Національний технічний університет України «КПІ ім. Ігоря Сікорського»

Факультет Інформатики та Обчислювальної Техніки

Кафедра Автоматизованих Систем Обробки Інформації та Управління

Лабораторна робота №3

з дисципліни «Дискретна математика»

на тему

«**Обхід 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. Причому на вхід програми подається вхідний файл з описом графу, зі структурою, яка вказана у практичному завданні No1 «Представлення графів». При реалізації алгоритмів вважати, що заданий граф є зв’язаним.

1. Обійти граф пошуком вшир. Користувач вводить початкову вершину графу.

Програма виконує обхід графу, починаючи з вказаної початкової вершини. На

екран виводиться протокол обходу – таблиця, яка містить наступні дані по кожній ітерації алгоритму обходу: поточна вершина, її BFS-номер, вміст черги (див. тему 28 електронного конспекту).

2. Обійти граф пошуком углиб. Аналогічно за пунктом 1 завдання, але програма виконує обхід графу пошуком углиб. На екран виводиться протокол обходу: поточна вершина, її DFS-номер, вміст стеку.

Граф:

9 12

1 2

1 9

2 9

3 9

3 4

4 9

5 9

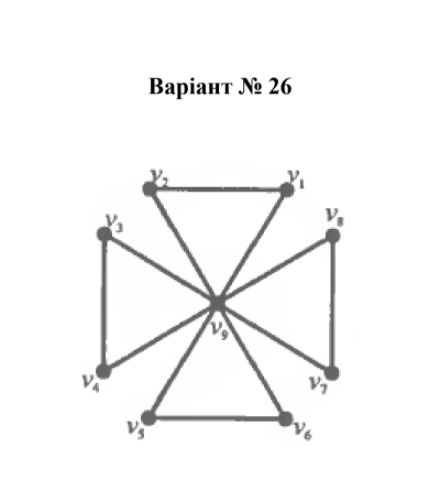
5 6

6 9

7 8

7 9

8 9



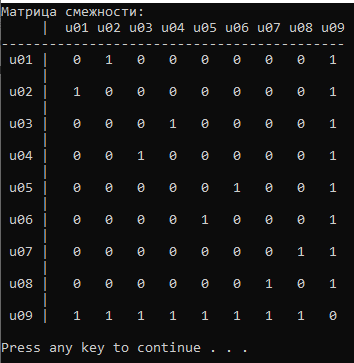
### 

### 2 Результати виконання програми

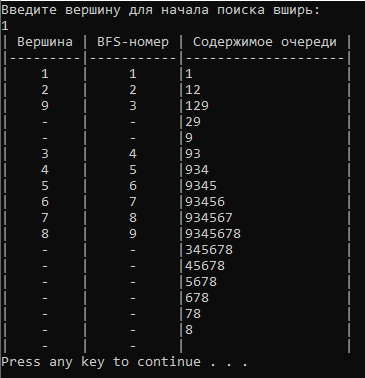
Головне меню:

### 

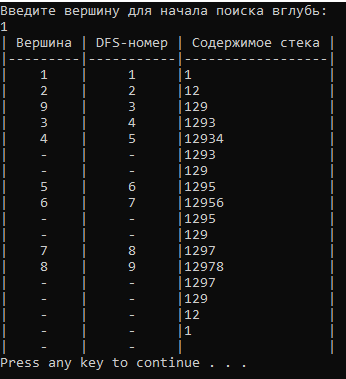
1) Матриця суміжності:



2) Пошук вшир (приклад для 1 вершини):



3)Пошук вглуб (приклад для 1 вершини):



### 3 Лістинг програми

[main.cpp](https://github.com/feedblackg44/kpilabs/blob/master/LabDiskretka3/LabDiskretka3/main.cpp):

#include <iostream>

#include <math.h>

#include <limits>

#include <locale.h>

#include <iomanip>

#include "feed.h"

#include "functions.h"

using namespace std;

int main()

{

SetDefaults();

int edge,

apex;

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

const char\* str\_menu = "Добро пожаловать в программу работы с графом:"

"\n1 - Вывод матрицы смежности"

"\n2 - Обход графа поиском вширь"

"\n3 - Обход графа поиском вглубь"

"\n0 - Выйти из программы";

while (true)

{

system(ClearScreen);

int choice = GetInt(str\_menu, "");

system(ClearScreen);

int\*\* matrix = MatrixSmezh(Graph, apex, edge, NON\_ORIENTED);

switch (choice)

{

case 1:

OutputMatrix("Матрица смежности", matrix, apex);

SystemPause();

break;

case 2:

ObhodVShir(matrix, apex);

SystemPause();

break;

case 3:

ObhodVGlub(matrix, apex);

SystemPause();

break;

case 0:

exit(0);

}

}

}

[functions.h:](https://github.com/feedblackg44/kpilabs/blob/master/LabDiskretka3/LabDiskretka3/functions.h)

#pragma once

#define FileIN "./Graph.txt"

#define FileOUT "./Output.txt"

#define ORIENTED true

#define NON\_ORIENTED false

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

int\*\* MatrixSmezh(int\* Graph, int apex, int size, bool oriented);

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

void ObhodVShir(int\*\* matrix, int apex);

void ObhodVGlub(int\*\* matrix, int apex);

bool IsTherePossibleEdge(int\*\* matrix, int DFS[], int apex, int\* from);

[functions.cpp:](https://github.com/feedblackg44/kpilabs/blob/master/LabDiskretka3/LabDiskretka3/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;

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\*\* MatrixSmezh(int\* Graph, int apex, int size, bool oriented)

{

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;

if (!oriented)

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

}

return GraphSmezh;

}

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

{

cout << matrixName << ":" << endl;

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

{

if (i == -2)

{

cout << " | ";

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

{

if (k < 10)

cout << " u0" << k;

else

cout << " u" << k;

}

}

else if (i == -1)

{

cout << "-------";

}

else

{

if (i < 9)

cout << " u0" << i + 1 << " | ";

else

cout << " u" << i + 1 << " | ";

}

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

{

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

{

cout << "----";

}

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

{

cout << matrix[i][j];

if (j + 1 != apex)

cout << " ";

}

}

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

{

cout << endl;

cout << " |";

}

cout << endl;

}

cout << endl;

}

void ObhodVShir(int\*\* matrix, int apex)

{

int u;

while (true)

{

u = GetInt("Введите вершину для начала поиска вширь:", "Вершина введена неправильно!");

if (u <= apex && u >= 1)

{

break;

}

else

{

system(ClearScreen);

cout << "Вершина введена неправильно!";

}

}

int\* BFS = new int[apex];

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

BFS[i] = 0;

vector<int> Q = {u};

cout << "| Вершина | BFS-номер | Содержимое очереди |" << endl;

cout << "|---------|-----------|--------------------|" << endl;

int k = 1;

while (Q.size() > 0)

{

int v = Q[0]-1;

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

{

if (k == 1)

{

BFS[u - 1] = k;

cout << "| " << u << " | " << k << " |";

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

cout << Q[j];

for (int j = 0; j < (20 - Q.size()); j++)

cout << " ";

cout << "|" << endl;

k++;

}

if (!BFS[i-1] && matrix[v][i-1])

{

BFS[i-1] = k;

Q.push\_back(i);

cout << "| " << i << " | " << k << " |";

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

cout << Q[j];

for (int j = 0; j < (20 - Q.size()); j++)

cout << " ";

cout << "|" << endl;

k++;

}

}

Q.erase(Q.begin());

cout << "| " << "-" << " | " << "-" << " |";

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

cout << Q[j];

for (int j = 0; j < (20 - Q.size()); j++)

cout << " ";

cout << "|" << endl;

}

}

void ObhodVGlub(int\*\* matrix, int apex)

{

int u;

while (true)

{

u = GetInt("Введите вершину для начала поиска вглубь:", "Вершина введена неправильно!");

if (u <= apex && u >= 1)

{

break;

}

else

{

system(ClearScreen);

cout << "Вершина введена неправильно!";

}

}

int\* DFS = new int[apex];

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

DFS[i] = 0;

vector<int> S = { u };

cout << "| Вершина | DFS-номер | Содержимое стека |" << endl;

cout << "|---------|-----------|------------------|" << endl;

int k = 1;

DFS[u - 1] = k;

cout << "| " << u << " | " << k << " |";

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

cout << S[j];

for (int j = 0; j < (18 - S.size()); j++)

cout << " ";

cout << "|" << endl;

k++;

bool done = false;

while (S.size() > 0)

{

int v = S.back() - 1;

int current = v;

if (IsTherePossibleEdge(matrix, DFS, apex, &current))

{

DFS[current-1] = k;

S.push\_back(current);

cout << "| " << current << " | " << k << " |";

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

cout << S[j];

for (int j = 0; j < (18 - S.size()); j++)

cout << " ";

cout << "|" << endl;

k++;

}

else

{

S.pop\_back();

cout << "| " << "-" << " | " << "-" << " |";

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

cout << S[j];

for (int j = 0; j < (18 - S.size()); j++)

cout << " ";

cout << "|" << endl;

}

}

}

bool IsTherePossibleEdge(int\*\* matrix, int DFS[], int apex, int \*from)

{

bool result = false;

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

{

if (matrix[\*from][i] && !DFS[i])

{

result = true;

\*from = i+1;

break;

}

}

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;

}