Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Пермский национальный исследовательский**

**политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**О Т Ч Ё Т**

**по лабораторной работе №21**

Дисциплина: основы алгоритмизации и программирования

Тема: Графы. Алгоритм Дейкстры

Вариант 7

Выполнил работу

студент группы РИС-20-1б

Нечаев Д.А.

Проверил

Доцент кафедры ИТАС

Полякова О.А.

Пермь, 2021

**Цель задачи**

Реализовать динамическую структуру данных

Реализовать алгоритм Дейкстры

Написать консольное приложение для демонстрации работы программы

Реализовать графический вывод графа

Постановка задачи

Реализовать алгоритм Дейкстры для графа, соответствующего выбранному варианту.

1. Интерфейс на усмотрение разработчика.

2. Визуализация графа в OpenGL.

3. Реализуется только метод Дейкстр

Анализ задачи

1. Определить какие операции должны быть выполнены по заданию:

Создание матрицы смежности

vector <vector<int>> matrix

Добавление элементов в граф на основе очереди

QUEUE.pop();

Вывод графа

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

{

if (result[i]==1000) cout << i+1 << " - недоступен\n";

else cout << i+1 << " - " << result[i] << endl;

}

Для решения задачи используются переменные:

Несколько целочисленных переменных для управления циклами, контроля размеров массивов и контроля длины строк

int len = 0;

int adress;

people x;

string user\_search;

int error = 0;

Указатели

people\* data\_base = new people[len];

string\* key = new string[len];

1. Ввод данных осуществляется посредством функции

cin >> (F1, x.fio);

Вывод данных осуществляется посредством функции

cout << i << ": " << key[i] << endl;

1. Для решения задачи будут использованы циклы. Например, цикл для ввода информации в структуры

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

{

cout << "Город: ";

cin >> city[i].name;

cout << "Численость населения: ";

cin >> city[i].population;

}

1. Для защиты от случайных ошибок используются циклы на корректный ввод информации

while (size < 1)

{

cout << "Введите кол-во городов: ";

cin >> size;

cout << endl;

}

while (menu != 1 && menu != 2)

{

cout << "Выбери метод сортировки: \n 1. Хоара \n 2. Шелла \n Ввод: ";

cin >> menu;

}

Вывод графа осуществлен средствами QTOpenGL  
myPainter::**myPainter**()

Код Алгоритм

#include <iostream>

#include <vector>

#include <windows.h>

#include <queue>

#include <algorithm>

using namespace std;

int x;

int len=6;

int result[6]={1000,1000,1000,1000,1000,1000};

bool flag[6]={false,false,false,false,false,false};

vector <vector<int>> matrix= { {0 ,4 ,0 ,0 ,8 ,0 },

{0 ,0 ,15,9 ,0 ,0 },

{0 ,15,0 ,0 ,12,0 },

{0 ,0 ,0 ,0 ,0 ,11},

{0 ,23,12,0 ,0 ,0 },

{0 ,0 ,6 ,0 ,0 ,0 } };

bool comp(int a,int b)

{

return matrix[x][a]<matrix[x][b];

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

queue <int> QUEUE;

cout << "Введите начальную вершину (1<=x<="<<len<<")\n";

cin >> x;

x--;

result[x]=0;

QUEUE.push(x);

while (true)

{

x=QUEUE.front();

vector <int> vec;

for (int i=0;i<len;i++) if (matrix[x][i]!=0 && !flag[i]) vec.push\_back(i);

if (vec.size()==0)

{

flag[x]=true;

QUEUE.pop();

if (QUEUE.size()==0) break;

continue;

}

sort(vec.begin(),vec.end(),comp);

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

{

result[vec[i]]=min(result[vec[i]],matrix[x][vec[i]]+result[x]);

QUEUE.push(vec[i]);

}

flag[x]=true;

QUEUE.pop();

if (QUEUE.size()==0) break;

}

cout << "Кратчайшие пути:\n";

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

{

if (result[i]==1000) cout << i+1 << " - недоступен\n";

else cout << i+1 << " - " << result[i] << endl;

}

system("pause");

}

Код печати:

#include "mypainter.h"

#include <QPaintEvent>

#include <QPainter>

#include <QWidget>

#include <QtCore/qmath.h>

int len=6; *//кол-во* *узлов* *в* *графе*

int R=300; *//радиус* *графа*

int r=40; *//радиус* *узла*

double pi=3.1415;

int HeightText=30;

int xa=600; *//отступ* *от* *вертикали*

int ya=400; *//отступ* *от* *горизонтали*

double angle=pi/8;*//угол* *раствора* *стрелки*

int ls=30; *//длина* *кончика* *стрелки*

int disp=60; *//разброс* *подписей* *ребер*

int matrix[6][6]={{0 ,4 ,0 ,0 ,8 ,0 },

{0 ,0 ,15,9 ,0 ,0 },

{0 ,15,0 ,0 ,12,0 },

{0 ,0 ,0 ,0 ,0 ,11},

{0 ,23,12,0 ,0 ,0 },

{0 ,0 ,6 ,0 ,0 ,0 }};

myPainter::**myPainter**()

{

front=QBrush(QColor(127,255,212));

back=QBrush(QColor(200,200,200));

textPen=QPen(QColor(33,33,33));

glFont.setPixelSize(HeightText);

}

void myPainter::**draw**(QPainter \*painter, QPaintEvent \*event)

{

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

painter->setBrush(front);

painter->setPen(textPen);

painter->setFont(glFont);

textPen.setWidth(4);

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

{

double a=pi\*3/2+pi\*2/len\*i; *//угол* *в* *полярных* *координатах* *(узел)*

int xc=cos(a)\*R+xa; *//декартовы* *координата* *узла*

int yc=sin(a)\*R+ya;

int c1=xc-20; *//координаты* *текста*

int c2=yc-HeightText/2;

int c3=40;

int c4=HeightText;

QString text; text.setNum(i+1);

painter->drawEllipse(QRectF(xc-r,yc-r,2\*r,2\*r));

painter->drawText(QRect(c1,c2,c3,c4),Qt::*AlignCenter*,text);

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

{

*if* (matrix[i][k]!=0)

{

*//линия*

double a2=pi\*3/2+pi\*2/len\*k;

double xc2=cos(a2)\*R+xa;

double yc2=sin(a2)\*R+ya;

double l=sqrt((xc2-xc)\*(xc2-xc)+(yc2-yc)\*(yc2-yc));

int xl1=(xc2-xc)/l\*r+xc;

int yl1=(yc2-yc)/l\*r+yc;

int xl2=xc2-(xc2-xc)/l\*r;

int yl2=yc2-(yc2-yc)/l\*r;

painter->drawLine(xl1,yl1,xl2,yl2);

*//стрелка*

double b;

*if* (xc2==xl2 && yl2<yc2) b=pi\*3/2;

*else* *if* (yc2==yl2 && xl2<xc2) b=pi;

*else* *if* (xc2==xl2 && yl2>yc2) b=pi/2;

*else* *if* (yc2==yl2 && xl2>xc2) b=0;

*else* *if* (xl2>xc2 && yl2<yc2) b=atan((yc2-yl2)/(xc2-xl2))+pi\*2;

*else* *if* (xl2<xc2 && yl2<yc2) b=atan((yc2-yl2)/(xc2-xl2))+pi;

*else* *if* (xl2<xc2 && yl2>yc2) b=atan((yc2-yl2)/(xc2-xl2))+pi;

*else* *if* (xl2>xc2 && yl2>yc2) b=atan((yc2-yl2)/(xc2-xl2));

int xs1=cos(b+angle)\*ls+xl2;

int ys1=sin(b+angle)\*ls+yl2;

int xs2=cos(b-angle)\*ls+xl2;

int ys2=sin(b-angle)\*ls+yl2;

painter->drawLine(xl2,yl2,xs1,ys1);

painter->drawLine(xl2,yl2,xs2,ys2);

*//текст*

c1=(xl2+xl1)/2-20+qrand()%disp-disp/2;

c2=(yl2+yl1)/2-HeightText/2+qrand()%disp-disp/2;

c3=40;

c4=HeightText;

text.setNum(matrix[i][k]);

painter->drawText(QRect(c1,c2,c3,c4),Qt::*AlignCenter*,text);

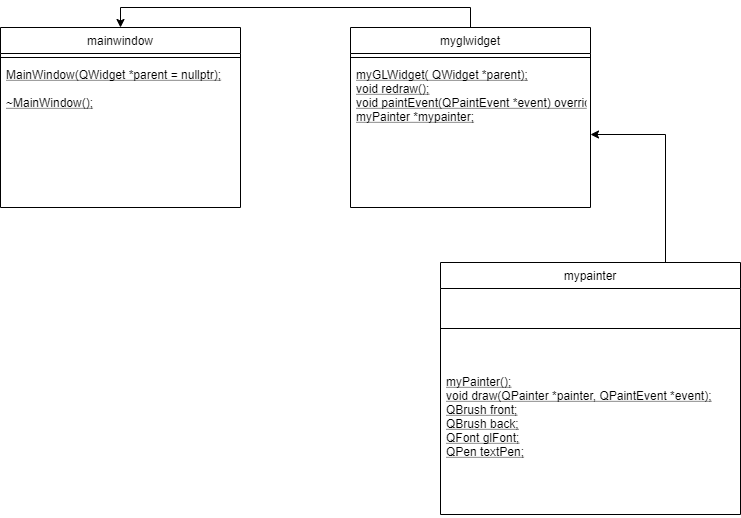
}

}

}

}

UML-диаграмма



Работа кода

