Міністерство освіти і науки України

Черкаський державний технологічний університет

Кафедра програмного забезпечення автоматизованих систем

Звіт

про виконання лабораторної роботи №4

з дисципліни «Проектний практикум»

|  |  |
| --- | --- |
| Перевірив:  Асистент кафедри ПЗАС  Півень О.Б. | Виконав:  студент 3-го курсу  групи ПЗ-1644  Близнюк А. О. |

Черкаси 2017

**Тема роботи:** Графи. Пошук в глибину.

**Мета роботи**: Ознайомлення з графами та методом пошуку в глибину.

**Теоретичні відомості**

Граф — це сукупність об'єктів із зв'язками між ними.

Об'єкти розглядаються як вершини, або вузли графу, а зв'язки — як дуги, або ребра. Для різних областей використання види графів можуть відрізнятися орієнтовністю, обмеженнями на кількість зв'язків і додатковими даними про вершини або ребра.

Велика кількість структур, які мають практичну цінність в [математиці](https://uk.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0) та [інформатиці](https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0), можуть бути представлені графами.

Алгоритм пошуку в глибину  — [алгоритм](https://uk.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC) для обходу [дерева](https://uk.wikipedia.org/wiki/%D0%94%D0%B5%D1%80%D0%B5%D0%B2%D0%BE_(%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85)), структури подібної до дерева, або [графа](https://uk.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)). Робота алгоритма починається з кореня дерева (або іншої обраної вершини в графі) і здійснюється обхід в максимально можливу глибину до переходу на наступну вершину.

Наведемо кроки алгоритму:

1. Почати з довільної вершини v. Виконати DFS(v):=1. Включити цю вершину в стек.

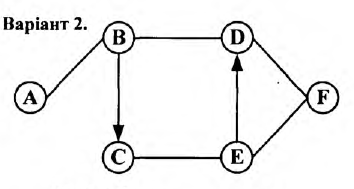
2. Розглянути вершину у верхівці стеку: нехай це вершина х. Якщо всі ребра, інцидентні вершині х, позначено, то перейти до кроку 4, інакше - до кроку 3.

3. Нехай {x,y} - непозначене ребро. Якщо DFS(у) уже визначено, то позначимо ребро {x,y} потовщено суцільною лінією, визначити DFS(у) як черговий DFS-номер, включити цю вершину в стек і перейти до кроку 2.

Виключити вершину х зі стеку. Якщо стек порожній, то зупинитись, інакше - перейти до кроку 2.

**Завдання:**

Створити програму, яка реалізує пошук шляху між двома довільними вершинами графа згідно варіанту. Номери вершин для пошуку шляху між ними користувач повинен ввести самостійно. Для отримання ойінки відмінно реалізувати граф за допомогою списків суміжності.



#include <iostream>

#include <cstdlib>

using namespace std;

const int n = 7;

int Graph[n][n]={{0,0,0,0,0,0,0},//0

{0,0,1,0,0,0,0},//1

{0,1,0,1,1,0,0},//2

{0,0,0,0,0,1,0},//3

{0,0,1,0,0,0,1},//4

{0,0,0,1,1,0,1},//5

{0,0,0,0,1,1,0}};//6

int mark[n]={0};

int prior[n]={0};

int start = 0;

int finish = 0;

void DFS(int v, int from){

if(mark[v]!=0) return;

mark[v] = 1;

prior[v] = from;

if(v == finish){

cout<<"Done!\n"; return;

}

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

if(Graph[v][i]!=0) DFS(i,v);

return;

}

void getPath(){

int path[n]={0};

int i=0;

for(int v = finish; v!=start; v=prior[v]){

if(v == 0){

cout<<"Not done!\n"; return;

}

path[i] = v;

i++;

}

path[i] = start;

while(i>=0){

cout<<path[i]<<" > ";

i--;

}

return;

}

int main()

{

setlocale(LC\_ALL, "Russian");

while(1) {

cout<<"\n1. Matrix symignosti\n2. Obrobka\n";

int key; cin>>key; cout<<endl;

switch(key){

case 1:

for(int i=1;i<n;i++){

for(int j=1;j<n;j++){

cout<<"["<<Graph[i][j]<<"] ";

}

cout<<endl;

}

break;

case 2:

cout<<"start: "; cin>>start;

cout<<"finish: "; cin>>finish;

if(start<1 || start>6 || finish<1 || finish>6) cout<<"Некоректно введено пошук шляху\n";

else {

cout<<"\n";

DFS(start, 0);

cout<<"\n";

getPath();

}

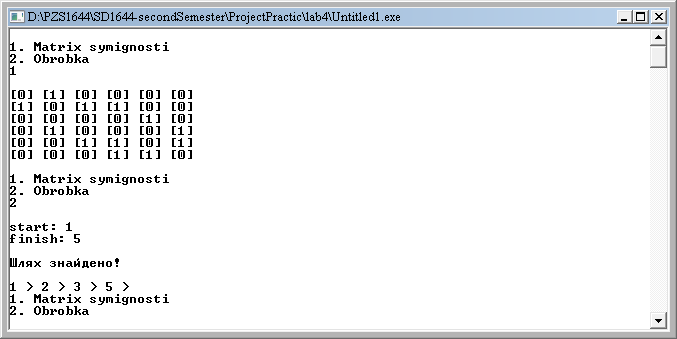
break;

}

}

return 0;

}



Висновки: Під час лабораторної роботи було освоєно навички роботи з графами та алгоритм пошуку в глибину.