МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М. Є. Жуковського

«Харківський авіаційний інститут»

факультет програмної інженерії та бізнесу

кафедра інженерії програмного забезпечення

**Лабораторна робота № 5**

з дисципліни « Алгоритми і структури даних »

*назва дисципліни*

на тему: «ГРАФИ. ФОРМИ ПОДАННЯ ГРАФІВ В ЕОМ»

Виконав: студент 2 курсу групи № 622п

освітньої програми

121 інженерія програмного забезпечення

(шифр і назва ОП)

Зайченко Ярослав Ігорович

(прізвище й ініціали студента)

Прийняв: канд. техн. наук, доцент на кафедрі 603 Волобуєва Ліна Олексіївна

(посада, науковий ступінь, прізвище й ініціали)

Кількість балів:

Харків – 2023

**Мета роботи** − вивчення різних форм подання графів в ЕОМ

Постановка задачі

1. Розробити програму, яка дозволяє:

* вводити граф з клавіатури;
* введення графа з текстового файлу;
* перевіряє числа вершин 𝑛 ≤ 20 і числом зв’язків 𝑚 ≤ 50 у зовнішнім форматі згідно з варіантом і перетворює у внутрішнє представлення.
* виводить на екран/форму внутрішнє представлення графа;

1. Розроблена програма має бути налагоджена і протестована на різних комбінаціях коректних і некоректних даних, які виявляються в процесі створення програми, а також аналізу можливих аномалій, пов’язаних з форматом вхідних даних.

Зміст звіту

Зміст звіту з лабораторної роботи №5 має включати наступні підрозділи:

1. Титульний аркуш
2. Постановку задачі в вказанням і описом варіанту
3. Теоретичні відомості повинні містити опис формату подання згідно з варіантом, рисунок графа та його подання у форматі, що відповідає варіанту.
4. Опис вхідних і вихідних даних
5. Опис формату вхідного файлу
6. Проектування необхідних класів і структур.
7. Функціональні тести
8. Лістинг програми та екранні форми результату роботи програми.
9. Висновки

**Варіант 9** індивідуального завдання до лабораторної роботи 5:

Тип графу – орієнтований;

Зовнішній формат подання графу – EL (Edge List);

Внутрішнє представлення списку – однозв'язний;

Внутрішнє представлення підсписку – однозв'язний циклічний

хід роботи

Опис програми:

Мова програмування: С#, операційна система Windows 11 Prо, Версія 23H2, Збірка ОС 22621.1325, процесор: Apple Silicon M1 Pro 3.20 GHz (ядер: 6), компілятор: Microsoft Visual Studio Community 2022 (64-розрядна версія ARM).

Теоретичні відомості

Опис формату подання список звʼязків (Edge List (EL))

Для подання ґрафа у вигляді списку зв'язків необхідно використовувати кілька різних роздільників, які дозволять зобразити кожний зв'язок у вигляді логічно зв'язаної пари чисел.

Основні елементи формату Edge List включають:

* **Ідентифікатори вершин.** Кожна вершина має свій унікальний ідентифікатор або мітку. Ідентифікатори можуть бути числами, символами або комбінацією обох.
* **Роздільники.** Для визначення пар зв’язків та розділення ідентифікаторів вершин використовуються роздільники. Найпоширенішим роздільником є пробіл, але його можна замінити комою, табуляцією чи іншим символом.
* **Напрямок зв'язку (опціонально).** У деяких випадках може бути важливим вказати напрямок зв'язку між вершинами. Це можна реалізувати додаванням спеціального символу чи індикатора, що вказує на напрямок.

Такий формат дозволяє лаконічно визначити зв'язки між вершинами графа, і його легко читати та редагувати.

Граф

Створимо граф, який буде задовольняти умовам поставленого завдання: не менше 11 вершин, але не більше 20, та кількість зв’язків не більше 50. Створений граф подано на рисунку 1.

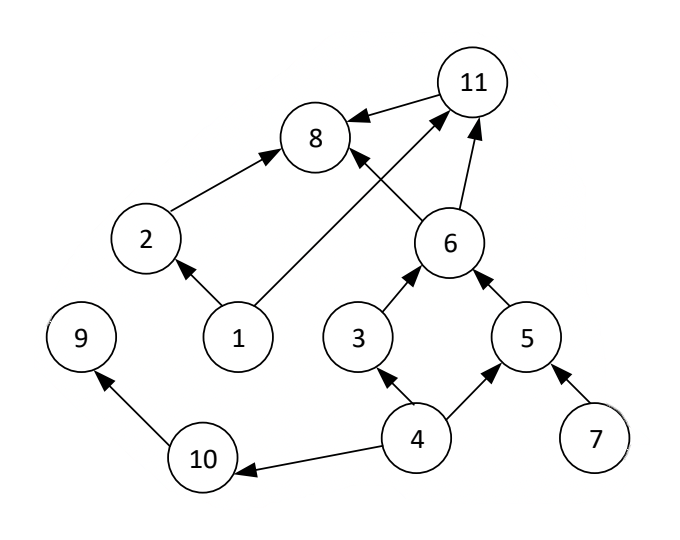


Рисунок 1 – Графічне подання графа

Подання графа у форматі Edge List (EL)

Відповідно до рисунку 1, подання графу у форматі заданого для варіанту 9 записано нижче:

*1 2; 1 11; 2 8; 3 6; 4 3; 4 5; 4 10; 5 6; 6 8; 6 11; 7 5; 10 9; 11 8;*

Опис вхідних і вихідних даних програми:

**Вхідні дані:**

* ushort input – змінна використовується для перевірки введеного числового значення на відповідність заданої в метод області допустимих значень;
* string input\_keyboard – приймає введений граф з клавіатури;
* var key – змінна для зчитування натиснутої клавіші. Використовується для відокремлення операцій в консолі за допомогою перевірки натискання клавіші Enter;
* inputFromFile – приймає введений граф з файлу з розширенням \*.txt;

**Вихідні дані:**

* Graph graph – структура даних яка зберігає дуги графа в однозв'язному списку та вершини в однозв'язному циклічному списку.
* Edge\_LinkedList current – запис поточної дуги переданого в метод графу для виведення на екран;

Опис формату вхідного файлу

Формату вхідного файлу для зчитування текстового файлу в змінну string та наступний запис графу включає наступні характеристики:

* Тип файлу: Текстовий файл (\*.txt);
* Кодування: UTF-8;
* Структура файлу: Список зв'язків (Edge List);
* Роздільник вершин: Пробіл
* Роздільник дуг: Крапка з комою(+пробіл);
* Приклад вмісту файлу:

1. 1 2; 1 11; 2 8; 3 6; 4 3;
2. 4 5;4 10;5 6;6 8;6 11;7 5

Проектування необхідних класів і структур.

Vershyna - Представляє вершину графа.

* int inf: Інформаційне поле, яке містить додаткову інформацію про вершину.

public class Vershyna

{

public int inf;

public Vershyna(int value)

}

Edge - Представляє ребро графа.

* Vershyna Start: Початкова вершина ребра.
* Vershyna End: Закінчальна вершина ребра.

public class Edge

{

public Vershyna Start;

public Vershyna End;

public Edge(Vershyna start, Vershyna end)

}

Pid\_sp\_Vershyny - Представляє вершину графа в однозв'язному підсписку.

* Vershyna inf: Інформаційне поле, яке містить вершину.
* Pid\_sp\_Vershyny Next: Вказівник на наступний елемент у підсписку.

public class Pid\_sp\_Vershyny

{

public Vershyna inf;

public Pid\_sp\_Vershyny Next;

public Pid\_sp\_Vershyny(Vershyna value)

}

Edge\_LinkedList - Представляє ребро графа в однозв'язному списку ребер.

* Edge Data: Ребро, яке міститься в вузлі списку.
* Edge\_LinkedList Next: Вказівник на наступний елемент у списку.

public class Edge\_LinkedList

{

public Edge Data { get; set; }

public Edge\_LinkedList Next { get; set; }

public Edge\_LinkedList(Edge data)

}

Graph - Представляє граф.

* Edge\_LinkedList Edges: Список ребер графа у вигляді однозв'язного списку.
* Pid\_sp\_Vershyny Vertices: Циклічний підсписок вершин графа.
* const int MaxEdges: Константа, що визначає максимально допустиму кількість ребер.
* int currentEdgeCount: Лічильник кількості ребер у графі.
* public void AddEdge(Edge edge): Додає ребро до графа, включаючи оновлення списку ребер та підсписку вершин.

public class Graph

{

public Edge\_LinkedList Edges { get; private set; }

public Pid\_sp\_Vershyny Vertices { get; private set; }

private const int MaxEdges = 50;

private int currentEdgeCount;

public void AddEdge(Edge edge)

{

if (currentEdgeCount >= MaxEdges)

throw new Exception($"Кількість зв'язків в графі перевищило максимально допустиму кількість - ({MaxEdges}).");

Edge\_LinkedList newEdgeNode = new Edge\_LinkedList(edge);

newEdgeNode.Next = Edges;

Edges = newEdgeNode;

Pid\_sp\_Vershyny newVertexNode = new Pid\_sp\_Vershyny(edge.Start);

newVertexNode.Next = Vertices;

Vertices = newVertexNode;

currentEdgeCount++;

}

}

Функціональні тести

Введення графу, в якого кількість вершин менша 11. Результат подано на рисунку 2.

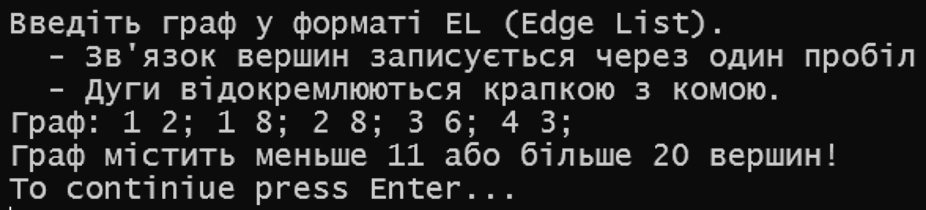


Рисунок 2 – помилка кількості вершин в графі

Введення графа з буквенними символами не перешкоджають роботі програми. Результат на рисунку 3.

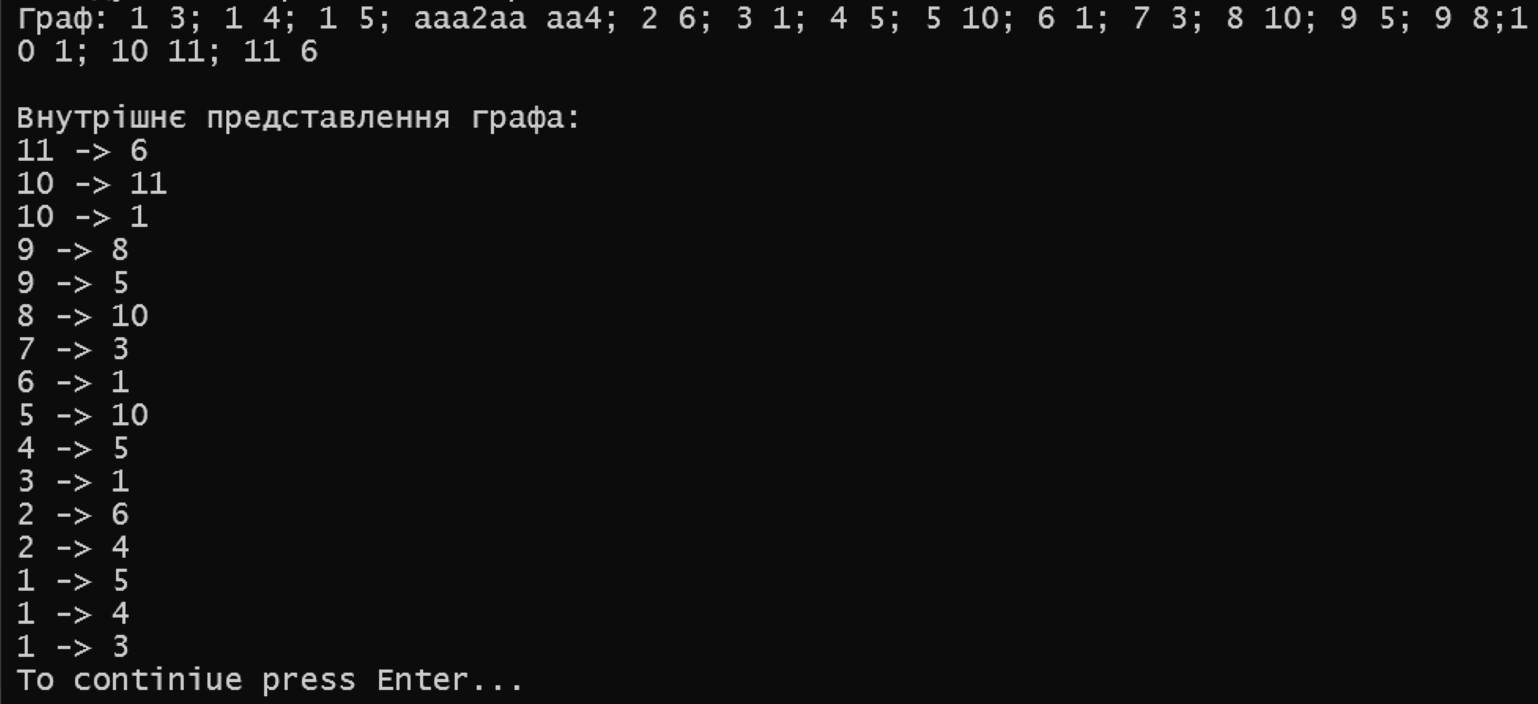


Рисунок 3 – введення букв у граф

При використанні не тих роздільників програма виводить повідомлення в яких елементах була помилка та при перевірці відповідності кількості вершин вивела помилку на перервала виконання. Результат на рисунку 4.

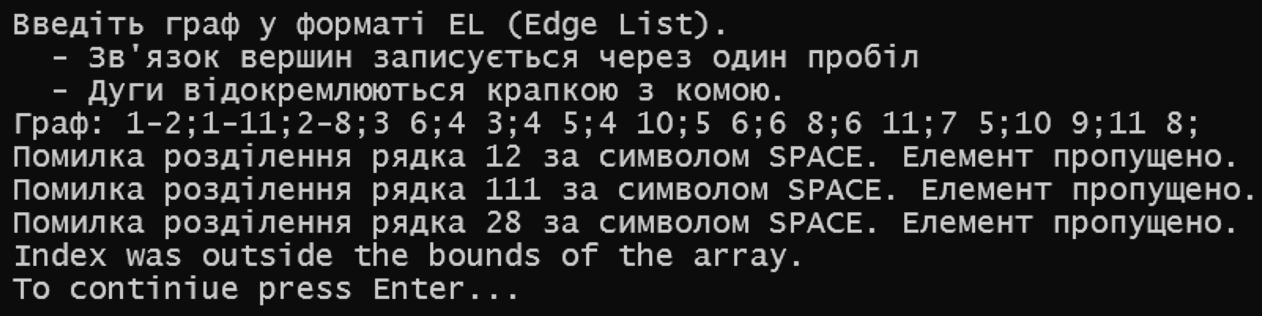


Рисунок 4 – використання неправильного роздільника в графі

Кількість вершин графа більше 20. Результат на рисунку 5

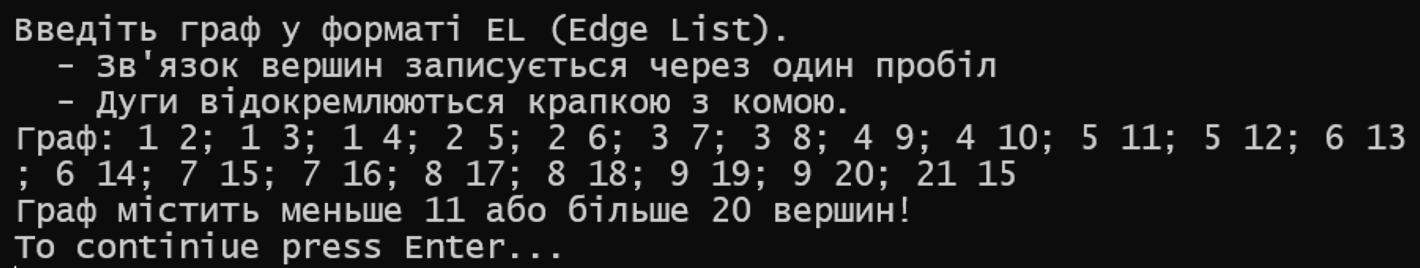


Рисунок 5 – граф з 21 вершиною

За умовою завдання в графі не може бути більше 50 зв’язків. Результат роботи програми на рисунку 6.

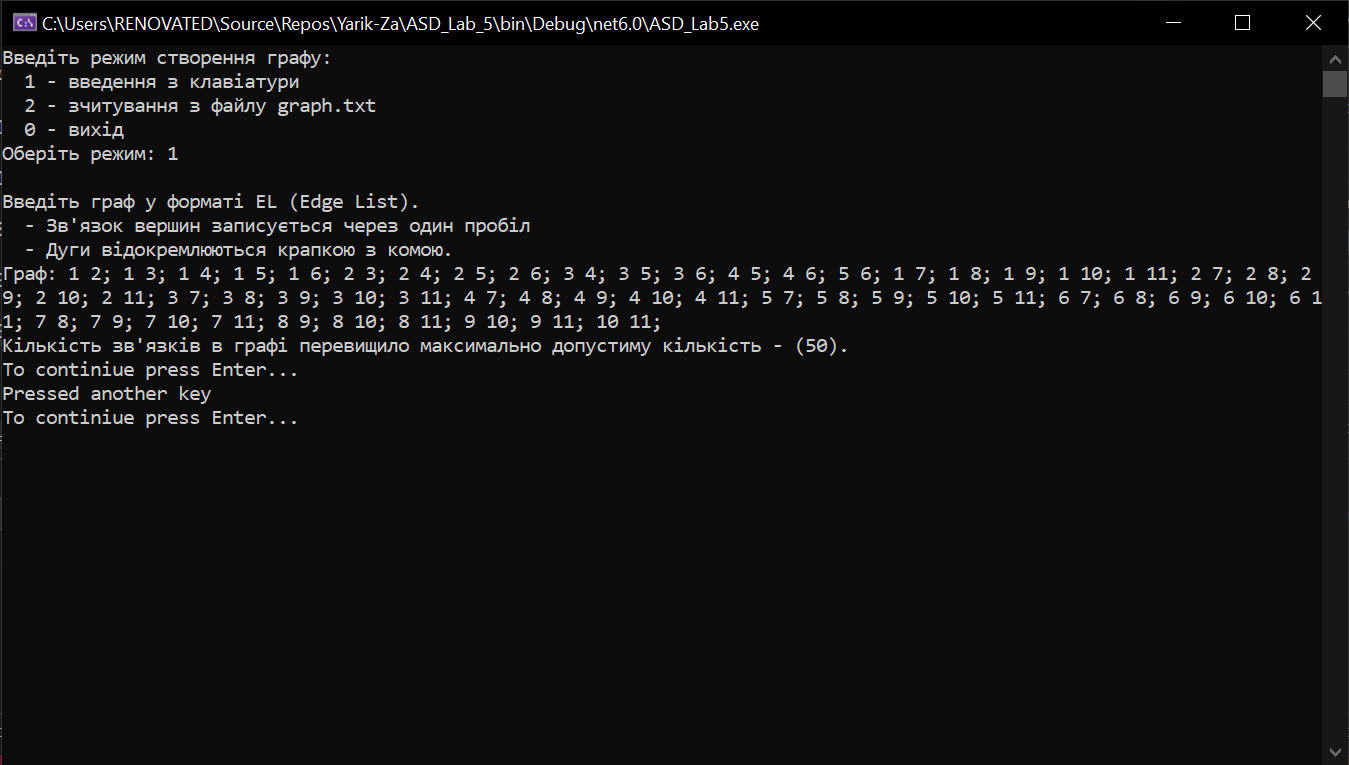


Рисунок 6 – граф з 50 зв’язками

Результати аналогічних тестових випадків але при використанні текстового файлу з внутрішнім поданням графу подано на рисунках 7-12

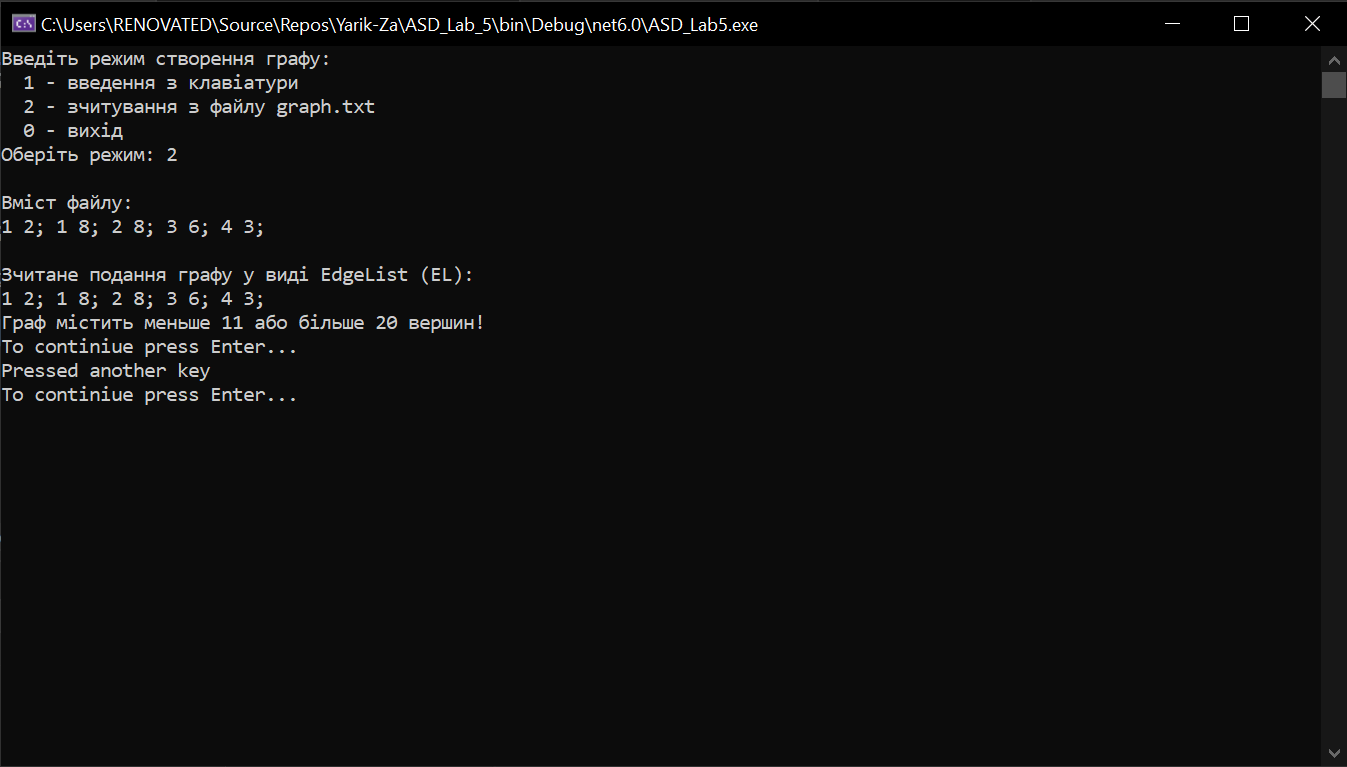


Рисунок 7 – граф з малою кількістю вершин у файлі

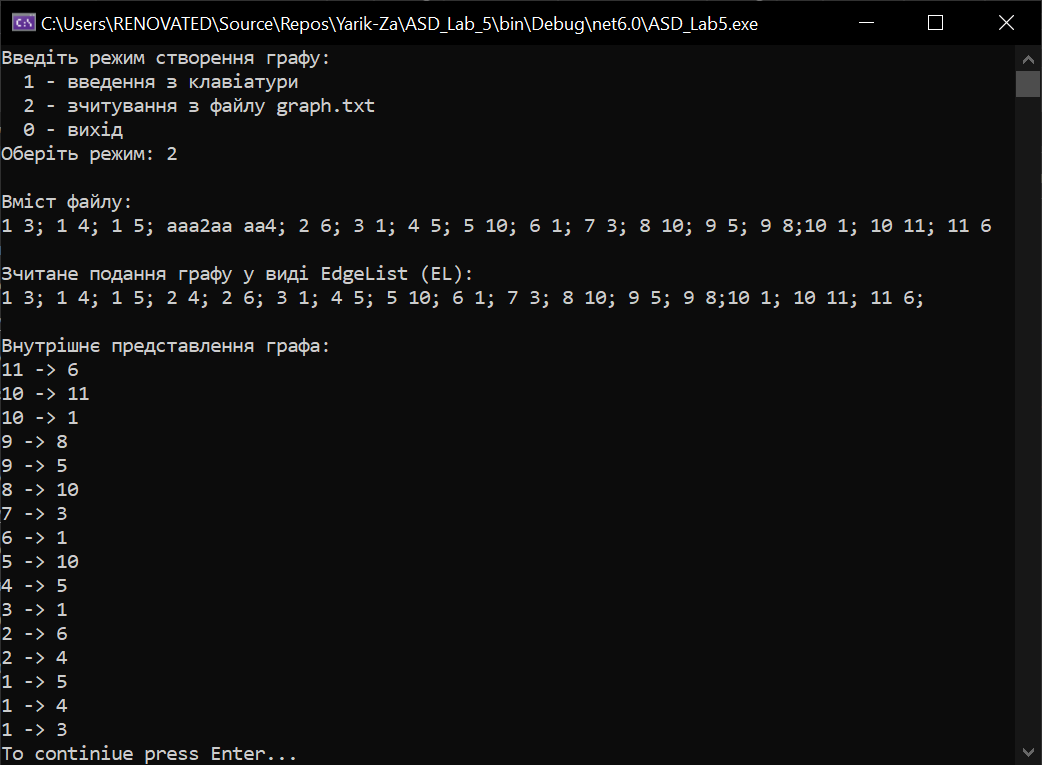


Рисунок 8 – граф з буквенними символами у файлі

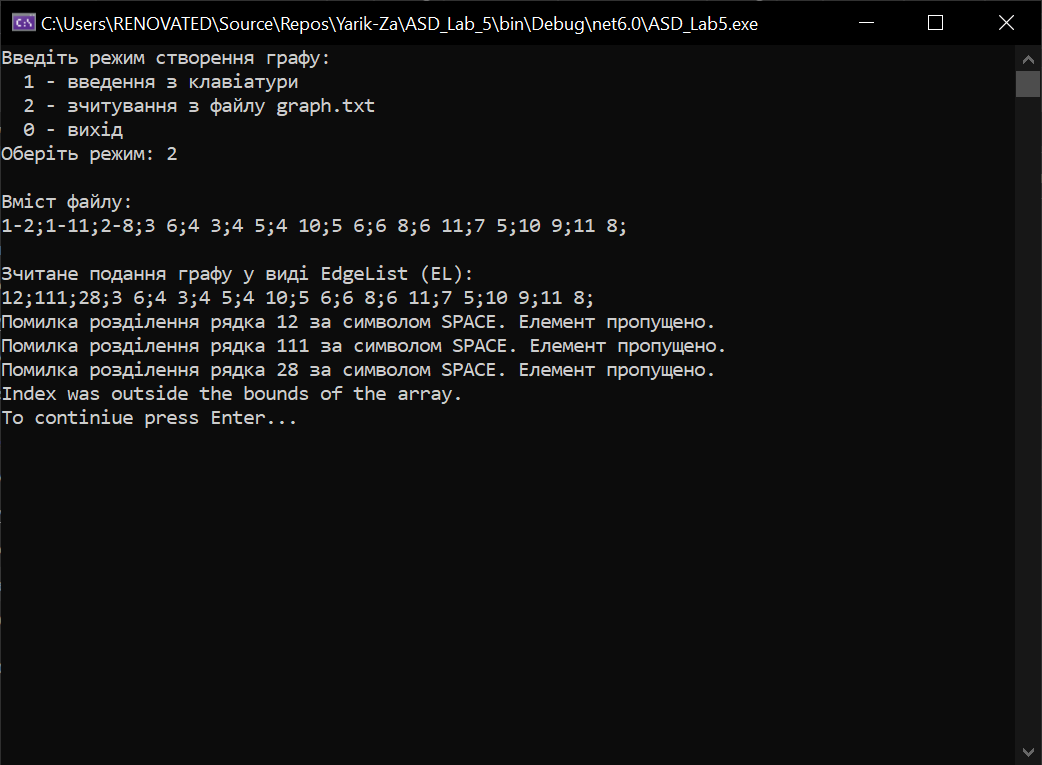


Рисунок 9 – граф з неправильними роздільниками у файлі

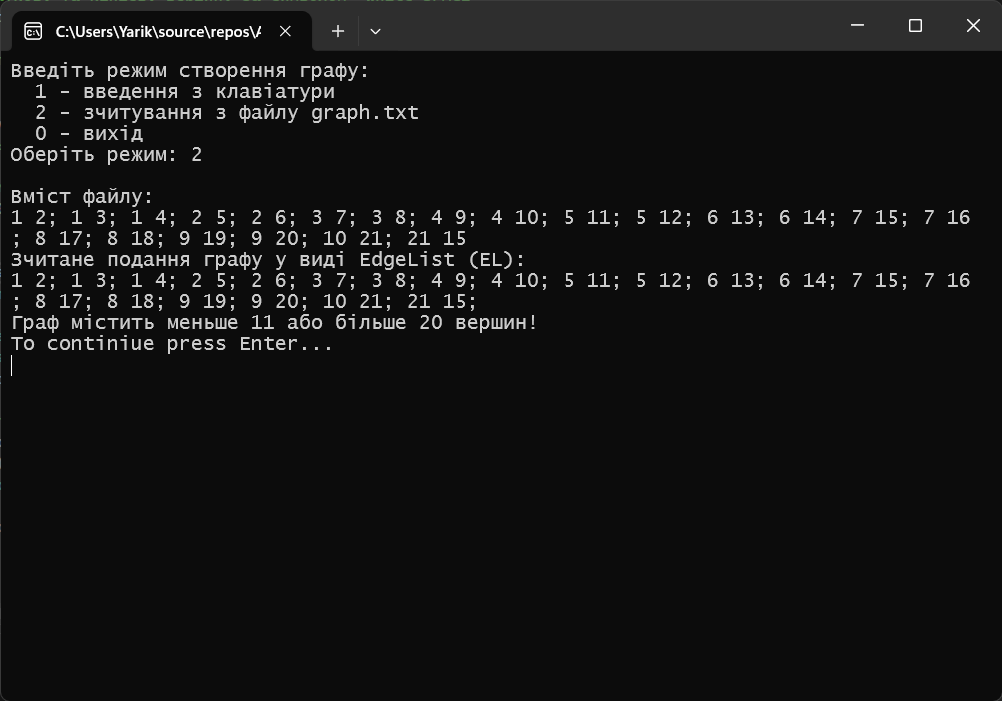


Рисунок 10 – граф з 21 вершиною в файлі

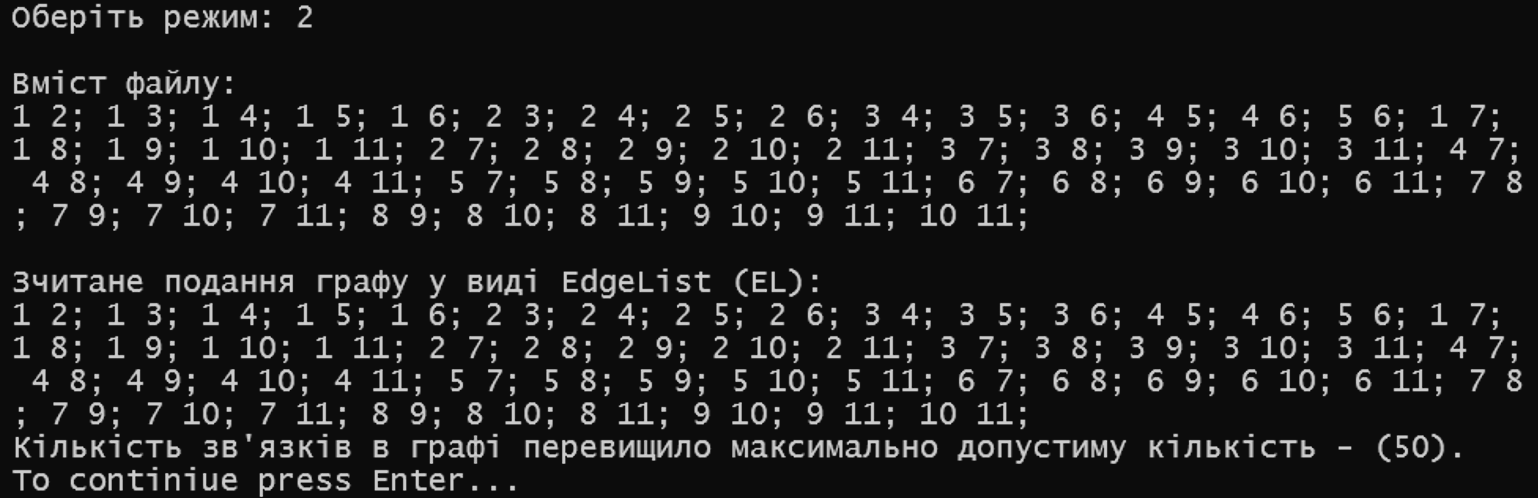


Рисунок 11 – граф з 50 зв’язками в файлі

ВИСНОВОК

У ході виконання лабораторної роботи було розроблено програму для роботи з графами, яка дозволяє вводити граф з клавіатури або зчитувати з текстового файлу, перевіряти кількість вершин та зв'язків згідно з вказаними обмеженнями (вершини ≤ 20, зв'язків ≤ 50) та перетворювати внутрішнє представлення графа.

Програма реалізована на мові програмування C# та використовує об'єктно-орієнтований підхід, використовуючи класи та методи для представлення вершин, зв'язків та самого графа. Для перевірки та обмеження кількості вершин та зв'язків використовуються відповідні умови.

Програма успішно пройшла тестування на різних комбінаціях коректних та некоректних даних. Вона ефективно обробляє введені графи, перевіряє їхню коректність та правильність перетворення внутрішнього представлення.

Завершений додаток має інтерактивний інтерфейс, який дозволяє вводити граф з клавіатури або зчитувати його з файлу, а також виводити внутрішнє представлення графа на екран. Програма забезпечує зручний та надійний інтерфейс для роботи з графами.

Розроблена програма відповідає поставленим завданням, є функціональною та надійною, і готова до подальшого використання у різних сценаріях введення та обробки графів.

*Додаток А*

Машинний лістинг програми

Program.cs:

using System.Text;

namespace ASD\_Lab5

{

internal class Program

{

static void Main()

{

// Устанавливаем кодировку консоли на UTF-8

Console.InputEncoding = Encoding.UTF8;

Console.OutputEncoding = Encoding.UTF8;

const string menu = ($"Введіть режим створення графу:\n" +

$" 1 - введення з клавіатури\n" +

$" 2 - зчитування з файлу graph.txt\n" +

$" 0 - вихід\n");

while (true)

{

try

{

switch (Input\_range(menu + "Оберіть режим: ", 2, 0))

{

case 0: Environment.Exit(0); break;

case 1:

Graph.currentEdgeCount = 0;

Graph keyboard\_graph = new Graph();

Console.Write($"Введіть граф у форматі EL (Edge List).\n - Зв'язок вершин записується через один пробіл\n - Дуги відокремлюються крапкою з комою.\nГраф: ");

string input\_keyboard = Console.ReadLine();

string[] graph\_edges = ParseGraph(input\_keyboard);

BuildGraph(keyboard\_graph, graph\_edges);

// Виведення внутрішнього представлення графа

PrintGraph(keyboard\_graph);

break;

case 2:

Graph.currentEdgeCount = 0;

Graph file\_graph = new Graph();

// зчитати граф з файлу

string inputFromFile = File.ReadAllText("graph.txt");

Console.WriteLine("Вміст файлу:\n" + inputFromFile);

string[] graph\_edges\_file = ParseGraph(inputFromFile);

Console.WriteLine("Зчитане подання графу у виді EdgeList (EL):\n" + ToString(graph\_edges\_file));

BuildGraph(file\_graph, graph\_edges\_file);

// Виведення внутрішнього представлення графа

PrintGraph(file\_graph);

break;

}

PressEnter();

Console.Clear();

}

catch (Exception ex) { Console.WriteLine(ex.Message); PressEnter(); }

}

}

public static string ToString(string[] vector)

{

string res = null;

foreach (var el in vector)

res += el + ";";

return res;

}

static string[] ParseGraph(string input\_EL)

{

// Перевірка на порожній ввід

if (string.IsNullOrWhiteSpace(input\_EL))

throw new Exception("Неправильний ввід.");

string parced\_input = null;

foreach (var symb in input\_EL)

if (char.IsDigit(symb) || symb == ';' || symb == ' ')

parced\_input += symb;

// Розділення вводу на ребра за символом ';'

string[] edges = parced\_input.Split(';', StringSplitOptions.RemoveEmptyEntries);

return edges;

}

static void BuildGraph(Graph graph, string[] edges)

{

foreach (string edgeStr in edges)

{

// Перевірка на порожній рядок

if (string.IsNullOrWhiteSpace(edgeStr))

throw new Exception("Пустий рядок");

string edgeStr\_Trim = edgeStr.Trim();

// Розділення ребра на початкову та кінцеву вершину за символом 'White\_SPACE'

string[] vertices = edgeStr\_Trim.Split(' ');

// Перевірка на правильність розділення ребра

if (vertices.Length != 2)

{

Console.WriteLine($"Помилка розділення рядка {edgeStr} за символом SPACE. Елемент пропущено.");

continue; // Пропустить некорректное ребро

}

// Перетворення рядків у числа

if (int.TryParse(vertices[0], out int startValue) && int.TryParse(vertices[1], out int endValue))

{

if (!CountOfVersh(edges))

throw new Exception("Граф містить меньше 11 або більше 20 вершин!");

Vershyna start = new Vershyna(startValue);

Vershyna end = new Vershyna(endValue);

Edge edge = new Edge(start, end);

// Додавання ребра до графа

if (Graph.currentEdgeCount <= 50)

graph.AddEdge(edge);

else throw new Exception("Граф має більше 50 зв'язків!");

}

else Console.WriteLine($"Помилка перетворення рядка {vertices[0]} або {vertices[1]} в число. Рядок пропущено");

}

}

//static bool CountOfVersh(string[] edges)

//{

// bool res = false;

// foreach (string el in edges)

// {

// string trimmedEl = el.Trim();

// string[] temp = trimmedEl.Split(' ');

// if (Convert.ToInt32(temp[1]) >= 11 && Convert.ToInt32(temp[0]) <= 20)

// res = true;

// }

// return res;

//}

static bool CountOfVersh(string[] edges)

{

bool res = false;

foreach (string el in edges)

{

string trimmedEl = el.Trim();

string[] temp = trimmedEl.Split(' ', StringSplitOptions.RemoveEmptyEntries);

// Перевірка, чи вершини знаходяться в межах від 1 до 20

if (Convert.ToInt32(temp[0]) < 1 || Convert.ToInt32(temp[0]) > 20 ||

Convert.ToInt32(temp[1]) < 1 || Convert.ToInt32(temp[1]) > 20)

{

res = false;

}

// Перевірка, чи кількість вершин у ребрах знаходиться в інтервалі від 11 до 20

if (Convert.ToInt32(temp[0]) >= 11 || Convert.ToInt32(temp[1]) >= 11 ||

Convert.ToInt32(temp[0]) <= 20 || Convert.ToInt32(temp[1]) <= 20)

{

res = true;

}

}

return res;

}

static void PrintGraph(Graph graph)

{

Console.WriteLine("\nВнутрішнє представлення графа:");

Edge\_LinkedList current = graph.Edges;

while (current != null)

{

Console.WriteLine($"{current.Data.Start.inf} -> {current.Data.End.inf}");

current = current.Next;

}

}

public static void PressEnter()

{

while (true)

{

Console.WriteLine("To continiue press Enter...");

var key = Console.ReadKey(intercept: true);

if (key.Key == ConsoleKey.Enter)

{

Console.Clear();

break; // Выход из цикла, если нажата клавиша Enter

}

else Console.WriteLine("Pressed another key");

}

}

public static ushort Input\_range(string text, ushort up\_range, ushort down\_range)

{

while (true)

{

ushort input;

try

{

Console.Write(text);

input = Byte.Parse(Console.ReadLine());

if (input > up\_range || input < down\_range)

throw new Exception($"Value should be in range {down\_range}-{up\_range}.");

}

catch (Exception ex)

{

Console.WriteLine($"{ex.Message}");

continue;

}

Console.WriteLine();

return input;

}

}

}

}

*Додаток Б*

Машинний лістинг створеного класу

**Graph.cs**

namespace ASD\_Lab5

{

public class Vershyna

{

public int inf;

public Vershyna(int value)

{

inf = value;

}

}

// Клас, що представляє ребро графа

public class Edge

{

public Vershyna Start;

public Vershyna End;

public Edge(Vershyna start, Vershyna end)

{

Start = start;

End = end;

}

}

// Клас, що представляє однозв'язний підсписок вершин графа

public class Pid\_sp\_Vershyny

{

public Vershyna inf;

public Pid\_sp\_Vershyny Next;

public Pid\_sp\_Vershyny(Vershyna value)

{

inf = value;

Next = null;

}

}

// Клас, що представляє однозв'язний список ребер графа

public class Edge\_LinkedList

{

public Edge Data { get; set; }

public Edge\_LinkedList Next { get; set; }

public Edge\_LinkedList(Edge data)

{

Data = data;

Next = null;

}

}

// Клас, що представляє граф

public class Graph

{

public Edge\_LinkedList Edges { get; private set; }

public Pid\_sp\_Vershyny Vertices { get; private set; }

// Максимальна допустима кількість зв'язків

private const int MaxEdges = 50;

// лічильник кількість зв'язків

public static int currentEdgeCount;

// Додавання ребра до графа

public void AddEdge(Edge edge)

{

// Проверка на превышение максимального количества связей

if (currentEdgeCount >= MaxEdges)

throw new Exception($"Кількість зв'язків в графі перевищило максимально допустиму кількість - ({MaxEdges}).");

Edge\_LinkedList newEdgeNode = new Edge\_LinkedList(edge);

newEdgeNode.Next = Edges;

Edges = newEdgeNode;

// Додавання вершини до циклічного підсписку

Pid\_sp\_Vershyny newVertexNode = new Pid\_sp\_Vershyny(edge.Start);

newVertexNode.Next = Vertices;

Vertices = newVertexNode;

currentEdgeCount++;

}

}

}