ЗВІТ

про виконання лабораторної роботи №6

«Представлення графів»

з дисципліни «Програмування дискретних структур»

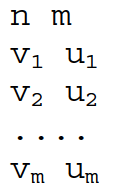
Студента групи КН-2327Б

Сухар Станіслав

Мета роботи: Розглянути та вивчити різні типи представлення графів в пам'яті комп'ютерів

Завдання: Реалізувати програму, яка виконує наступні функції.

1. Зчитування графу з вхідного файлу. На вхід подається текстовий файл наступного вигляду



Тут *n -* кількість вершин графу (ціле число, більше нуля), *m -* кількість ребер графу (ціле число, більше нуля), *vi* та *ui -* початкова та кінцева вершина ребра *i* ( 1≤*vi*≤*n* , 1≤*ui*≤*n,* цілі числа). Індексація вершин у файлі ведеться з 1. Вважається, що граф є орієнтованим.

Таким чином можна сказати, що граф задається у файлі списком ребер.

1. Вивід матриць інцидентності та суміжності. За вимогою користувача програма

повинна виводити матриці інцидентності та суміжності (окремі функції) на екран та у текстовий файл, який вказує користувач.

1. Текст програми:

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

void readGraph(const string& filename, int& n, int& m, int\*\*& edges) {

cout << "Шлях до файлу: " << filename << endl;

ifstream inputFile(filename);

if (!inputFile.is\_open()) {

cout << "Помилка при вiдкриттi файлу: " << filename << endl;

return;

}

inputFile >> n >> m;

edges = new int\* [m];

for (int i = 0; i < m; ++i) {

edges[i] = new int[2];

inputFile >> edges[i][0] >> edges[i][1];

}

inputFile.close();

}

int\*\* createAdjacencyMatrix(int n, int m, int\*\* edges) {

int\*\* adjacencyMatrix = new int\* [n];

for (int i = 0; i < n; ++i) {

adjacencyMatrix[i] = new int[n]();

}

for (int i = 0; i < m; ++i) {

int v = edges[i][0] - 1;

int u = edges[i][1] - 1;

adjacencyMatrix[v][u] = 1;

}

return adjacencyMatrix;

}

int\*\* createIncidenceMatrix(int n, int m, int\*\* edges) {

int\*\* incidenceMatrix = new int\* [n];

for (int i = 0; i < n; ++i) {

incidenceMatrix[i] = new int[m]();

}

for (int i = 0; i < m; ++i) {

int v = edges[i][0] - 1;

int u = edges[i][1] - 1;

incidenceMatrix[v][i] = 1;

incidenceMatrix[u][i] = -1;

}

return incidenceMatrix;

}

void printMatrix(int\*\* matrix, int rows, int cols, const string& matrixName, const string& outputFile) {

ofstream outFile(outputFile, ios::app);

cout << matrixName << ":\n";

outFile << matrixName << ":\n";

for (int i = 0; i < rows; ++i) {

for (int j = 0; j < cols; ++j) {

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

outFile << matrix[i][j] << " ";

}

cout << endl;

outFile << endl;

}

cout << endl;

outFile << endl;

outFile.close();

}

void deleteMatrix(int\*\* matrix, int rows) {

for (int i = 0; i < rows; ++i) {

delete[] matrix[i];

}

delete[] matrix;

}

int main() {

setlocale(LC\_CTYPE, "ukr");

int n, m;

int\*\* edges = nullptr;

string inputFile = "pds.txt";

string outputFile = "pds1.txt";

cout << "Введiть назву файлу з вхiдними даними (за замовчуванням 'pds.txt'): ";

getline(cin, inputFile);

if (inputFile.empty()) inputFile = "pds.txt";

cout << "Введiть назву файлу для збереження результату (за замовчуванням 'pds1.txt'): ";

getline(cin, outputFile);

if (outputFile.empty()) outputFile = "pds1.txt";

readGraph(inputFile, n, m, edges);

if (edges == nullptr) {

cout << "Помилка: данi графа не були завантаженi." << endl;

return 1;

}

int\*\* adjacencyMatrix = createAdjacencyMatrix(n, m, edges);

int\*\* incidenceMatrix = createIncidenceMatrix(n, m, edges);

printMatrix(adjacencyMatrix, n, n, "Матриця сумiжностi", outputFile);

printMatrix(incidenceMatrix, n, m, "Матриця iнцидентностi", outputFile);

deleteMatrix(adjacencyMatrix, n);

deleteMatrix(incidenceMatrix, n);

for (int i = 0; i < m; ++i) {

delete[] edges[i];

}

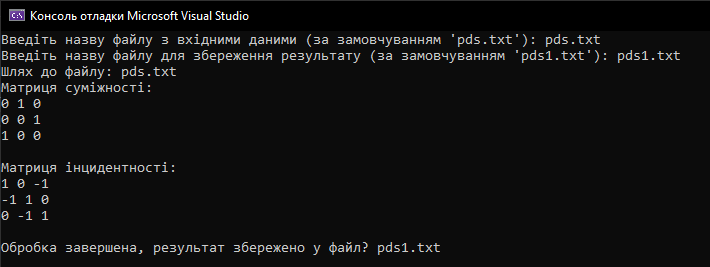
delete[] edges;

cout << "Обробка завершена, результат збережено у файлі " << outputFile << endl;

return 0;

}

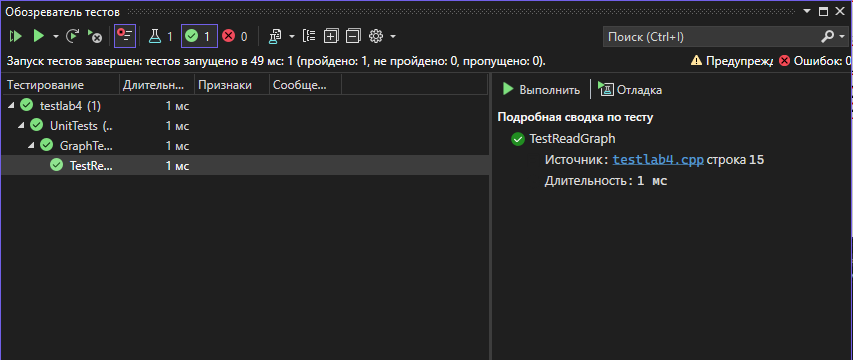
1. Результат виконання коду:



1. UnitTest



1. Результат UnitTest:



1. Побудова cтруктурної діаграми:

Зображення, що містить текст, квитанція

Автоматично згенерований опис

Посилання на Git-репозиторій: <https://github.com/stasyambitch/PDS_LAB6>

Висновок:На даній лабораторній роботі, я поглибив і закріпив розуміння основних понять комбінаторного аналізу, сформував навички складання алгоритмів та програм комбінаторного аналізу.