < WinW / 7) && (y > 3 \* (WinH / 10) + 20) && (y < 4 \* (WinH / 10)))**

**{**

**if (button == GLUT\_LEFT\_BUTTON && state == GLUT\_DOWN)**

**{**

**ButtonSalesmanSecond = false;**

**prohod = false;**

**graph.EraseLastVert();**

**int cur = graph.GetAmountVerts();**

**return;**

**}**

**}**

**if ((x > 10) && (x < WinW / 7) && (y > 4 \* (WinH / 10) + 20) && (y < 5 \* (WinH / 10)))**

**{**

**if (button == GLUT\_LEFT\_BUTTON && state == GLUT\_DOWN)**

**{**

**ButtonSalesmanSecond = false;**

**prohod = false;**

**int sourceVertex, targetVertex, edgeWeight;**

**cout << "Исходная вершина: ";**

**cin >> sourceVertex;**

**cout << "Конечная вершина: ";**

**cin >> targetVertex;**

**cout << "Вес ребра: ";**

**cin >> edgeWeight;**

**graph.InsertEdge(sourceVertex, targetVertex, edgeWeight);**

**return;**

**}**

**}**

**if ((x > 10) && (x < WinW / 7) && (y > 5 \* (WinH / 10) + 20) && (y < 6 \* (WinH / 10)))**

**{**

**if (button == GLUT\_LEFT\_BUTTON && state == GLUT\_DOWN)**

**{**

**ButtonSalesmanSecond = false;**

**prohod = false;**

**int sourceVertex, targetVertex;**

**cout << "Исходная вершина: ";**

**cin >> sourceVertex;**

**cout << "Конечная вершина: ";**

**cin >> targetVertex;**

**graph.EraseEdge(sourceVertex, targetVertex);**

**return;**

**}**

**}**

**}**

**void drawBtnSalesman()**

**{**

**if (ButtonCheck == 1)**

**glColor3f(0.603, 0.803, 0.196);**

**else**

**glColor3f(0.980, 0.980, 0.823);**

**glBegin(GL\_QUADS);**

**glVertex2i(10, WinH - 20);**

**glVertex2i(10, WinH - WinH / 10);**

**glVertex2i(WinW / 7, WinH - WinH / 10);**

**glVertex2i(WinW / 7, WinH - 20);**

**glEnd();**

**glColor3f(0.0f, 0.0f, 0.0f);**

**glBegin(GL\_LINE\_LOOP);**

**glVertex2i(10, WinH - 20);**

**glVertex2i(10, WinH - WinH / 10);**

**glVertex2i(WinW / 7, WinH - WinH / 10);**

**glVertex2i(WinW / 7, WinH - 20);**

**glEnd();**

**string name = "Salesman";**

**glRasterPos2i(WinW / 34, 0.93 \* WinH);**

**for (int i = 0; i < name.length(); i++)**

**glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13, name[i]);**

**}**

**void drawBtnNewGraph()**

**{**

**if (ButtonCheck == 2)**

**glColor3f(0.603, 0.803, 0.196);**

**else**

**glColor3f(0.980, 0.980, 0.823);**

**glBegin(GL\_QUADS);**

**glVertex2i(10, WinH - (WinH / 10) - 20);**

**glVertex2i(10, WinH - 2 \* (WinH / 10));**

**glVertex2i(WinW / 7, WinH - 2 \* (WinH / 10));**

**glVertex2i(WinW / 7, WinH - (WinH / 10) - 20);**

**glEnd();**

**glColor3f(0.0f, 0.0f, 0.0f);**

**glBegin(GL\_LINE\_LOOP);**

**glVertex2i(10, WinH - (WinH / 10) - 20);**

**glVertex2i(10, WinH - 2 \* (WinH / 10));**

**glVertex2i(WinW / 7, WinH - 2 \* (WinH / 10));**

**glVertex2i(WinW / 7, WinH - WinH / 10 - 20);**

**glEnd();**

**string name = "Create new";**

**string name1 = "Graph";**

**glRasterPos2i(WinW / 34, (0.83 \* WinH) + 5);**

**for (int i = 0; i < name.length(); i++)**

**glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13, name[i]);**

**glRasterPos2i(WinW / 34, (0.83 \* WinH) - 13);**

**for (int i = 0; i < name1.length(); i++)**

**glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13, name1[i]);**

**}**

**void drawBtnAddVertex()**

**{**

**if (ButtonCheck == 3)**

**glColor3f(0.603, 0.803, 0.196);**

**else**

**glColor3f(0.980, 0.980, 0.823);**

**glBegin(GL\_QUADS);**

**glVertex2i(10, WinH - 2 \* (WinH / 10) - 20);**

**glVertex2i(10, WinH - 3 \* (WinH / 10));**

**glVertex2i(WinW / 7, WinH - 3 \* (WinH / 10));**

**glVertex2i(WinW / 7, WinH - 2 \* (WinH / 10) - 20);**

**glEnd();**

**glColor3f(0.0f, 0.0f, 0.0f);**

**glBegin(GL\_LINE\_LOOP);**

**glVertex2i(10, WinH - 2 \* (WinH / 10) - 20);**

**glVertex2i(10, WinH - 3 \* (WinH / 10));**

**glVertex2i(WinW / 7, WinH - 3 \* (WinH / 10));**

**glVertex2i(WinW / 7, WinH - 2 \* (WinH / 10) - 20);**

**glEnd();**

**string name = "Add Vertex";**

**glRasterPos2i(WinW / 34, 0.73 \* WinH);**

**for (int i = 0; i < name.length(); i++)**

**glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13, name[i]);**

**}**

**void drawBtnDelVertex()**

**{**

**if (ButtonCheck == 4)**

**glColor3f(0.603, 0.803, 0.196);**

**else**

**glColor3f(0.980, 0.980, 0.823);**

**glBegin(GL\_QUADS);**

**glVertex2i(10, WinH - 3 \* (WinH / 10) - 20);**

**glVertex2i(10, WinH - 4 \* (WinH / 10));**

**glVertex2i(WinW / 7, WinH - 4 \* (WinH / 10));**

**glVertex2i(WinW / 7, WinH - 3 \* (WinH / 10) - 20);**

**glEnd();**

**glColor3f(0.0f, 0.0f, 0.0f);**

**glBegin(GL\_LINE\_LOOP);**

**glVertex2i(10, WinH - 3 \* (WinH / 10) - 20);**

**glVertex2i(10, WinH - 4 \* (WinH / 10));**

**glVertex2i(WinW / 7, WinH - 4 \* (WinH / 10));**

**glVertex2i(WinW / 7, WinH - 3 \* (WinH / 10) - 20);**

**glEnd();**

**string name = "Delete last ";**

**string name1 = "Vertex";**

**glRasterPos2i(WinW / 34, (0.63 \* WinH) + 5);**

**for (int i = 0; i < name.length(); i++)**

**glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13, name[i]);**

**glRasterPos2i(WinW / 34, (0.63 \* WinH) - 13);**

**for (int i = 0; i < name1.length(); i++)**

**glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13, name1[i]);**

**}**

**void drawBtnAddEdge()**

**{**

**if (ButtonCheck == 5)**

**glColor3f(0.603, 0.803, 0.196);**

**else**

**glColor3f(0.980, 0.980, 0.823);**

**glBegin(GL\_QUADS);**

**glVertex2i(10, WinH - 4 \* (WinH / 10) - 20);**

**glVertex2i(10, WinH - 5 \* (WinH / 10));**

**glVertex2i(WinW / 7, WinH - 5 \* (WinH / 10));**

**glVertex2i(WinW / 7, WinH - 4 \* (WinH / 10) - 20);**

**glEnd();**

**glColor3f(0.0f, 0.0f, 0.0f);**

**glBegin(GL\_LINE\_LOOP);**

**glVertex2i(10, WinH - 4 \* (WinH / 10) - 20);**

**glVertex2i(10, WinH - 5 \* (WinH / 10));**

**glVertex2i(WinW / 7, WinH - 5 \* (WinH / 10));**

**glVertex2i(WinW / 7, WinH - 4 \* (WinH / 10) - 20);**

**glEnd();**

**string name = "Add Edge";**

**glRasterPos2i(WinW / 34, 0.53 \* WinH);**

**for (int i = 0; i < name.length(); i++)**

**glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13, name[i]);**

**}**

**void drawBtnDelEdge()**

**{**

**if (ButtonCheck == 6)**

**glColor3f(0.603, 0.803, 0.196);**

**else**

**glColor3f(0.980, 0.980, 0.823);**

**glBegin(GL\_QUADS);**

**glVertex2i(10, WinH - 5 \* (WinH / 10) - 20);**

**glVertex2i(10, WinH - 6 \* (WinH / 10));**

**glVertex2i(WinW / 7, WinH - 6 \* (WinH / 10));**

**glVertex2i(WinW / 7, WinH - 5 \* (WinH / 10) - 20);**

**glEnd();**

**glColor3f(0.0f, 0.0f, 0.0f);**

**glBegin(GL\_LINE\_LOOP);**

**glVertex2i(10, WinH - 5 \* (WinH / 10) - 20);**

**glVertex2i(10, WinH - 6 \* (WinH / 10));**

**glVertex2i(WinW / 7, WinH - 6 \* (WinH / 10));**

**glVertex2i(WinW / 7, WinH - 5 \* (WinH / 10) - 20);**

**glEnd();**

**string name = "Delete Edge";**

**glRasterPos2i(WinW / 34, 0.43 \* WinH);**

**for (int i = 0; i < name.length(); i++)**

**glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13, name[i]);**

**}**

**void makeGraph()**

**{**

**standView = false;**

**int amountVerts, amountEdges, sourceVertex, targetVertex, edgeWeight;**

**cout << "Введите количество вершин в графе: ";**

**cin >> amountVerts;**

**cout << "Введите количество ребер в графе: ";**

**cin >> amountEdges;**

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

**graph.InsertVertex(i);**

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

**{**

**cout << "Исходная вершина: ";**

**cin >> sourceVertex;**

**cout << "Конечная вершина: ";**

**cin >> targetVertex;**

**cout << "Вес ребра: ";**

**cin >> edgeWeight;**

**graph.InsertEdge(sourceVertex, targetVertex, edgeWeight);**

**}**

**cout << endl;**

**graph.Print();**

**}**

**void reshape(int w, int h)**

**{**

**WinW = w;**

**WinH = h;**

**glViewport(0, 0, (GLsizei)WinW, (GLsizei)WinH);**

**glMatrixMode(GL\_PROJECTION);**

**glLoadIdentity();**

**gluOrtho2D(0, (GLdouble)WinW, 0, (GLdouble)WinH);**

**glutPostRedisplay();**

**}**

**void display()**

**{**

**glMatrixMode(GL\_PROJECTION);**

**glLoadIdentity();**

**gluOrtho2D(0, WinW, 0, WinH);**

**glViewport(0, 0, WinW, WinH);**

**glClearColor(1.0, 1.0, 1.0, 1.0);**

**glClear(GL\_COLOR\_BUFFER\_BIT);**

**drawBtnNewGraph();**

**drawBtnSalesman();**

**drawBtnAddVertex();**

**drawBtnDelVertex();**

**drawBtnAddEdge();**

**drawBtnDelEdge();**

**graph.DrawGraph();**

**if (prohod)**

**{**

**for (int i = 0; i < New\_Way.size() - 1; i++)**

**{**

**int vertPos1 = graph.GetVertPos(New\_Way[i]);**

**int vertPos2 = graph.GetVertPos(New\_Way[i + 1]);**

**drawLine(i + 1, vertcrd[vertPos2].x, vertcrd[vertPos2].y, vertcrd[vertPos1].x, vertcrd[vertPos1].y, true);**

**}**

**for (int i = 0; i < graph.GetAmountVerts(); i++)**

**{**

**for (int j = 0; j < New\_Way.size(); j++)**

**{**

**if (i == New\_Way[j])**

**drawCircle(vertcrd[i].x, vertcrd[i].y, R / 2, true);**

**if (i == 0)**

**drawCircle(vertcrd[i].x, vertcrd[i].y, R / 2, true);**

**}**

**drawText(i + 1, vertcrd[i].x, vertcrd[i].y, false);**

**}**

**}**

**glutSwapBuffers();**

**}**

**int\*\* Change\_Matrix()**

**{**

**int n = graph.GetAmountVerts();**

**int\*\* matrix = new int\* [n];**

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

**matrix[i] = new int[n];**

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

**for (int j = 0; j < n; j++)**

**{**

**int elem = graph.GetAdjMatrixElem(i, j);**

**if (elem == 0 or i == j)**

**matrix[i][j] = -1;**

**else**

**matrix[i][j] = elem;**

**}**

**if (ButtonSalesmanSecond == true)**

**cout << "Начальная матрица: \n";**

**Print\_Matrix(matrix);**

**return matrix;**

**}**

**int\* Search\_MinElem(int\* line, int n)**

**{**

**int min = 1000000;**

**for (int j = 0; j < n; j++)**

**if (line[j] >= 0 && line[j] < min)**

**min = line[j];**

**for (int j = 0; j < n; j++)**

**if (line[j] >= 0)**

**line[j] -= min;**

**return line;**

**}**

**void Print\_Matrix(int\*\* matrix)**

**{**

**int n = graph.GetAmountVerts();**

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

**{**

**for (int j = 0; j < n; j++)**

**cout << setw(4) << matrix[i][j];**

**cout << endl;**

**}**

**}**

**int\*\* Reduct\_Matrix(int\*\* oldmatrix)**

**{**

**int\*\* matrix = oldmatrix;**

**int n = graph.GetAmountVerts();**

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

**matrix[i] = Search\_MinElem(matrix[i], n);**

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

**{**

**int min = 1000000;**

**for (int j = 0; j < n; j++)**

**if (matrix[j][i] >= 0 && matrix[j][i] < min)**

**min = matrix[j][i];**

**for (int j = 0; j < n; j++)**

**if (matrix[j][i] >= 0)**

**matrix[j][i] -= min;**

**}**

**if (ButtonSalesmanSecond == true)**

**{**

**cout << "\nРедуцированная матрица: \n";**

**Print\_Matrix(matrix);**

**}**

**return matrix;**

**}**

**int\*\* High\_Zero(int\*\* oldmatrix)**

**{**

**int n = graph.GetAmountVerts();**

**int\*\* matrix = Reduct\_Matrix(oldmatrix);**

**int max = -1;**

**int line = 0, column = 0;**

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

**for (int j = 0; j < n; j++)**

**{**

**if (matrix[i][j] == 0)**

**{**

**int minLine = 1000000;**

**int minColumn = 1000000;**

**for (int k = 0; k < n; k++)**

**if (matrix[i][k] != -1 && k != j && matrix[i][k] < minLine)**

**minLine = matrix[i][k];**

**for (int k = 0; k < n; k++)**

**if (matrix[k][j] != -1 && k != i && matrix[k][j] < minColumn)**

**minColumn = matrix[k][j];**

**if (max < minColumn + minLine)**

**{**

**max = minColumn + minLine;**

**line = i;**

**column = j;**

**}**

**}**

**}**

**pair<int, int> p;**

**p.first = line + 1;**

**p.second = column + 1;**

**Way.push\_back(p);**

**matrix[line][column] = -1;**

**matrix[column][line] = -1;**

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

**{**

**matrix[line][i] = -1;**

**matrix[i][column] = -1;**

**}**

**if (ButtonSalesmanSecond == true)**

**{**

**cout << endl;**

**cout << "Матрица после удаления 0 с наибольшей оценкой: \n";**

**Print\_Matrix(matrix);**

**cout << "\nПромежуточные отрезки путей: ";**

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

**cout << Way[i].first << " -> " << Way[i].second;**

**cout << endl;**

**}**

**return matrix;**

**}**

**void Print\_Result()**

**{**

**int second = Way[0].second;**

**int i = 2;**

**New\_Way.push\_back(Way[0].first);**

**New\_Way.push\_back(Way[0].second);**

**while (i != graph.GetAmountVerts() + 1)**

**for (int j = 1; j < graph.GetAmountVerts(); j++)**

**if (Way[j].first == second)**

**{**

**second = Way[j].second;**

**New\_Way.push\_back(second);**

**i++;**

**}**

**cout << "Кротчайший путь : ";**

**for (int i = 0; i < New\_Way.size(); i++)**

**{**

**cout << New\_Way[i];**

**if (i != New\_Way.size() - 1)**

**cout << " -> ";**

**}**

**cout << endl;**

**int sum = 0;**

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

**{**

**int line = Way[i].first - 1;**

**int column = Way[i].second - 1;**

**sum += graph.GetAdjMatrixElem(line, column);**

**}**

**cout << "\nДлина пути : " << sum << endl;**

**}**

**void setCoords(int i, int n)**

**{**

**int rad;**

**int x0 = WinW / 2;**

**int y0 = WinH / 2;**

**if (WinW > WinH)**

**{**

**R = 5 \* (WinH / 26) / n;**

**rad = WinH / 2 - R - 10;**

**}**

**else**

**{**

**R = 5 \* (WinW / 26) / n;**

**rad = WinW / 2 - R - 10;**

**}**

**float theta = 2.0f \* 3.1415926f \* i / n;**

**int y1 = rad \* cos(theta) + y0;**

**int x1 = rad \* sin(theta) + x0;**

**vertcrd[i].x = x1;**

**vertcrd[i].y = y1;**

**}**

**bool SalesmanPossible(int\*\* matrix)**

**{**

**if (graph.IsEmpty())**

**return false;**

**for (int i = 0; i < graph.GetAmountVerts(); i++)**

**{**

**int cnt = 0;**

**for (int j = 0; j < graph.GetAmountVerts(); j++)**

**if (matrix[i][j] > 0)**

**cnt++;**

**if (cnt < 1)**

**return false;**

**}**

**return true;**

**}**

**void drawCircle(int x, int y, int R, bool r)**

**{**

**glColor3f(1.0f, 1.0f, 1.0f);**

**float x1, y1;**

**glBegin(GL\_POLYGON);**

**for (int i = 0; i < 360; i++)**

**{**

**float theta = 2.0f \* 3.1415926f \* float(i) / float(360);**

**y1 = R \* cos(theta) + y;**

**x1 = R \* sin(theta) + x;**

**glVertex2f(x1, y1);**

**}**

**glEnd();**

**if (r)**

**glColor3f(1.0f, 0.0f, 0.0f);**

**else**

**glColor3f(0.0f, 0.0f, 0.0f);**

**float x2, y2;**

**glBegin(GL\_LINE\_LOOP);**

**for (int i = 0; i < 360; i++)**

**{**

**float theta = 2.0f \* 3.1415926f \* float(i) / float(360);**

**y2 = R \* cos(theta) + y;**

**x2 = R \* sin(theta) + x;**

**glVertex2f(x2, y2);**

**}**

**glEnd();**

**}**

**void drawCircleMouseOn(int x, int y, int R)**

**{**

**glColor3f(0.9, 0.9, 0.9);**

**float x1, y1;**

**glBegin(GL\_POLYGON);**

**for (int i = 0; i < 360; i++)**

**{**

**float theta = 2.0f \* 3.1415926f \* float(i) / float(360);**

**y1 = R \* cos(theta) + y;**

**x1 = R \* sin(theta) + x;**

**glVertex2f(x1, y1);**

**}**

**glEnd();**

**glColor3f(0.0, 0.392, 0.0);**

**float x2, y2;**

**glBegin(GL\_LINE\_LOOP);**

**for (int i = 0; i < 360; i++)**

**{**

**float theta = 2.0f \* 3.1415926f \* float(i) / float(360);**

**y2 = R \* cos(theta) + y;**

**x2 = R \* sin(theta) + x;**

**glVertex2f(x2, y2);**

**}**

**glEnd();**

**}**

**void drawText(int text, int x1, int y1, bool red)**

**{**

**if (red)**

**glColor3f(1.0, 0.0, 0.0);**

**else**

**glColor3f(0.0, 0.0, 0.0);**

**GLvoid\* font = GLUT\_BITMAP\_TIMES\_ROMAN\_24;**

**string s = to\_string(text);**

**glRasterPos2i(x1 - 5, y1 - 5);**

**for (size\_t j = 0; j < s.length(); j++)**

**glutBitmapCharacter(font, s[j]);**

**}**

**void drawLine(int text, int x0, int y0, int x1, int y1, bool b)**

**{**

**if (b)**

**glColor3f(1.0f, 0.0f, 0.0f);**

**else**

**glColor3f(0.0f, 0.0f, 0.0f);**

**glBegin(GL\_LINES);**

**glVertex2i(x0, y0);**

**glVertex2i(x1, y1);**

**glEnd();**

**if (b)**

**drawText(text, (x0 + x1) / 2 - 15, (y0 + y1) / 2 - 15, b);**

**else**

**drawText(text, (x0 + x1) / 2 + 15, (y0 + y1) / 2 + 15, b);**

**float vx = x0 - x1;**

**float vy = y0 - y1;**

**float s = 1.0f / sqrt(vx \* vx + vy \* vy);**

**vx \*= s;**

**vy \*= s;**

**x1 = x1 + R \* vx;**

**y1 = y1 + R \* vy;**

**if (b)**

**glColor3f(1.0f, 0.0f, 0.0f);**

**else**

**{**

**glColor3f(0.0f, 0.0f, 0.0f);**

**glBegin(GL\_TRIANGLES);**

**glVertex2f(x1, y1);**

**glVertex2f(x1 + 10 \* (vx + vy), y1 + 10 \* (vy - vx));**

**glVertex2f(x1 + 10 \* (vx - vy), y1 + 10 \* (vy + vx));**

**}**

**glEnd();**

**}**

**void drawVertex(int n, bool r)**

**{**

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

**{**

**if (vertmouse[i])**

**drawCircleMouseOn(vertcrd[i].x, vertcrd[i].y, R);**

**else**

**drawCircle(vertcrd[i].x, vertcrd[i].y, R, false);**

**drawText(i + 1, vertcrd[i].x, vertcrd[i].y, false);**

**}**

**}**

**Commiglut.cpp**

**#include** <iostream>

#include <GL/freeglut.h>

#include "Graph.h"

using namespace std;

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

{

setlocale(LC\_ALL, "rus");

glutInit(&argc, argv);

makeGraph();

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGBA);

glutInitWindowSize(WinW, WinH);

glutCreateWindow("Graph");

glutDisplayFunc(display);

glutReshapeFunc(reshape);

glutMouseFunc(mouseClick);

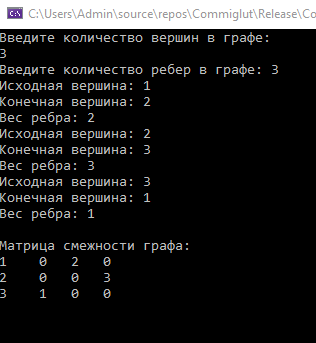
glutPassiveMotionFunc(mouseMove);

glutMainLoop();

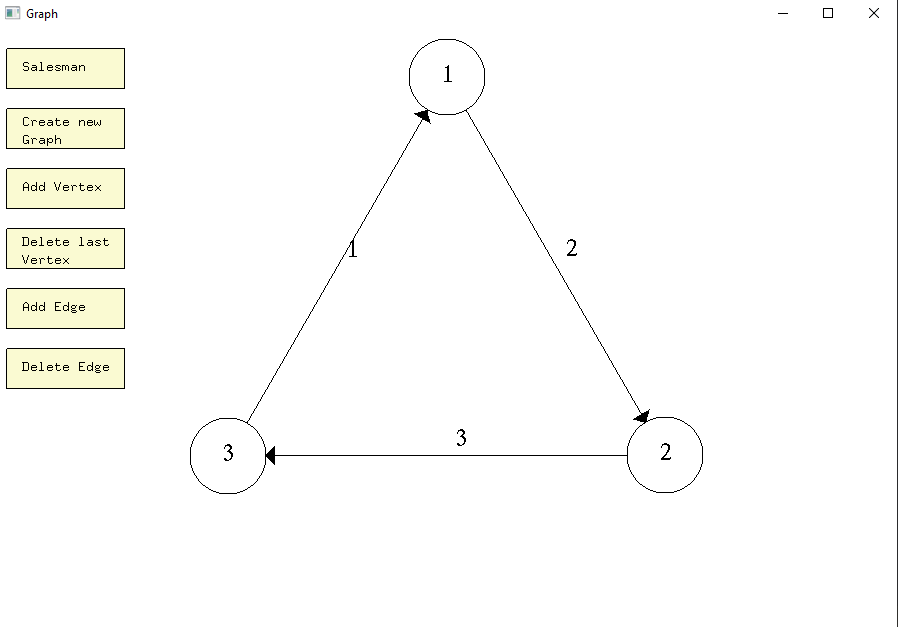
return 0;

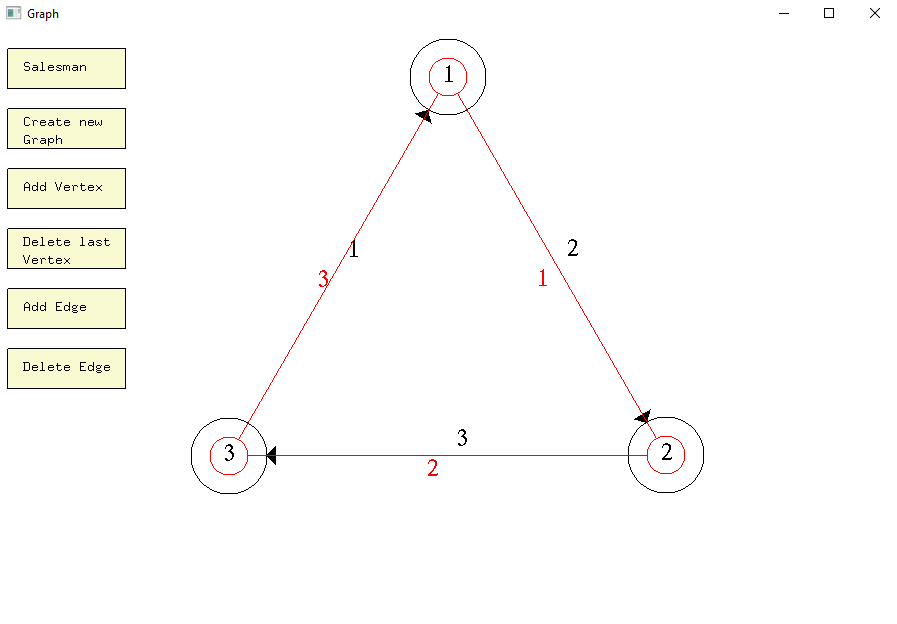
}

**Вывод программы**

****

***Рисунок 1 – вывод программы***

***Рисунок 2 – визуализация графа***

***Рисунок 3 – визуализация решения задачи***

**Вывод**

Программа выполняет свою задачу.

**Заключение**

Для решения задачи потребовались знания языка программирования, а конкретнее C++. Программа выполняет те условия, что были указаны в постановке задачи и работает без проблем.