Міністерство освіти і науки України

Національний технічний університет України

“Київський політехнічний інститут ім. Ігоря Сікорського”

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації та управління

ЗВІТ

про виконання лабораторного практикуму №6(1)

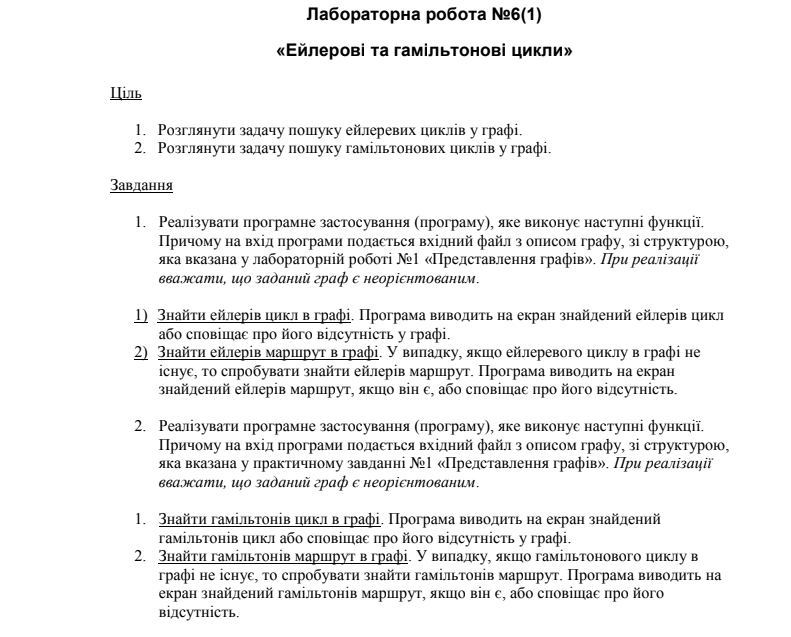
Виконав:

студент 1- го курсу ФІОТ

групи *ІП-91*

*Кінчур Вадим Вікторович*

Київ 2020

2. Умова лабораторної роботи

3. Програмний код (С++):

#include <iostream>

#include <stack>

#include <fstream>

#include <vector>

#include <numeric>

#include <list>

using namespace std;

template<typename T>

T\*\* fill\_matrix(const int, const int, const T);

template<typename T>

void delete\_matrix(T\*\*, const int);

void print\_vector(vector<int>);

void get\_input(const char\*, int\*\*&, int&);

int\* get\_degrees(int\*\*, const int);

bool euler\_exist(int\*, const int);

vector<int> get\_euler(int\*\*, const int);

int get\_adjacent(int\*\*, const int, const int);

void print\_euler(int\*\*, const int);

list<int>\* get\_adj\_list(int\*\*, const int);

vector<int> try\_to\_make\_cycle(vector<int>, list<int>\*);

vector<int> get\_hamilton(int\*\*, const int);

void print\_hamilton(int\*\*, const int);

void show\_menu();

void process\_menu(int\*\*, const int);

int main() {

int\*\* matrix = nullptr;

int vertex\_num;

get\_input("input.txt", matrix, vertex\_num);

process\_menu(matrix, vertex\_num);

delete\_matrix(matrix, vertex\_num);

}

template<typename T>

T\*\* fill\_matrix(const int row\_size, const int col\_size, const T filler) {

T\*\* matrix = new T\* [row\_size];

for (auto i = 0; i < row\_size; ++i) {

matrix[i] = new T[col\_size];

}

for (auto i = 0; i < row\_size; ++i) {

for (auto j = 0; j < col\_size; ++j) {

matrix[i][j] = filler;

}

}

return matrix;

}

template<typename T>

void delete\_matrix(T\*\* matrix, const int row\_size) {

for (auto i = 0; i < row\_size; ++i)

delete[] matrix[i];

delete[] matrix;

}

void print\_vector(vector<int> vec) {

for (auto el : vec) {

cout << el << " ";

}

cout << '\n';

}

void get\_input(const char\* file\_name, int\*\*& matrix, int& v\_num){

ifstream file\_handler(file\_name);

int e\_num;

file\_handler >> v\_num >> e\_num;

matrix = fill\_matrix(v\_num, v\_num, 0);

for (auto i = 0; i < e\_num; ++i) {

int start, end;

file\_handler >> start >> end;

matrix[start - 1][end - 1] += 1;

matrix[end - 1][start - 1] += 1;

}

file\_handler.close();

}

int\* get\_degrees(int\*\* matrix, const int size) {

int\* degrees = new int[size];

for (auto i = 0; i < size; ++i) {

degrees[i] = accumulate(matrix[i], matrix[i] + size, 0);

}

return degrees;

}

bool euler\_exist(int\* degrees, const int vertex\_number) {

int counter = 0;

for (auto i = 0; i < vertex\_number; ++i) {

counter += (degrees[i] & 1);

if (counter > 2) return false;

}

return true;

}

int get\_adjacent(int\*\* matrix, const int v, const int size){

for (auto i = 0; i < size; ++i) {

if (matrix[v][i] > 0) return i;

}

return size;

}

int get\_start(int\* degrees, const int v\_num){

for (auto i = 0; i < v\_num; ++i)

if (degrees[i]) return i;

}

void print\_euler(int\*\* matrix, const int v\_num) {

auto path = get\_euler(matrix, v\_num);

if (path.size() == 0) {

cout << "Euler path and circuit don`t exist!\n";

}

else{

if (path.front() == path.back())

cout << "Euler circuit : \n";

else

cout << "Euler path : \n";

print\_vector(path);

}

}

vector<int> get\_euler(int\*\* matrix, const int v\_num){

auto\* deg = get\_degrees(matrix, v\_num);

if (!euler\_exist(deg, v\_num))

return vector<int>();

stack<int> stack;

vector<int> path;

auto start = get\_start(deg, v\_num);

stack.push(start);

while (!stack.empty()) {

const auto v = stack.top();

const auto index = get\_adjacent(matrix, v, v\_num);

if (index == v\_num) {

stack.pop();

path.push\_back(v + 1);

}

else {

--matrix[v][index];

--matrix[index][v];

stack.push(index);

}

}

delete[] deg;

return path;

}

list<int>\* get\_adj\_list(int\*\* matrix, const int size) {

auto\* adj\_list = new list<int>[size];

for (auto i = 0; i < size; ++i) {

for (auto j = 0; j < size; ++j) {

if (matrix[i][j]) {

adj\_list[i].push\_back(j);

}

}

}

return adj\_list;

}

vector<int> try\_to\_make\_cycle(vector<int> path, list<int>\* adj\_list) {

for (auto w : adj\_list[path.back() - 1]) {

if (w == path.front() - 1) {

path.push\_back(w + 1);

break;

}

}

return path;

}

vector<int> get\_hamilton(int\*\* matrix, const int v\_num){

auto\* adj\_list = get\_adj\_list(matrix, v\_num);

bool\* visited = new bool[v\_num];

int start = 0;

vector<int> path;

for (auto i = 0; i < v\_num; ++i)

visited[i] = false;

visited[start] = true;

path.push\_back(start + 1);

while (path.size() != 0 && path.size() != v\_num) {

start = path.back() - 1;

for (auto w : adj\_list[start]) {

if (!visited[w]) {

path.push\_back(w + 1);

visited[w] = true;

break;

}

}

if (path.back() - 1 == start) {

path.pop\_back();

}

}

if (path.size())

path = try\_to\_make\_cycle(path, adj\_list);

delete[] visited;

delete[] adj\_list;

return path;

}

void print\_hamilton(int\*\* matrix, const int size){

auto path = get\_hamilton(matrix, size);

if (!path.size()) {

cout << "Hamiltonian path and cycle don`t exist!\n";

}

else {

if (path.front() == path.back()) {

cout << "Hamiltonian cycle : \n";

}

else {

cout << "Hamiltonuan path : \n";

}

print\_vector(vector<int>(path));

}

}

void show\_menu(){

cout << "Choose one of the following : \n"

<< "1 - print Eulerian path or circuit\n"

<< "2 - print Hamiltonian path or circuit\n";

}

void process\_menu(int\*\* matrix, const int v\_num) {

int choice = -1;

show\_menu();

while (choice < 0 || choice > 2) {

cout << "\nType you choice : ";

cin >> choice;

}

if (choice == 1) {

print\_euler(matrix, v\_num);

}

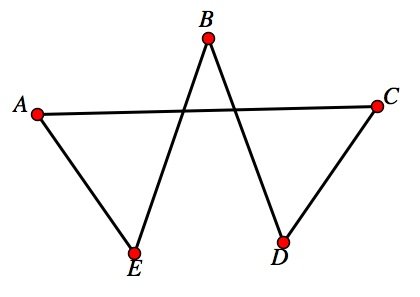
else {

print\_hamilton(matrix, v\_num);

}

}

4. Результати виконання :



**Вхідні дані :**

5 5

1 3

1 4

2 4

2 5

3 5

