Національний технічний університет України «КПІ ім. Ігоря Сікорського»

Факультет Інформатики та Обчислювальної Техніки

Кафедра Автоматизованих Систем Обробки Інформації та Управління

Лабораторна робота №2

з дисципліни «Дискретна математика»

на тему

«**Характеристики графів**»

Варіант: 26

Виконав:

студент гр. ІС - 02

Плостак Ілля

Викладач:

доц. Рибачук Л.В.

Київ – 2020

### Зміст

[Зміст 2](#_Toc190845267)

[1 Постановка задачі 3](#_Toc190845268)

[2 Результати виконання програми 4](#_Toc190845269)

[3 Лістинг програми](#_Toc190845271) 5

### 1 Постановка задачі

**Завдання**

Реалізувати програмне застосування (програму), яке виконує наступні функції: на вхід програми подається вхідний файл з описом неорієнтованого графу, зі структурою, яка вказана у лабораторній роботі No1 «Представлення графів» (списком ребер).

1. Визначити матриці відстаней та досяжності графу. Програма за запитом

користувача виводить на екран та/або у файл матрицю відстаней D та матрицю

досяжності R графу, за алгоритмом піднесення у степінь (конспект стор. 155-156).

2. Визначення метричних характеристик графу. Програма виводить наступні

характеристики:

a. Діаметр графу

b. Радіус графу

c. Центр графу

d. Яруси графу із переліком вершин, які входять до кожного ярусу

Граф:

9 12

1 2

1 9

2 9

3 9

3 4

4 9

5 9

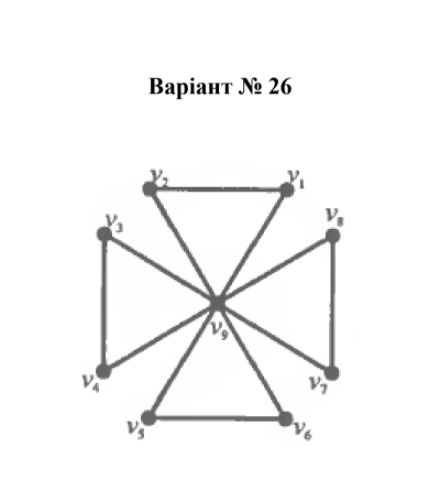
5 6

6 9

7 8

7 9

8 9



### 

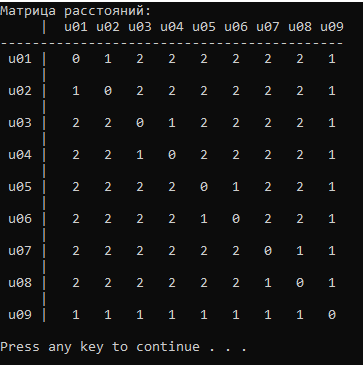
### 2 Результати виконання програми

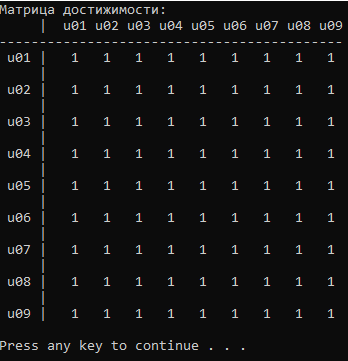
### Початок програми/пункт 7:

### 

Головне меню:

### 

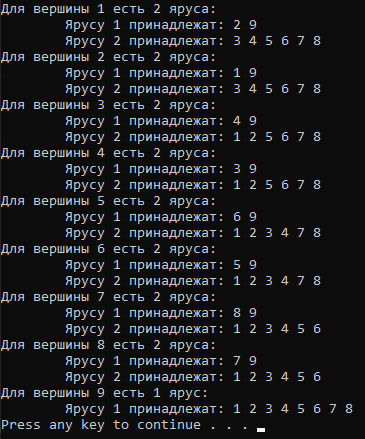
1) 

2) 

3) 

4) 

5) 

6) 

### 3 Лістинг програми

[main.cpp](https://github.com/feedblackg44/kpilabs/blob/master/LabDiskretka2/LabDiskretka2/main.cpp):

#include <iostream>

#include <fstream>

#include <sstream>

#include <math.h>

#include <limits>

#include <locale.h>

#include <iomanip>

#include <string>

#include <vector>

#include "feed.h"

#include "functions.h"

using namespace std;

int main()

{

SetDefaults();

int edge,

apex;

int\* Graph = GetGraph(&apex, &edge);

bool mode = GetBool("Введите режим в котором будет работать программа:\n1 - вывод на экран\n0 - вывод в файл", "Режим введён неправильно!", true);

const string str = "Добро пожаловать в программу работы с графом:";

const string str1 = "\n1 - Вывод матрицы расстояний";

const string str2 = "\n2 - Вывод матрицы достижимости";

const string str3 = "\n3 - Вывод диаметра графа";

const string str4 = "\n4 - Вывод радиуса графа";

const string str5 = "\n5 - Вывод центра графа";

const string str6 = "\n6 - Вывод ярусов графа с перечислением вершин";

const string str7 = "\n7 - Поменять режим вывода";

const string str0 = "\n0 - Выйти из программы";

string str\_out = str + str1 + str2 + str3 + str4 + str5 + str6 + str7 + str0;

while (true)

{

system(ClearScreen);

int choice = GetInt(str\_out.c\_str(), "Повторите ввод!");

system(ClearScreen);

switch (choice)

{

case 1:

OutputMatrix("Матрица расстояний", MatrixOfDistance(Graph, apex, edge, mode), apex, mode);

SystemPause();

break;

case 2:

MatrixDosyag(MatrixOfDistance(Graph, apex, edge, mode), apex, mode);

SystemPause();

break;

case 3:

GraphDiam(MatrixOfDistance(Graph, apex, edge, mode), apex, mode);

SystemPause();

break;

case 4:

GraphCentreNRad(MatrixOfDistance(Graph, apex, edge, mode), apex, mode, false);

SystemPause();

break;

case 5:

GraphCentreNRad(MatrixOfDistance(Graph, apex, edge, mode), apex, mode, true);

SystemPause();

break;

case 6:

GraphYarus(MatrixOfDistance(Graph, apex, edge, mode), apex, mode);

SystemPause();

break;

case 7:

mode = GetBool("Введите режим в котором будет работать программа:\n1 - вывод на экран\n0 - вывод в файл", "Режим введён неправильно!", true);

break;

case 0:

exit(1);

}

}

SystemPause();

return 0;

}

[functions.h:](https://github.com/feedblackg44/kpilabs/blob/master/LabDiskretka2/LabDiskretka2/functions.h)

#pragma once

#define FileIN "./Graph.txt"

#define FileOUT "./Output.txt"

int\* GetGraph(int\* apex, int\* edge);

void PrintGraph(int\* Graph, int size);

int\*\* MatrixOfDistance(int\* Graph, int apex, int size, bool mode);

void OutputMatrix(const char matrixName[], int\*\* matrix, int apex, bool mode);

int\*\* MatrixMult(int\*\* matrix, int apex, int power);

bool IsThereZero(int\*\* matrix, int apex);

void MatrixDosyag(int\*\* matrix, int apex, bool mode);

void GraphDiam(int\*\* matrix, int apex, bool mode);

void GraphCentreNRad(int\*\* matrix, int apex, bool mode, bool centre);

void GraphYarus(int\*\* matrix, int apex, bool mode);

[functions.cpp:](https://github.com/feedblackg44/kpilabs/blob/master/LabDiskretka2/LabDiskretka2/functions.cpp)

#include <iostream>

#include <fstream>

#include <sstream>

#include <math.h>

#include <limits>

#include <locale.h>

#include <iomanip>

#include <string>

#include <vector>

#include "feed.h"

#include "functions.h"

using namespace std;

void PrintGraph(int\* Graph, int size)

{

vector<int> GraphFirstNumbers(size);

vector<int> GraphSecondNumbers(size);

for (int i = 0; i < size \* 2; i++)

{

if (i % 2 == 0)

GraphFirstNumbers[i / 2] = Graph[i];

else

GraphSecondNumbers[i / 2] = Graph[i];

}

cout << "Вот заданный граф:" << endl;

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

{

cout << GraphFirstNumbers[i] << " " << GraphSecondNumbers[i] << endl;

}

}

int\* GetGraph(int\* apex, int\* edge)

{

ifstream inFile;

inFile.open(FileIN);

if (!inFile.is\_open())

{

cerr << "Error: Could not open the file!" << endl;

SystemPause();

exit(1);

}

string str;

istringstream iss;

getline(inFile, str);

iss.str(str);

iss >> \*apex;

iss >> \*edge;

vector<int> Graph;

int length = 0;

while (getline(inFile, str))

{

length++;

int temp1, temp2;

istringstream iss\_temp;

iss\_temp.str(str);

iss\_temp >> temp1;

iss\_temp >> temp2;

Graph.push\_back(temp1);

Graph.push\_back(temp2);

}

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

{

if (Graph[i] > \*apex)

{

cerr << "Error: one of apexes is invalid!" << endl;

SystemPause();

exit(1);

}

}

if (length != \*edge)

{

cerr << "Error: amount of edges is invalid!" << endl;

SystemPause();

exit(1);

}

int\* temp = new int[\*edge \* 2];

for (int i = 0; i < \*edge \* 2; i++)

{

temp[i] = Graph[i];

}

inFile.close();

return temp;

}

int\*\* MatrixOfDistance(int\* Graph, int apex, int size, bool mode)

{

int\*\* GraphSmezh = new int\*[apex];

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

{

GraphSmezh[i] = new int[apex];

for (int j = 0; j < apex; j++)

{

GraphSmezh[i][j] = 0;

}

}

vector<int> GraphFirstNumbers(size);

vector<int> GraphSecondNumbers(size);

for (int i = 0; i < size \* 2; i++)

{

if (i % 2 == 0)

GraphFirstNumbers[i / 2] = Graph[i];

else

GraphSecondNumbers[i / 2] = Graph[i];

}

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

{

GraphSmezh[GraphFirstNumbers[i] - 1][GraphSecondNumbers[i] - 1] = 1;

GraphSmezh[GraphSecondNumbers[i] - 1][GraphFirstNumbers[i] - 1] = 1;

}

int h = 1;

int\*\* matrix\_res = new int\* [apex];

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

{

matrix\_res[i] = new int[apex];

for (int j = 0; j < apex; j++)

{

matrix\_res[i][j] = 0;

}

}

while (IsThereZero(matrix\_res, apex))

{

int\*\* temp\_matrix = new int\* [apex];

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

{

temp\_matrix[i] = new int[apex];

for (int j = 0; j < apex; j++)

{

temp\_matrix[i][j] = 0;

}

}

temp\_matrix = MatrixMult(GraphSmezh, apex, h);

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

{

for (int j = 0; j < apex; j++)

{

if (temp\_matrix[i][j] > 0 && matrix\_res[i][j] == 0 && i != j)

{

matrix\_res[i][j] = h;

}

}

}

h++;

}

return matrix\_res;

}

void MatrixDosyag(int\*\* matrix, int apex, bool mode)

{

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

{

for (int j = 0; j < apex; j++)

{

if (i == j)

{

matrix[i][j] = 1;

}

else if (matrix[i][j] > 1)

{

matrix[i][j] = 1;

}

}

}

OutputMatrix("Матрица достижимости", matrix, apex, mode);

}

void GraphDiam(int\*\* matrix, int apex, bool mode)

{

int Max = matrix[0][0];

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

{

for (int j = 0; j < apex; j++)

{

if (Max <= matrix[i][j])

{

Max = matrix[i][j];

}

}

}

ostream\* fout;

if (mode)

fout = &cout;

else

fout = new ofstream(FileOUT);

\*fout << "Диаметр графа: " << Max << endl;

if (fout != &cout)

delete fout;

}

void GraphYarus(int\*\* matrix, int apex, bool mode)

{

ostream\* fout;

if (mode)

fout = &cout;

else

fout = new ofstream(FileOUT);

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

{

int Max = 1;

for (int j = 0; j < apex; j++)

{

if (Max <= matrix[i][j])

{

Max = matrix[i][j];

}

}

string str = Declination(" ярусa", " ярус", " ярусов", Max);

\*fout << "Для вершины " << i+1 << " есть "<< Max << str << ":";

for (int d = 0; d < Max; d++)

{

\*fout << "\n\tЯрусу " << d+1 << " принадлежат: ";

for (int j = 0; j < apex; j++)

{

if (d+1 == matrix[i][j])

{

\*fout << j+1 << " ";

}

}

\*fout << " ";

}

\*fout << endl;

}

if (fout != &cout)

delete fout;

}

void GraphCentreNRad(int\*\* matrix, int apex, bool mode, bool centre)

{

int Max = matrix[0][0];

int Centr = 1;

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

{

for (int j = 0; j < apex; j++)

{

if (Max <= matrix[i][j])

{

Max = matrix[i][j];

}

if (Max >= matrix[Centr-1][j])

{

Centr = i + 1;

}

}

}

int R;

if(Centr - 1 != 0)

R = matrix[Centr - 1][0];

else

R = matrix[Centr - 1][1];

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

{

if (R > matrix[Centr-1][i] && Centr-1 != i)

R = matrix[Centr-1][i];

}

ostream\* fout;

if (mode)

fout = &cout;

else

fout = new ofstream(FileOUT);

if (centre)

\*fout << "Центр графа: " << Centr << endl;

else

\*fout << "Радиус графа: " << R << endl;

if (fout != &cout)

delete fout;

}

void OutputMatrix(const char matrixName[], int\*\* matrix, int apex, bool mode)

{

ostream\* fout;

if (mode)

fout = &cout;

else

fout = new ofstream(FileOUT);

\*fout << matrixName << ":" << endl;

for (int i = -2; i < apex; i++)

{

if (i == -2)

{

\*fout << " | ";

for (int k = 1; k <= apex; k++)

{

if (k < 10)

\*fout << " u0" << k;

else

\*fout << " u" << k;

}

}

else if (i == -1)

{

\*fout << "-------";

}

else

{

if (i < 9)

\*fout << " u0" << i + 1 << " | ";

else

\*fout << " u" << i + 1 << " | ";

}

for (int j = -2; j <= apex + 1; j++)

{

if (i == -1 && j >= 0 && j < apex)

{

\*fout << "----";

}

else if (i >= 0 && j >= 0 && i < apex && j < apex)

{

\*fout << matrix[i][j];

if (j + 1 != apex)

\*fout << " ";

}

}

if (i >= 0 && i < apex - 1)

{

\*fout << endl;

\*fout << " |";

}

\*fout << endl;

}

\*fout << endl;

if (fout != &cout)

delete fout;

}

int\*\* MatrixMult(int\*\* matrix, int apex, int power)

{

int\*\* outMatrix = new int\* [apex];

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

{

outMatrix[i] = new int[apex];

for (int j = 0; j < apex; j++)

{

outMatrix[i][j] = matrix[i][j];

}

}

for (int b = 1; b < power; b++)

{

int\*\* tempMatrix = new int\* [apex];

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

{

tempMatrix[i] = new int[apex];

for (int j = 0; j < apex; j++)

{

tempMatrix[i][j] = outMatrix[i][j];

}

}

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

{

for (int j = 0; j < apex; j++)

{

outMatrix[i][j] = 0;

for (int k = 0; k < apex; k++)

{

outMatrix[i][j] += tempMatrix[i][k] \* matrix[k][j];

}

}

}

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

{

delete tempMatrix[i];

}

delete[] tempMatrix;

}

return outMatrix;

}

bool IsThereZero(int\*\* matrix, int apex)

{

bool result = false;

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

{

for (int j = 0; j < apex; j++)

{

if (!matrix[i][j] && i != j)

result = true;

}

}

return result;

}

[feed.h](https://github.com/feedblackg44/kpilabs/blob/master/include/feed.h):

#pragma once

#define DEFAULT\_DELAY 20 // Default delay for PrintFormatted func

#ifndef M\_PI

#define M\_PI acos(-1.0)

#endif

#ifdef \_WIN32

#include <Windows.h>

#define ClearScreen "cls"

#else

#include <unistd.h>

#define ClearScreen "clear"

#endif

#pragma warning(disable : 4996)

#ifdef max

#undef max

#endif

double RoundTo(double number, int decimalPlace);

/\*

\* Round the number to specified precision (number of digits after the decimal point)

\*/

double GetDouble(const char promptMessage[], const char failMessage[], bool clr = false);

/\*

\* Get double number from user in console using promptMessage and failMessage

\*/

int GetInt(const char promptMessage[], const char failMessage[], bool clr = false);

/\*

\* Get integer number from user in console using promptMessage and failMessage

\*/

bool GetBool(const char promptMessage[], const char failMessage[], bool clr = false);

/\*

\* Get bool value from user in console using promptMessage and failMessage

\*/

double Minimum(double number1, double number2);

/\*

\* Return minimum number of two numbers

\*/

void SetDefaults();

/\*

\* Everything that need to be done at the start of the program

\*/

void PrintFormatted(const char name[], double number, int width, int precision, bool align);

/\*

\* Formatted printing.

\*

\* name - number's name to show

\* number - number itself

\* width - number of digits to show (0 to igonre)

\* precision - number of digits after decimal point (0 to igonre)

\* align - whether to align numbers by a decimal point

\*/

void PrintSlow(const char str[], int delay, bool endline);

/\*

\* Slow animated-like printing.

\*

\* str - string to print

\* delay - delay between printing symbols (milliseconds)

\* endline - whether to print an endline symbol

\*/

void CoutReset();

/\*

\* Reset current output stream flags

\*/

void SleepFor(int milliseconds);

/\*

\* Program sleep time in milliseconds (same for Linux and Win)

\*/

void ClearScr();

/\*

\* Clear screen in milliseconds (same for Linux and Win)

\*/

void SystemPause();

/\*

\* Cross-platform system pause

\*/

void BirthdayParty();

/\*

\* :)

\*/

void LabHeader(int index);

/\*

\* Show index number of lab work, name and group

\*/

const char\* Declination(const char pluralWord234[], const char singleWord[], const char pluralWord[], int amount);

/\*

\* Get correct word form in dependance of its amount

\*/

int RandomInInterval(int bottom, int upper);

/\*

\* Get random number from selected gap

\*/

[feed.cpp](https://github.com/feedblackg44/kpilabs/blob/master/include/feed.cpp):

#include <iostream>

#include <cmath>

#include <limits>

#include <locale>

#include <iomanip>

#include <string>

#include <cstring>

#include <ctime>

#include "feed.h"

using namespace std;

void SetDefaults()

{

setlocale(LC\_ALL, "ru-RU.UTF-8");

BirthdayParty();

}

double RoundTo(double number, int decimalPlace)

{

double D = round(number \* pow(10, decimalPlace)) / pow(10, decimalPlace);

return D;

}

double GetDouble(const char promptMessage[], const char failMessage[], bool clr)

{

double d\_Number;

while (true)

{

cout << promptMessage << "\n";

cin >> d\_Number;

if (clr)

system(ClearScreen);

if (cin.fail())

{

cout << failMessage << "\n";

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

}

else

{

return d\_Number;

}

}

}

int GetInt(const char promptMessage[], const char failMessage[], bool clr)

{

float f\_Number;

while (true)

{

cout << promptMessage << "\n";

cin >> f\_Number;

if (clr)

system(ClearScreen);

if (cin.fail())

{

cout << failMessage << "\n";

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

}

else

{

if (round(f\_Number) == f\_Number)

return round(f\_Number);

else

cout << failMessage << "\n";

}

}

}

bool GetBool(const char promptMessage[], const char failMessage[], bool clr)

{

float f\_Number;

while (true)

{

cout << promptMessage << "\n";

cin >> f\_Number;

if (clr)

system(ClearScreen);

if (cin.fail())

{

cout << failMessage << "\n";

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

}

else

{

if (f\_Number == 1 || f\_Number == 0)

return bool(f\_Number);

else

cout << failMessage << "\n";

}

}

}

double Minimum(double number1, double number2)

{

if (number1 < number2)

return number1;

else

return number2;

}

void PrintFormatted(const char name[], double number, int width, int precision, bool align)

{

cout << name << " = ";

if (align)

cout << fixed;

else

cout << defaultfloat;

if (width > 0)

cout << setw(width);

if (precision > 0)

cout << setprecision(precision);

cout << number << endl;

}

void PrintSlow(const char str[], int delay, bool endline)

{

int count = strlen(str);

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

{

cout << str[i];

SleepFor(delay);

}

if (endline)

cout << endl;

}

void CoutReset()

{

cout << defaultfloat;

}

void SleepFor(int milliseconds)

{

#ifdef \_WIN32

Sleep(milliseconds);

#else

milliseconds \*= 1000;

usleep(milliseconds);

#endif

}

void ClearScr()

{

system(ClearScreen);

}

void SystemPause()

{

#ifdef \_WIN32

system("pause");

#else

cout << "Press any key to continue . . .";

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cin.get();

cout << endl;

#endif

}

void BirthdayParty()

{

time\_t t = time(NULL);

tm\* timePtr = localtime(&t);

if ((timePtr->tm\_mday == 3) && (timePtr->tm\_mon + 1 == 12))

{

cout << "Давайте поздравим с Днём Рождения разработчика - Плостака Илью :)\n" << endl;

SystemPause();

ClearScr();

}

}

void LabHeader(int index)

{

cout << "Лабораторная работа №" << index << endl;

cout << "Выполнил: Плостак Илья" << endl;

cout << "Группа: ИС-02\n" << endl;

}

const char\* Declination(const char pluralWord234[], const char singleWord[], const char pluralWord[], int amount)

{

const char\* resultWord;

if (amount % 10 <= 4 && amount % 10 >= 2 && (amount % 100 < 12 || amount % 100 > 14))

resultWord = pluralWord234;

else if (amount % 10 == 1 && amount % 100 != 11)

resultWord = singleWord;

else

resultWord = pluralWord;

return resultWord;

}

int RandomInInterval(int bottom, int upper)

{

srand(time(NULL));

int output = rand() % (upper - bottom) + bottom;

return output;

}