ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД УКООПСПІЛКИ
«ПОЛТАВСЬКИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТОРГІВЛІ»

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ БІЗНЕСУ
ТА СУЧАСНИХ ТЕХНОЛОГІЙ

ФОРМА НАВЧАННЯ ЗАОЧНА

КАФЕДРА МАТЕМАТИЧНОГО МОДЕЛЮВАННЯТА СОЦІАЛЬНОЇ ІНФОРМАТИКИ

Допускається до захисту

Завідувач кафедри _____О.О. Ємець
(підпис)

«_____»_____ 2021 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА ДО БАКАЛАВРСЬКОЇ РОБОТИ

на тему

ВІЗУАЛІЗАТОР НЕОРІЄНТОВАНИХ ГРАФІВ (НА БАЗІ МАТРИЦЬ СУМІЖНОСТІ ТА ІНЦИДЕНЦІЙ) ТА РОЗРОБКА ЙОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

зі спеціальності 122 «Комп'ютерні науки та інформаційні технології»

Виконавець роботи Горбенко Владислав Сергійович _____ «___»_____2021 р.
(підпис)
Науковий керівник к.ф.-м.н., проф., Ємець Єлизавета Михайлівна
_____ «___»_____ 2021 р.
(підпис)

ПОЛТАВА 2021 р.

РЕФЕРАТ

Записка: 44 с., 43 рис., 3 табл., 1 додаток (на 8 сторінках), 10 джерел.

Предмет розробки – візуалізатор, що демонструє представлення неорієнтованих графів без петель за допомогою матриць суміжності та інцидентності.

Мета роботи – створити програму-візуалізатор, що за матрицею суміжності або інцидентності будує у графічному виді граф.

Методи розробки – мова об'єктно-орієнтованого програмування Object Pascal в середовищі візуального програмування Delphi.

Створено два алгоритми тренажеру та блок-схему.

Розроблено алгоритми візуалізатора, де за матрицями суміжності та інцидентності рисується неорієнтований граф. Створено блок-схеми алгоритмів.

Створено програму-візуалізатор, в якій за матрицями суміжності або інцидентності малюється граф.

Ключові слова: НЕОРІЄНТОВАНИЙ ГРАФ, МАТРИЦЯ СУМІЖНОСТІ, МАТРИЦЯ ІНЦИДЕНЦІЙ, ВІЗУАЛІЗАТОР.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Умовні позначення, символи, одиниці, скорочення, терміни	**Пояснення умовних позначень, символів, одиниць, скорочень, термінів**
G	Граф.
V	Множина вершин графа.
v_i	Вершина графа.
E	Множина ребер графа.
(v_i, v_j), $e = (v_i, v_i)$, e, e_k	Ребро графа.
n	Кількість вершин графа.
m	Кількість ребер графа.
$A = \{a_{ij}\}_{i,j=1}^{n}$	Матриця суміжності.
a_{ij}	Елемент матриці суміжності, що стоїть на перетині рядку з номером i та стовпця з номером j.
$B = \{b_{ij}\}_{i=1,j=1}^{n,m}$	Матриця інцидентності (інциденцій).
b_{ij}	Елемент матриці інцидентності, що стоїть на перетині рядку з номером i та стовпця з номером j.
$\varnothing$	Пуста множина
$\lvert V \rvert$	Потужність множини V. Кількість елементів в множині V.

ВСТУП

Теорія графів отримала широко розповсюдження через те, що в житті зустрічається багато задач, що зводять до неї. Для розв'язування задач використовують комп'ютерну техніку. У програміста виникає потреба збереження графа у пам'яті комп'ютері. Для представлення графів використовуються різні способи, зокрема, матриці суміжності та інцидентності. Такі матриці у програмах можна задати, наприклад, двовимірними масивами (якщо часто виконується перевірка наявності ребер у графі) або списків ненульових елементів (для розріджених матриць), або двійковими векторами (коли у графі мало ребер).

Для розуміння, як задавати графи матриця суміжності та інцидентності, студентами допомагають програми-візуалізатори. Їх основа мета – це наочність, динаміка, візуалізація.

Таким чином, тема випускової роботи є *актуальною*.

Об'єкт розробки – програма-візуалізатор графів.

Предмет розробки – візуалізатор, що демонструє представлення неорієнтованих графів без петель за допомогою матриць суміжності та інцидентності.

Мета та задачі дипломного проектування – створити програму-візуалізатор, що за матрицею суміжності або інцидентності будує у графічному виді граф. Обмежитись неорієнтованими графами без петель.

Методи розробки – це мова об'єктно-орієнтованого програмування Object Pascal в середовищі візуального програмування Delphi.

Структура пояснювальної записки. Документація складається з 4 розділів. У постановці задачі (1-ий розділ) викладено вимоги до дипломного проектування та програми. У інформаційному огляді (2-ий розділ) висвітлено роботу інших вже існуючих і представлених в глобальній мережі візуалізаторів. В теоретичній частині (3-ий розділ) подані означення, що потрібні для побудови та розуміння

візуалізатора; викладено алгоритм програми та блок-схему алгоритму; В практичній частині (4-ий розділ) описано створену програму та подано інструкції з її використання. *Загальний обсяг* пояснювальної записки – 44 сторінки. Обсяг додатків – 8 сторінок.

1. ПОСТАНОВКА ЗАДАЧІ

При виконанні випускової роботи слід ознайомитись з темою «Графи». Вивчити необхідну термінологію. Зосередити увагу на представленні графів матрицями суміжності та матриця інцидентності.

Здійснити огляд візуалізаторів або інших програмних продуктів, що представляють графи за допомогою матриць суміжності та інцидентності. Проаналізувати існуючи розробки.

Розробити алгоритм візуалізатора, який:

1) за заданою матрицею суміжності будує неорієнтований граф без петель;

2) за заданою матрицею інцидентності будує неорієнтований граф без петель.

Створити блок-схему алгоритму.

Написати програму-візуалізатор, яка за введеною (заданою) матрицею суміжності або інцидентності малює граф. Для спрощення на вхід подавати лише неорієнтований граф до 10 вершин, без петель.

У програмі передбачити збереження намальованого графу у вигляді графічного файлу. Для матриці інцидентності передбачити підпис ребер.

Протестувати програму.

Описати створену програми. Створити інструкцію по роботі з програмою.

Надати пояснення по коду створеного програмного продукту.

2. ІНФОРМАЦІЙНИЙ ОГЛЯД

2.1. Огляд розробок, аналогічних темі випускової роботи

1) Ресурс **«Редактор графів»** https://www.semestr.online/graph/graph.php (рис. 2.1-2.10).

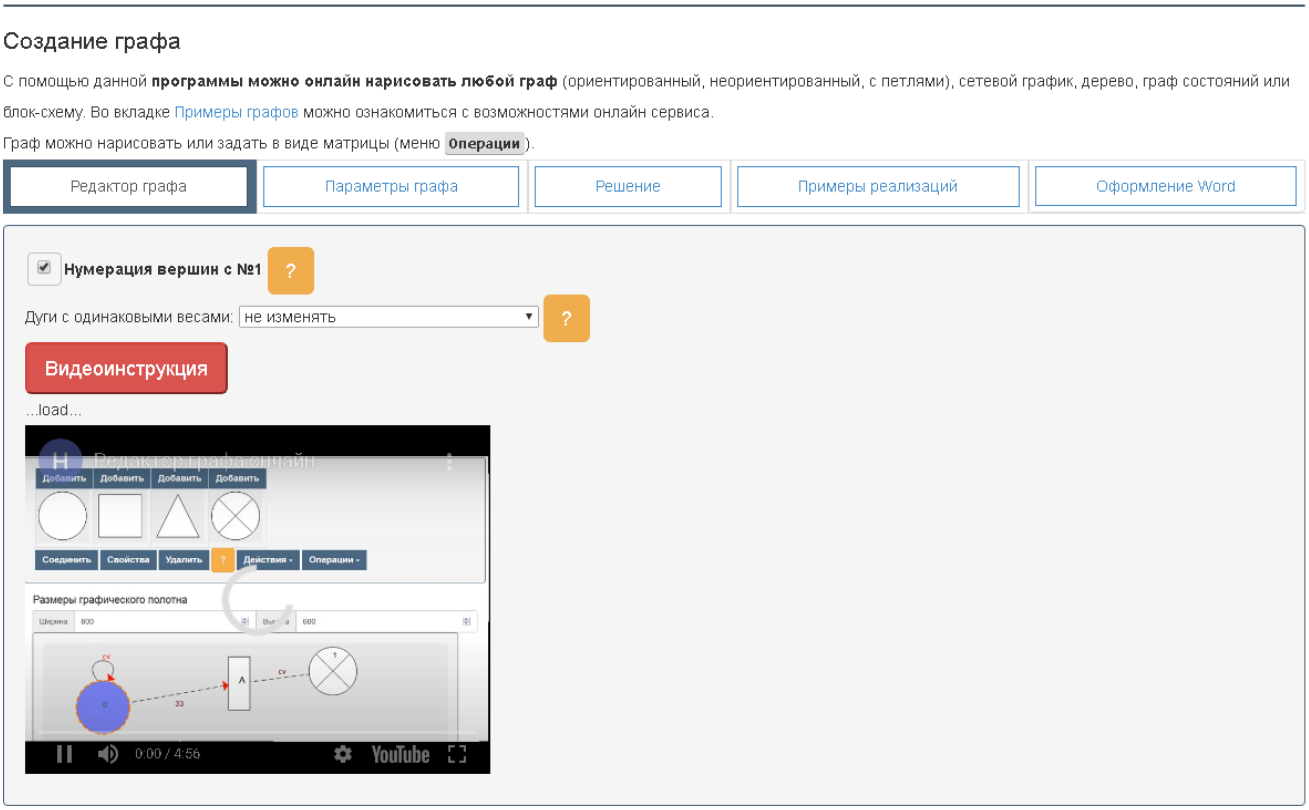


Рисунок 2.1 – Ресурс «Редактор графів»

Веб-додаток дозволяє намалювати граф, а після цього отримати матрицю суміжності, інцидентності, матрицю відстаней, найкоротший шлях між двома заданим вершинами.

Є можливість задати форму для вузлів графа: коло, прямокутник, трикутник, коло, що поділене на 4 сектори (рис. 2.2). Можна керувати розміром геометричних фігур, які використовуються для позначення вузлів графа. Можна задавати текст, що буде розташований у середині вузла. Можна задати колір для вузла (обведення та заповнення), для тексту, для ребра. Є можливість вибору

форми лінії або дуги, наприклад, пунктир або суцільна лінія. Те ж саме можна зробити з лінією, що є обведенням вузла.

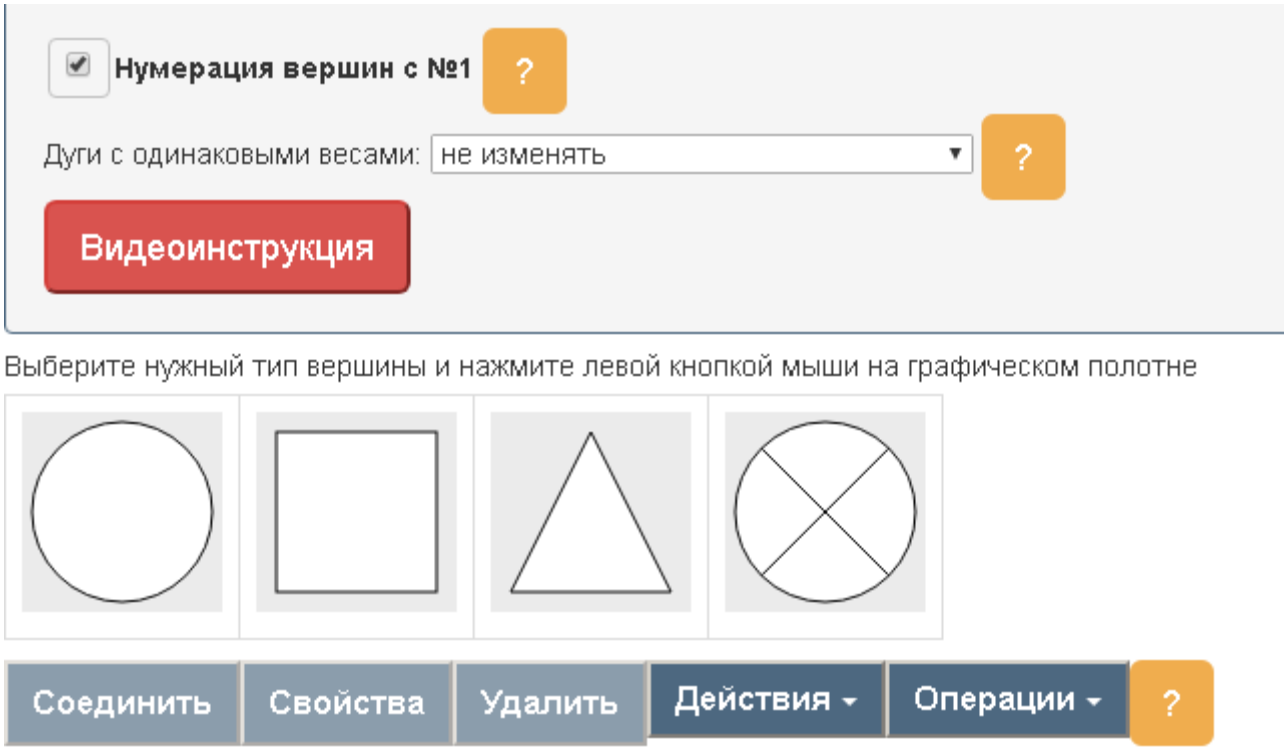


Рисунок 2.2 – Меню ресурсу

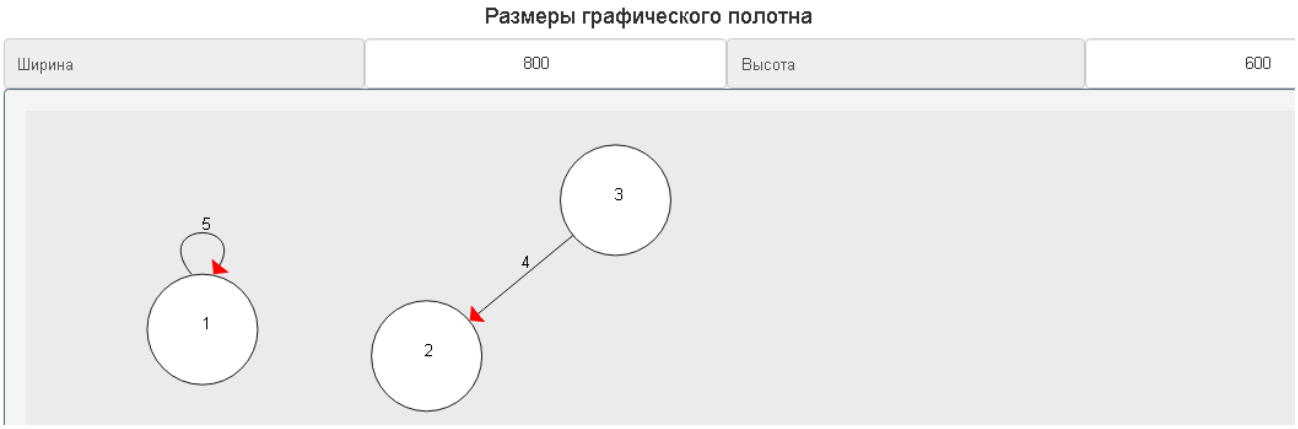


Рисунок 2.3 – Полотно для малювання графа з побудованим прикладом

Є можливість редагування розмірів полотна (рис. 2.3), на якому рисується граф. Побудований граф можна зберегти у форматі картинки та текстового документа (рис. 2.4).

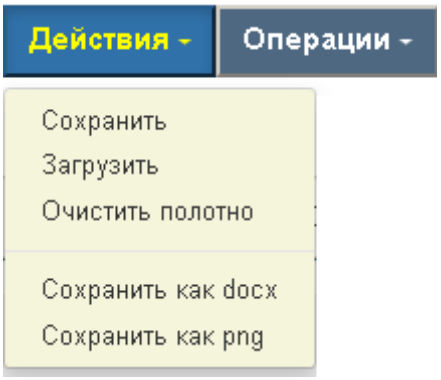


Рисунок 2.4 – Пункт меню «Дії»

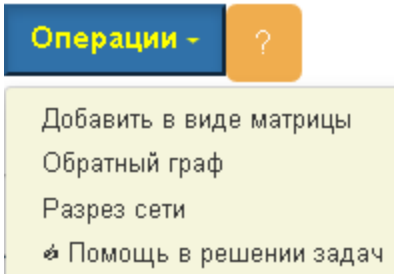


Рисунок 2.5 – Пункт меню «Операції»

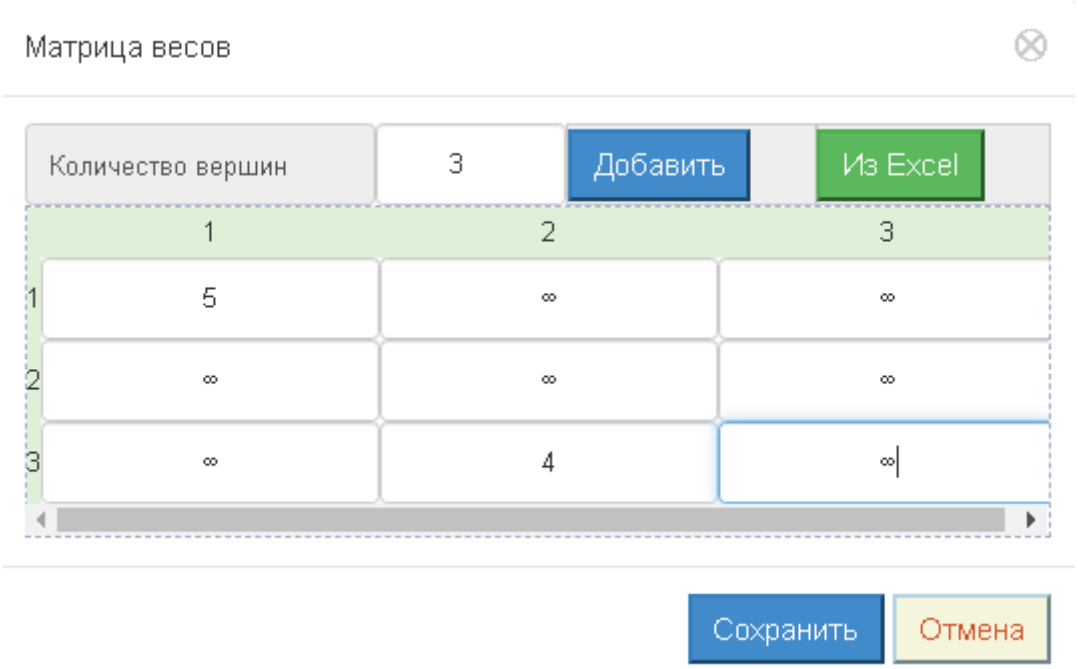


Рисунок 2.6 – Задання графу через матрицю ваг

Граф може бути орієнтований (рис 2.3, 2.7) або неорієнтований (рис. 2.8), або змішаний, з петлями (рис 2.3, 2.7) або без них (рис. 2.8).

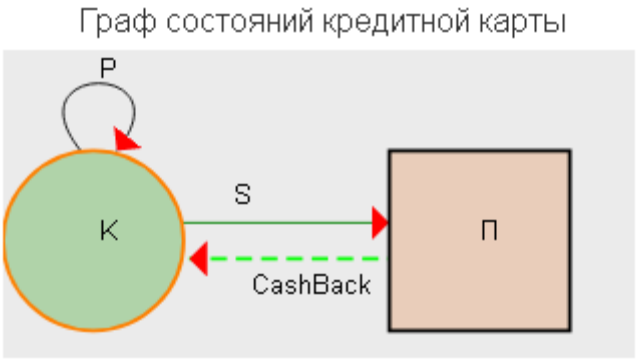


Рисунок 2.7 – Приклад графа (з довідки ресурсу)

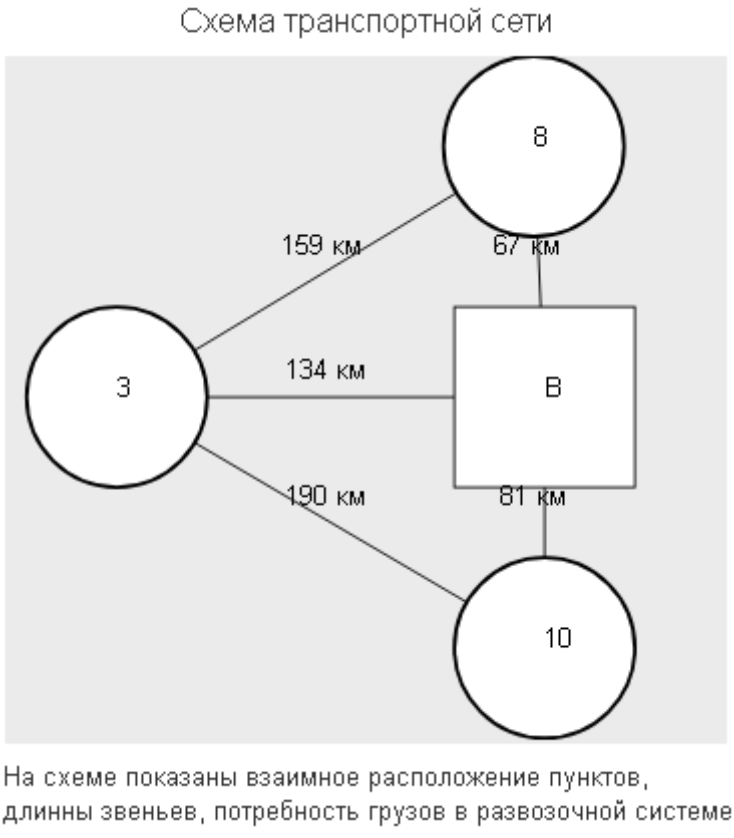


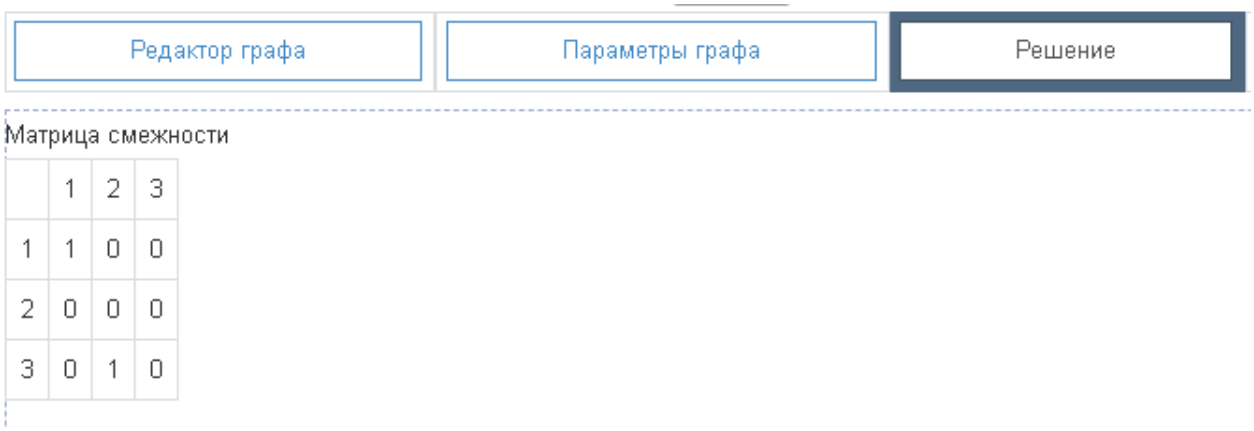
Рисунок 2.8 – Приклад графа (з довідки ресурсу)

Є можливість намалювати граф за заданою матрицею ваг (рис. 2.5-2.6), яку, наприклад. можна експортувати з файлу Excel.

Ресурс містить відео та текстову інструкцію (рис. 2.1). Показані приклади різних графів. Вони ж є ілюстраціями практичних задач (рис. 2.7-2.8).

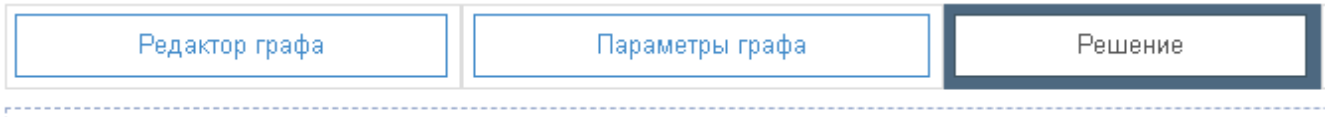
Є необхідні теоретичні відомості з теорії графів.

На рис. 2.9 показана матриця суміжності для графа з рис. 2.3. На рис. 2.10 представлена матриця інцидентності для графа з рис. 2.3.



	1	2	3
1	1	0	0
2	0	0	0
3	0	1	0

Рисунок 2.9 – Матриця суміжності для графа з рис. 2.3



	g(1,1)	g(3,2)
1	-1	0
2	0	-1
3	0	1

Рисунок 2.10 – Матриця інцидентності для графа з рис. 2.3

2) Ресурс http://grafoanalizator.unick-soft.ru/program/ дозволяє завантажити програму **«Графоаналізатор»** (рис. 2.11-2.18).

Тут є можливість працювати з орієнтований або неорієнтованим графом, з вагами ребер або без них (рис. 2.11). Граф можна задати графічно (рис. 2.13). Вершини можуть бути підписані цифрами, літерами (рис. 2.13) або по-іншому, як бажає користувач.

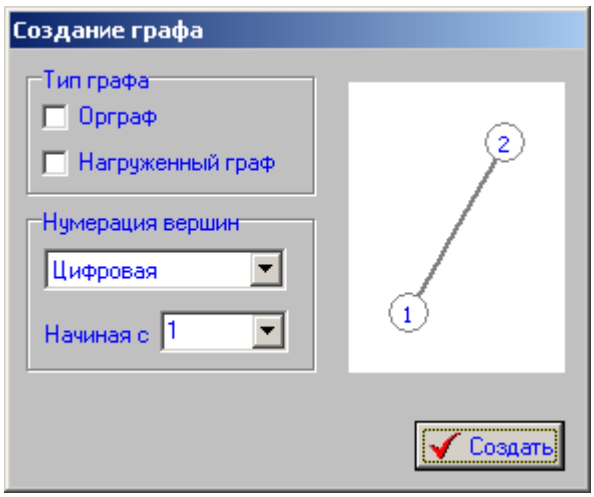


Рисунок 2.11 – Створення графа

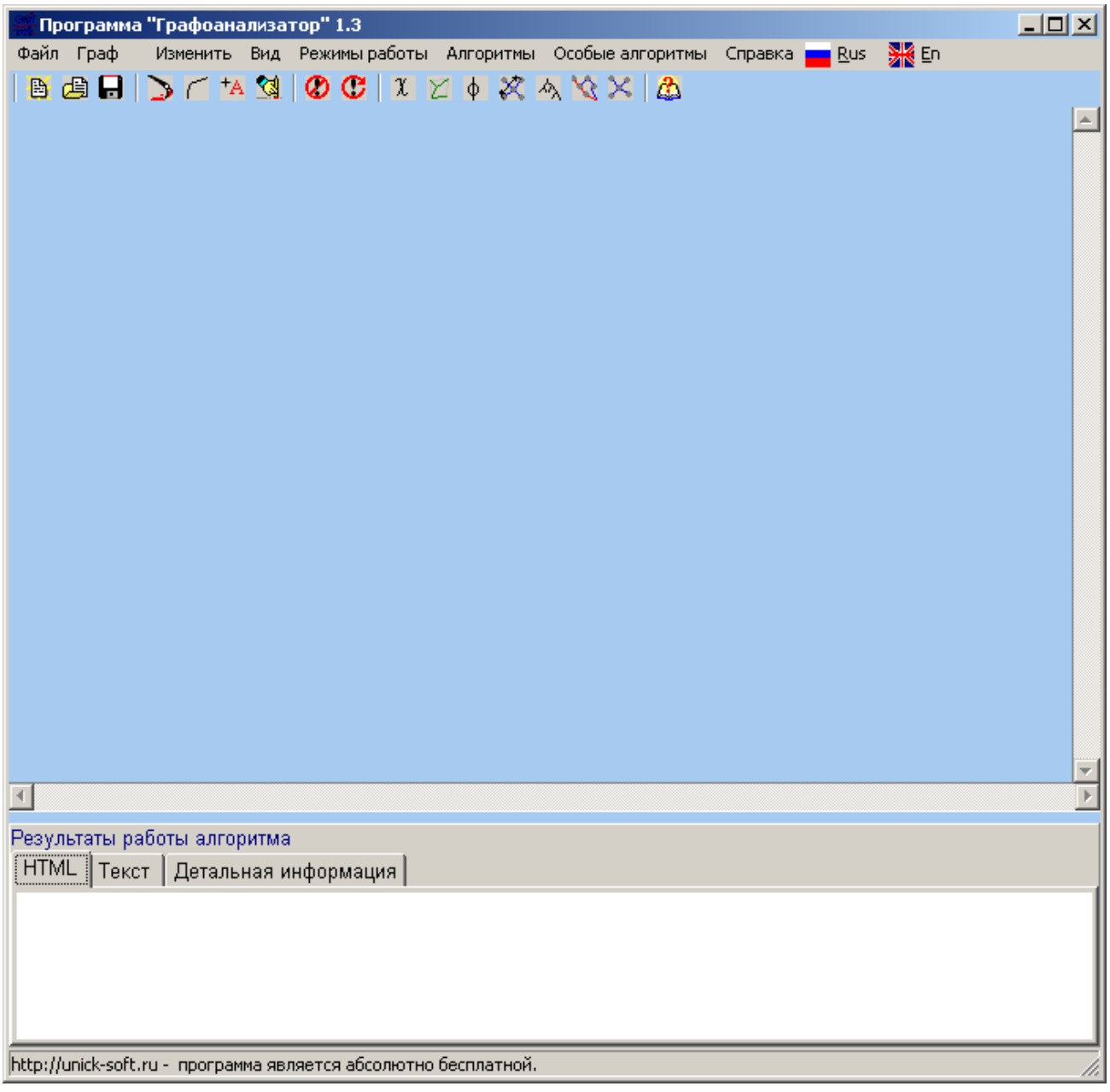


Рисунок 2.12 – Головне вікно програми

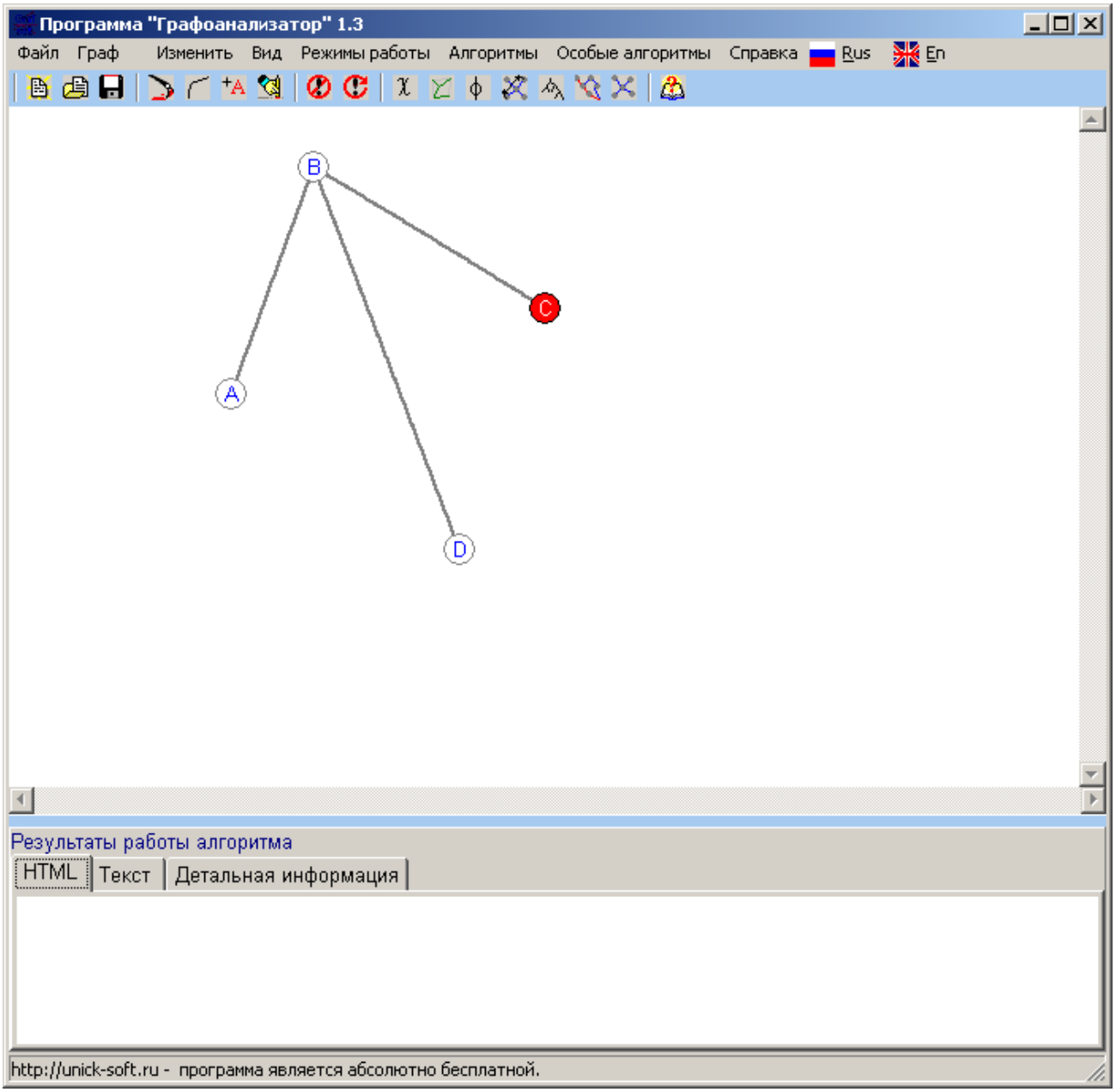


Рисунок 2.13 – Побудований граф

Для заданого графа можна отримати матрицю суміжності (рис. 2.13, 2.14). Можна змінити граф, змінюючи його матрицю суміжності (рис. 2.15, 2.16).

З матрицями інциденцій програма не працює.

Є багато задач на графах, які розв’язує програма (рис. 2.17), наприклад, пошук найкоротшого шляху між двома вершинами за різними алгоритмами.

Є детальна довідка по роботі з програмою, представлена і в мережі інтернет (http://old.unick-soft.ru/doc/graf/index.html) і pdf-файлом в програмі.

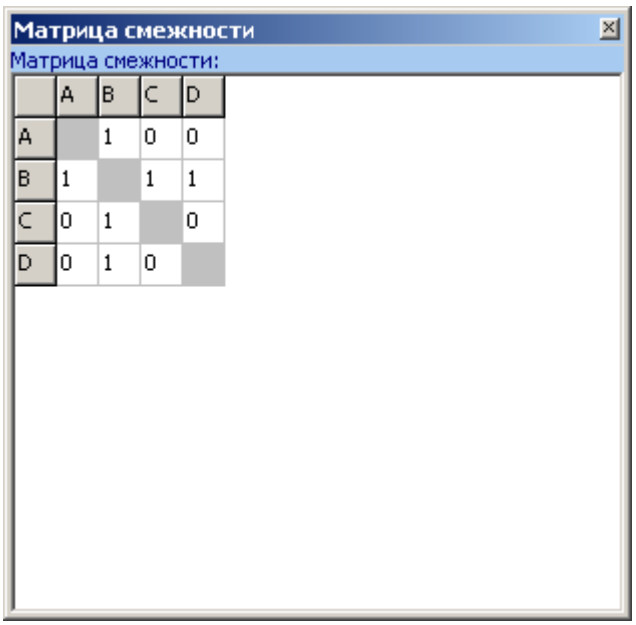


Рисунок 2.14 – Матриця суміжності графа з рис. 2.13

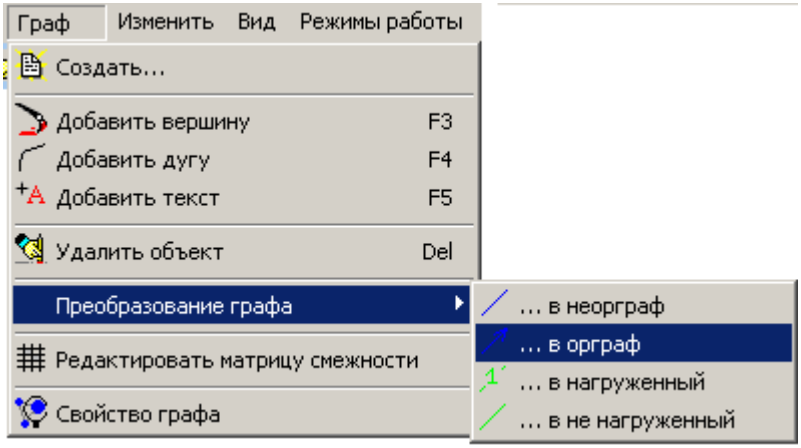


Рисунок 2.15 – Можливості програми

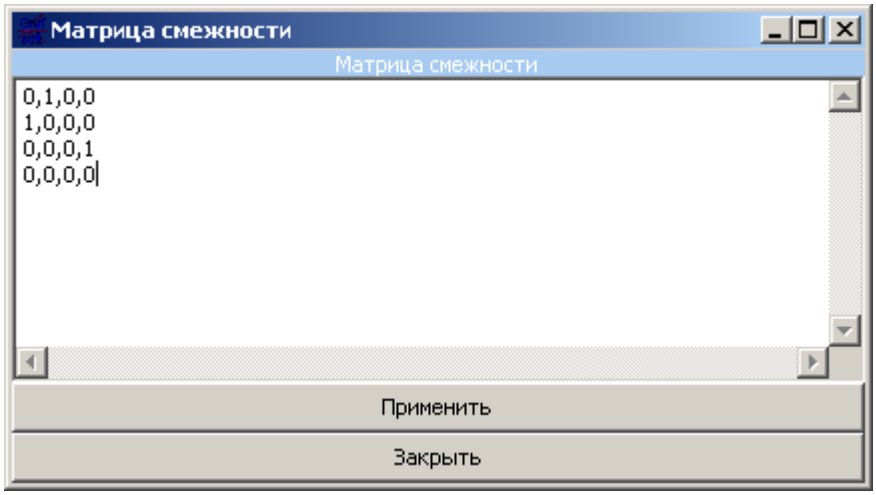


Рисунок 2.16 – Редагування графу за матрицею суміжності

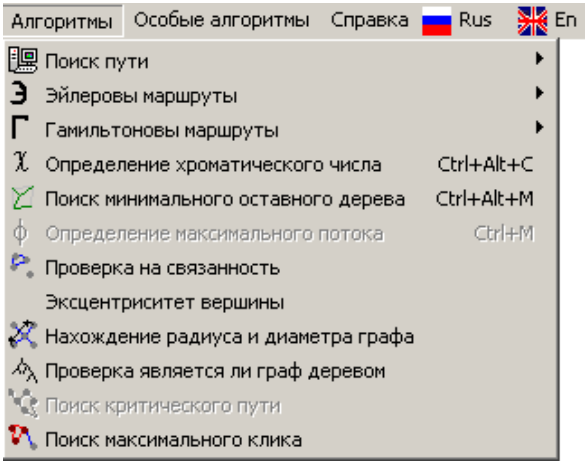


Рисунок 2.17 – Алгоритм програми

Рисунок 2.18 – Пункт меню «Про програму»

2.2. Позитивні риси розглянутих розробок

Ресурс «Редактор графів»:

Можливість задання графа матрицями та отримання графічного представлення графа. Та, навпаки, задання графа рисунком та отримання матриць суміжності, інцидентності тощо.

Широкі можливості з редагування рисунку.

Можливість побудови різних типів графів.

Наявність відео та текстової довідки.

Наявність ілюстративних прикладів.

Програма «Графоаналізатор»:

Можливість побудови різних типів графів.

Наявність детальної довідки.

Наявність алгоритмів для розв'язування широкого кола задач на графах.

Простий та інтутивнозрозщумілий інтерфейс.

2.3. Негативні риси розглянутих розробок

Ресурс «Редактор графів»:

Російськомовний інтерфейс.

При малюванні графа слід примусово видаляти текст над ребрами.

Деякий функціонал ресурсу є платним.

Програма «Графоаналізатор»:

Російськомовний інтерфейс.

Не використовуються матриці інциденцій.

Довідка, представлена в Інтернеті, показує програму з трохи іншим інтерфейсом, ніж у завантаженій версії.

2.4. Необхідність та актуальність теми

Розглянуті додатки не задовольняють навчальний процес на 100%, отже доцільною є розробка візуалізатора за темою випускової роботи.

3. ТЕОРЕТИЧНА ЧАСТИНА

3.1. Необхідні теоретичні відомості

Граф можна представити як набір точок і ліній, що сполучають пари точок. Точки називаються вершинами (або вузлами) графа, а лінії – його ребрами. Лінії можуть бути як направленими, так і ненаправленими.

Наведемо більш строге визначення.

Графом G називається сукупність двох множин $G=(V,E)$, де V – множина вершин ($V\neq\varnothing$), а $E\subset V\times V$ – множина ребер. Ребро задається парою вершин: $E=\{(v_i,v_j):v_i,v_j\in V\}$.

Розглянемо приклади графів.

Приклад 1.

Є граф (рис. 3.1):

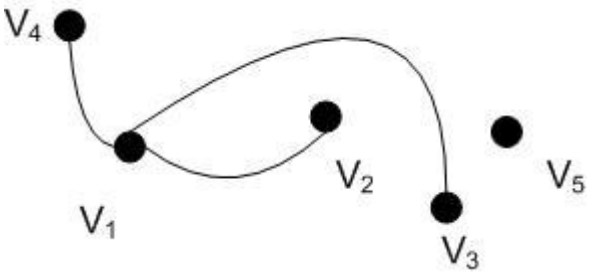


Рисунок 3.1 – Граф

Його множина вершин – $V=\{v_1,v_2,v_3,v_4,v_5\}$. Його множина ребер – це $E=\{(v_1,v_2),(v_1,v_3),(v_1,v_4)\}$.

Приклад 2.

Є граф (рис. 3.2):

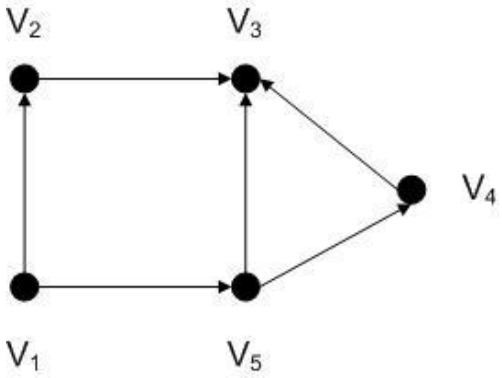
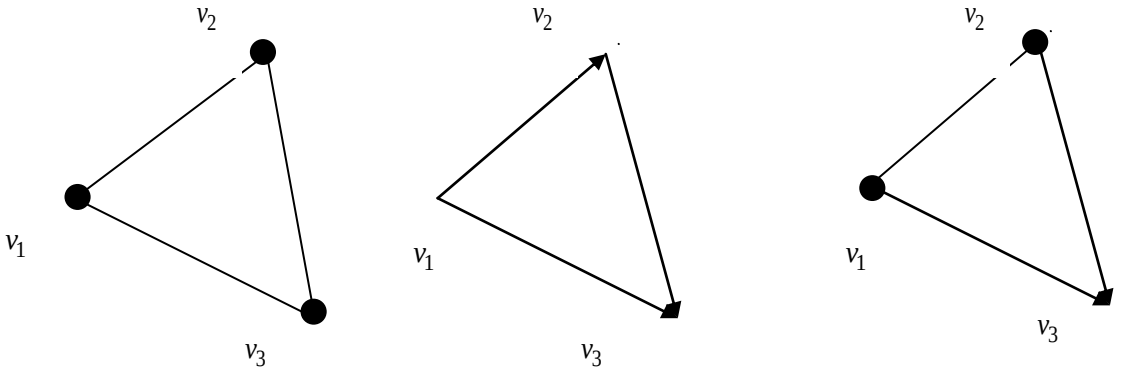


Рисунок 3.2 – Граф

Його множина вершин – це $V = \{v_1, v_2, v_3, v_4, v_5\}$. Його множина ребер – це $E = \{(v_1, v_2), (v_1, v_5), (v_2, v_3), (v_4, v_3), (v_5, v_3), (v_5, v_4)\}$.

Розрізняють неорієнтовані, орієнтовані та змішані графи.

Граф називається **неорієнтованим**, якщо пари (v_i, v_j) і (v_j, v_i) задають одне й те саме ребро. Для **орієнтованого** графа пари (v_i, v_j) і (v_j, v_i) задають різні ребра. У такому випадку для ребра також використовується термін «дуга». Можуть бути **змішані** графи, частина ребер яких орієнтована, а частина ні (рис. 3.3).



Неорієнтований граф | Орієнтований граф | Змішаний граф

Рисунок 3.3 – Приклади графів

Позначимо через $n = |V|$ – число вершин, а через $m = |E|$ число ребер графа.

Ребро $e = (v_i, v_i)$ називаються **петлею** (кінцеві вершини збігаються).

Вершини v_i і v_j називаються **суміжними**, якщо існує ребро, що їх поєднує.

Ребро $e = (v_i, v_j)$ називають **інцидентним** вершинам v_i і v_j.

Ребра називаються **суміжними**, якщо вони інцидентні одній і тій самій вершині.

Граф однозначно задається **матрицею суміжності** $A = \{a_{ij}\}_{i,j=1}^{n}$, де

$$a_{ij} = \begin{cases} 1, \textit{ якщо є ребро } (v_i, v_j); \\ 0, \textit{ в іншому разі}. \end{cases}$$

Приклад 3.

Є неорієнтований граф (рис. 3.4):

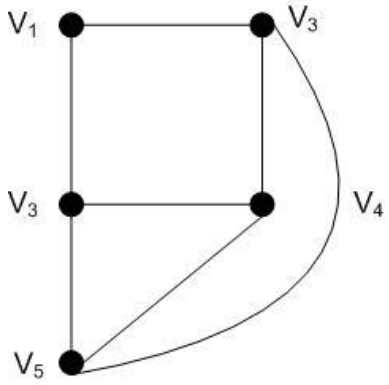


Рисунок 3.4 – Неорієнтований граф

Згідно означення матриці суміжності його можна представити у вигляді:

$$A = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{array}{c} \begin{array}{ccccc} v_1 & v_2 & v_3 & v_4 & v_5 \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix} \end{array}$$

Приклад 4.

Є орієнтований граф (рис. 3.5):

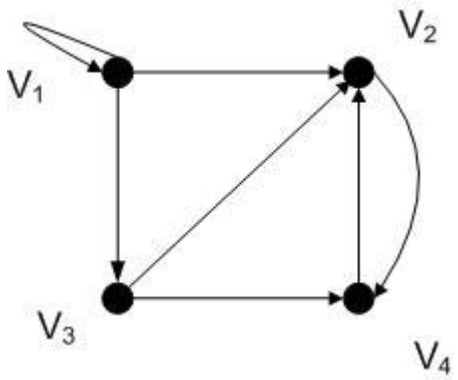


Рисунок 3.5 – Орієнтований граф

Згідно означення матриці суміжності його можна представити у вигляді:

$$A = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} \begin{array}{c} \begin{array}{cccc} v_1 & v_2 & v_3 & v_4 \end{array} \\ \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix} \end{array}$$

Граф однозначно задається **матрицею інцидентності (інциденцій)** $B = \{b_{ij}\}_{i=1,\,j=1}^{n,m}$, яка задається наступним чином.

Для неорієнтованого графа:

$$b_{ij} = \begin{cases} -1, & \textit{якщо ребро } \mathrm{e_j} \textit{ інцидентно вершині } v_\mathrm{i}; \\ 0, & \textit{якщо ребро } \mathrm{e_j} \textit{ не інцидентно вершині } v_\mathrm{i}. \end{cases}$$

Для орієнтованого графа:

$$b_{ij} = \begin{cases} -1, \textit{якщо дуга } e_\mathrm{j} \text{ входить у } \textit{вершину } v_\mathrm{i}; \\ 1, \textit{якщо дуга } \mathrm{e_j} \text{ виходить з } \textit{вершини } v_\mathrm{i}; \\ 0, \textit{якщо дуга } e_i \textit{ не інцидентна вершині } v_\mathrm{i}. \end{cases}$$

Приклад 5.

Є неорієнтований граф (рис. 3.6):

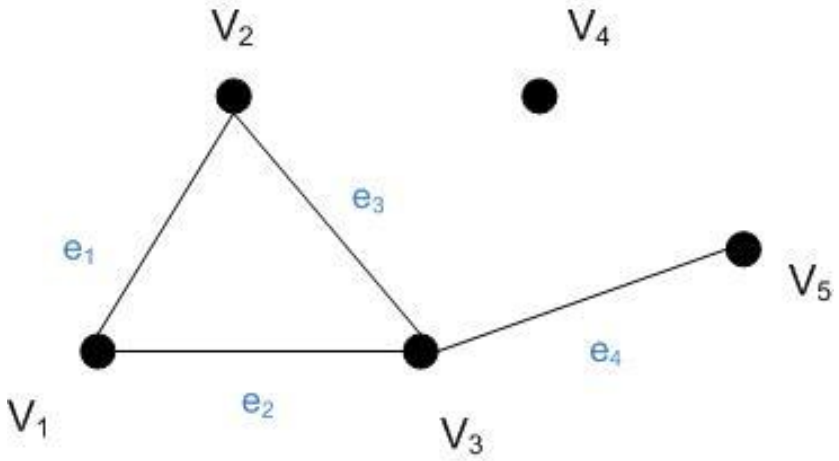


Рисунок 3.6 – Неорієнтований граф

Згідно означення матриці інцидентності його можна представити у вигляді:

$$B = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{array}{c} \begin{array}{cccc} e_1 & e_2 & e_3 & e_4 \end{array} \\ \begin{pmatrix} -1 & -1 & 0 & 0 \\ -1 & 0 & -1 & 0 \\ 0 & -1 & -1 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \end{array}$$

Приклад 6.

Є орієнтований граф (рис. 3.7):

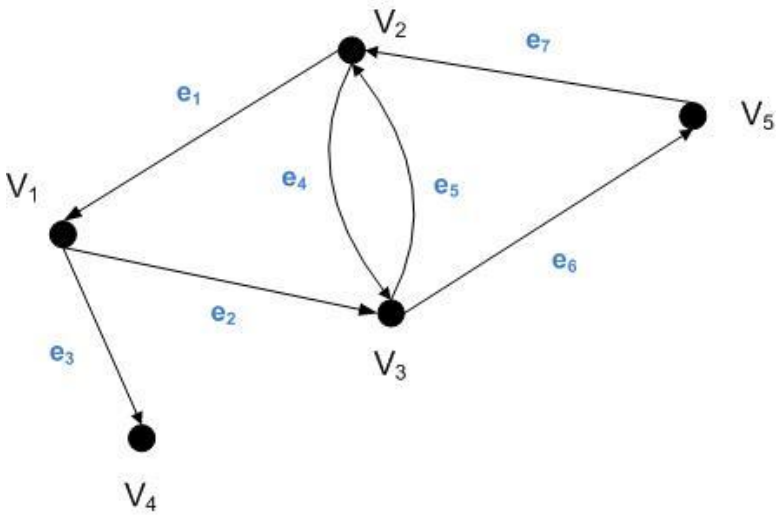


Рисунок 3.7 – Орієнтований граф

Згідно означення матриці інцидентності його можна представити у вигляді:

$$B = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{array}{c} \begin{array}{ccccccc} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 \end{array} \\ \begin{pmatrix} -1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & -1 & 0 & -1 \\ 0 & -1 & 0 & -1 & 1 & 1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix} \end{array}$$

3.2. Алгоритм побудови графа за матрицею суміжності

В бакалаврській роботі будемо розглядати лише неорієнтовані графи.

Сформулюємо алгоритм побудови неорієнтованого графа без петель за матрицею суміжності:

1. Ввести кількість вершин графа n.

2. Ввести матрицю суміжності $A = \{a_{ij}\}_{i,\,j=1,n}$.

3. На площині випадково розташувати n точок – вершин графу.

4. Переглянути всю матрицю суміжності. Якщо елемент $a_{ij} = 1$, то з'єднати ребром вершини з номерами i та j.

3.3. Алгоритм побудови графа за матрицею інцидентності

Сформулюємо алгоритм побудови неорієнтованого графа без петель за матрицею інцидентності:

1. Ввести кількість вершин графа n.

2. Ввести кількість ребер графа m.

3. Ввести матрицю суміжності $B = \{b_{ij}\}_{i=1,\, j=1}^{n,m}$.

4. На площині випадково розташувати n точок – вершин графу.

5. Переглянути всю матрицю інцидентності по стовпцям (ребрам графа). Якщо елементи $b_{ij} = -1$ та $b_{kj} = -1$, то з'єднати ребром вершини з номерами i та k. Підписати ребро як e_j.

3.4. Блок-схеми алгоритмів

На рисунках 3.8-3.9 представлена блок-схема алгоритму побудови графа за матрицею суміжності.

На рисунках 3.10-3.11 представлена блок-схема алгоритму побудови графа за матрицею інцидентності.

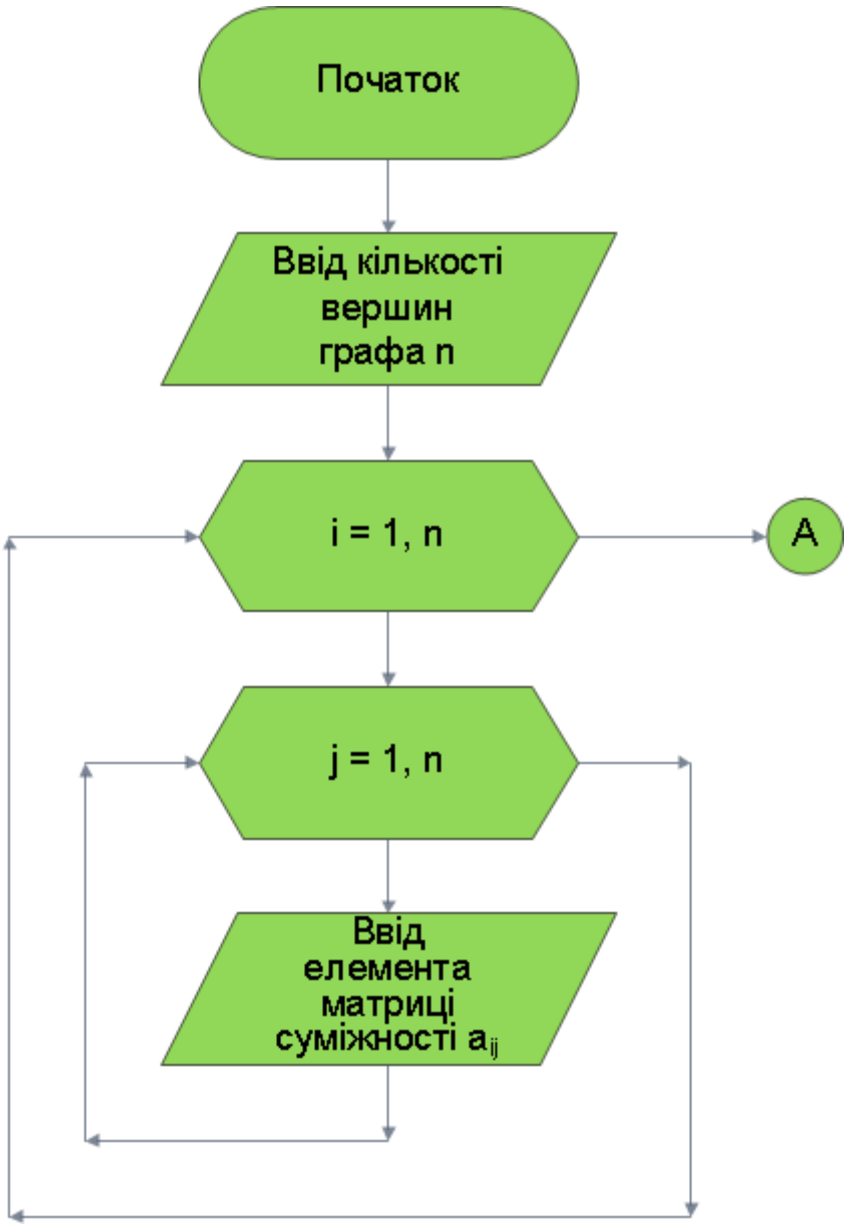


Рисунок 3.8 – Блок-схема алгоритму побуди графа за матрицею суміжності

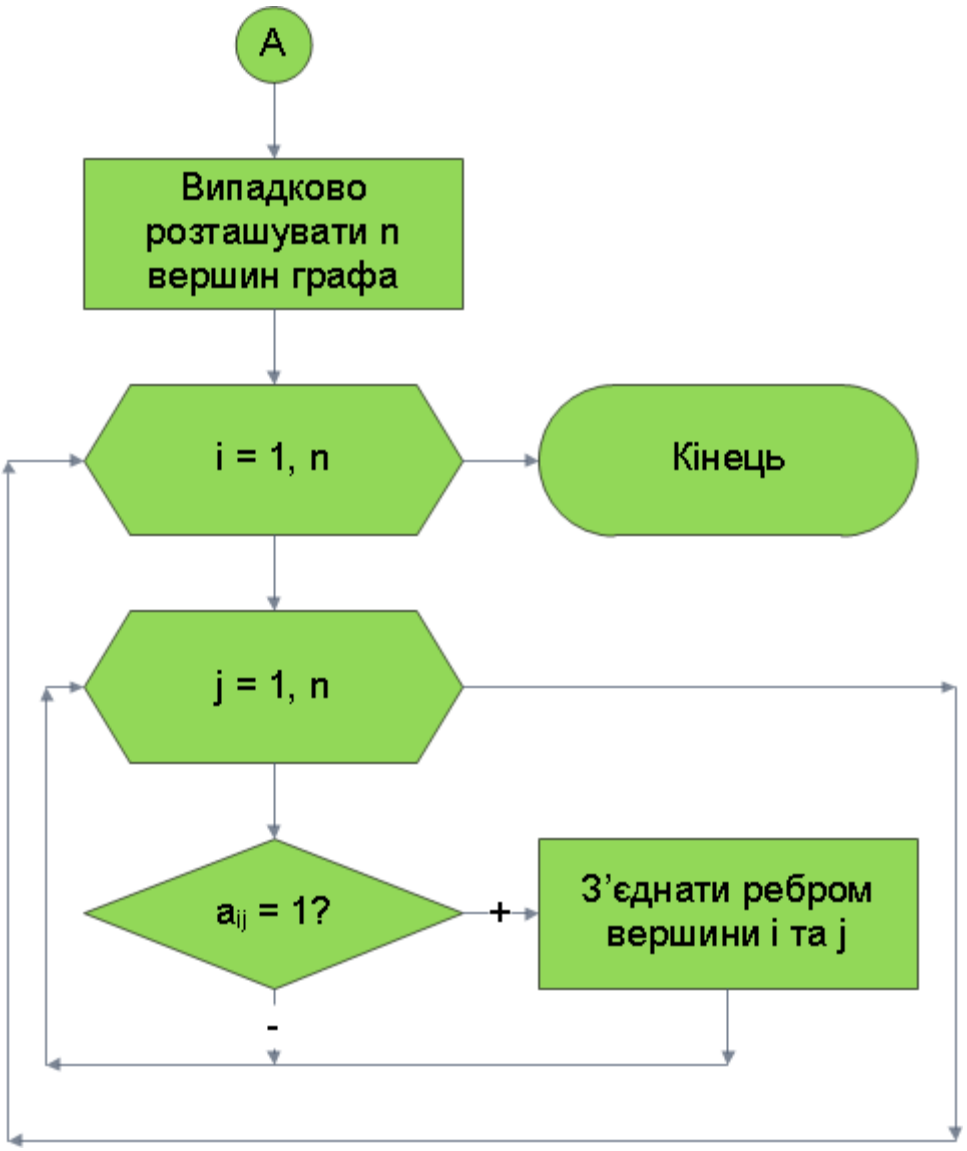


Рисунок 3.9 – Блок-схема алгоритму (продовження рис. 3.8)

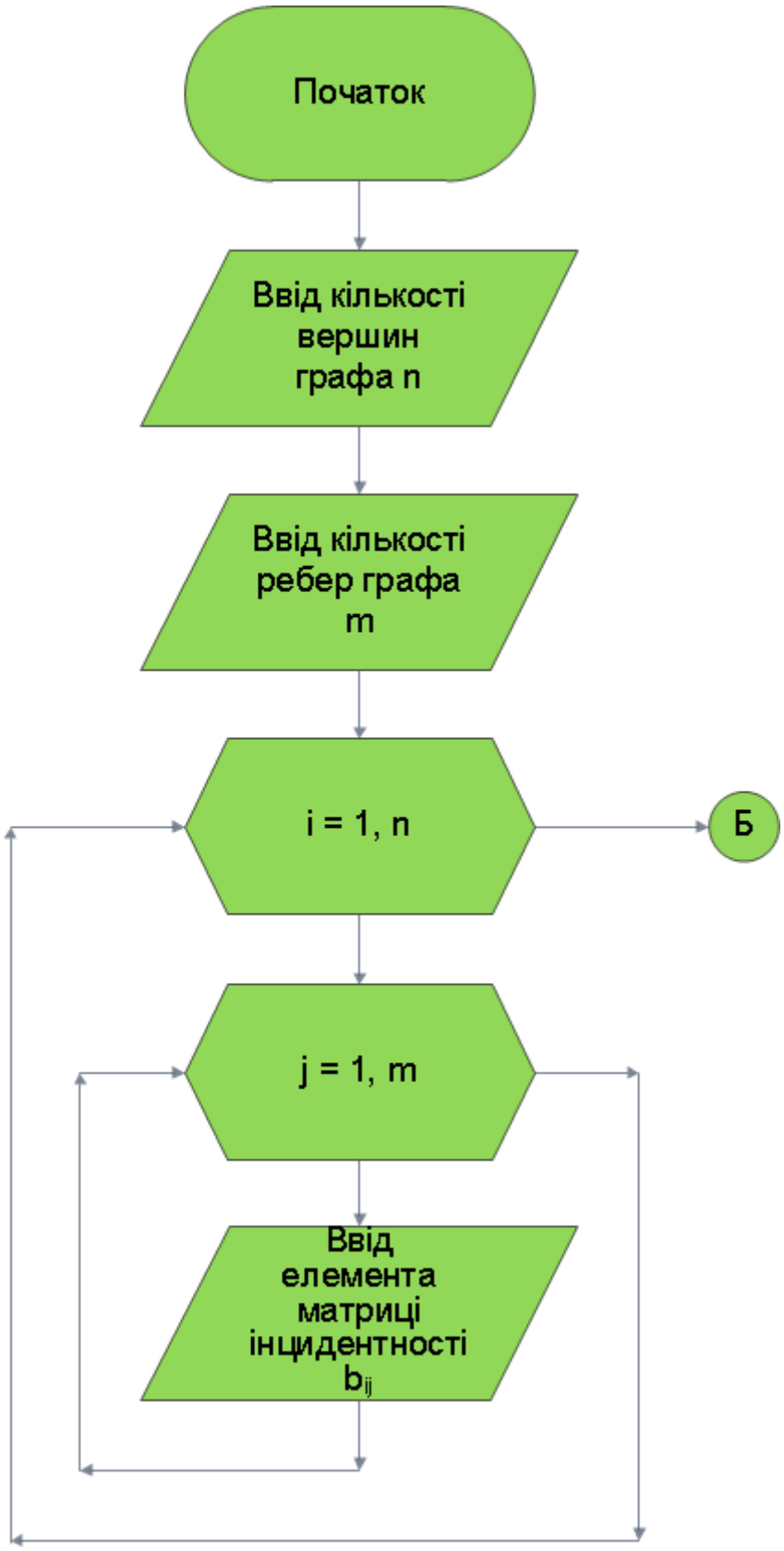


Рисунок 3.10 – Блок-схема алгоритму побуди графа за матрицею інцидентності

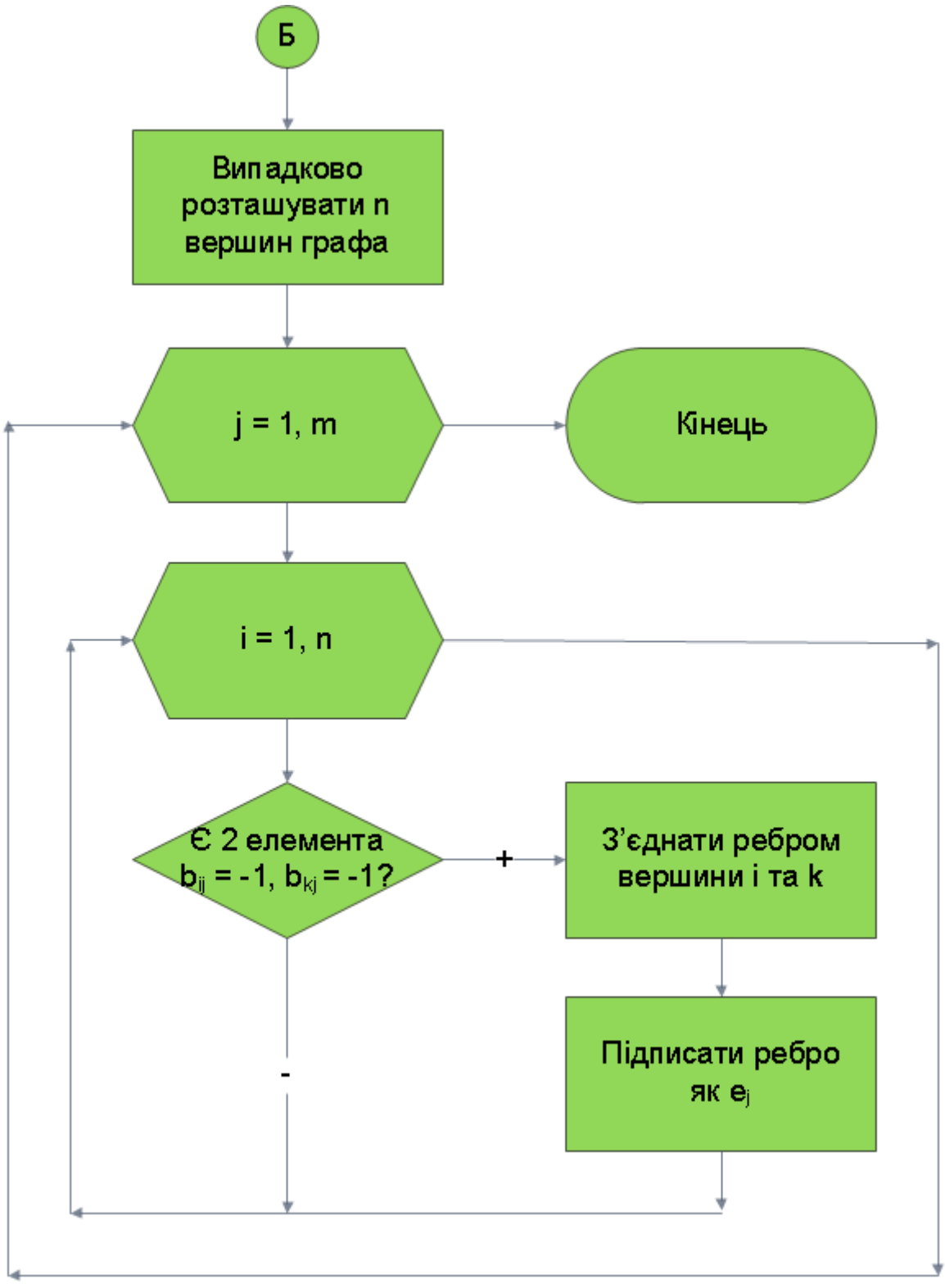


Рисунок 3.11 – Блок-схема алгоритму (продовження рис. 3.10)

4. ПРАКТИЧНА ЧАСТИНА

4.1. Опис програмної реалізації

Програма була написана у середовищі `Delphi` з використанням мови програмування `Object Pascal`.

Розглянемо побудову графа за матрицею суміжності. Було створено форму заданого вигляду (рис. 4.1):

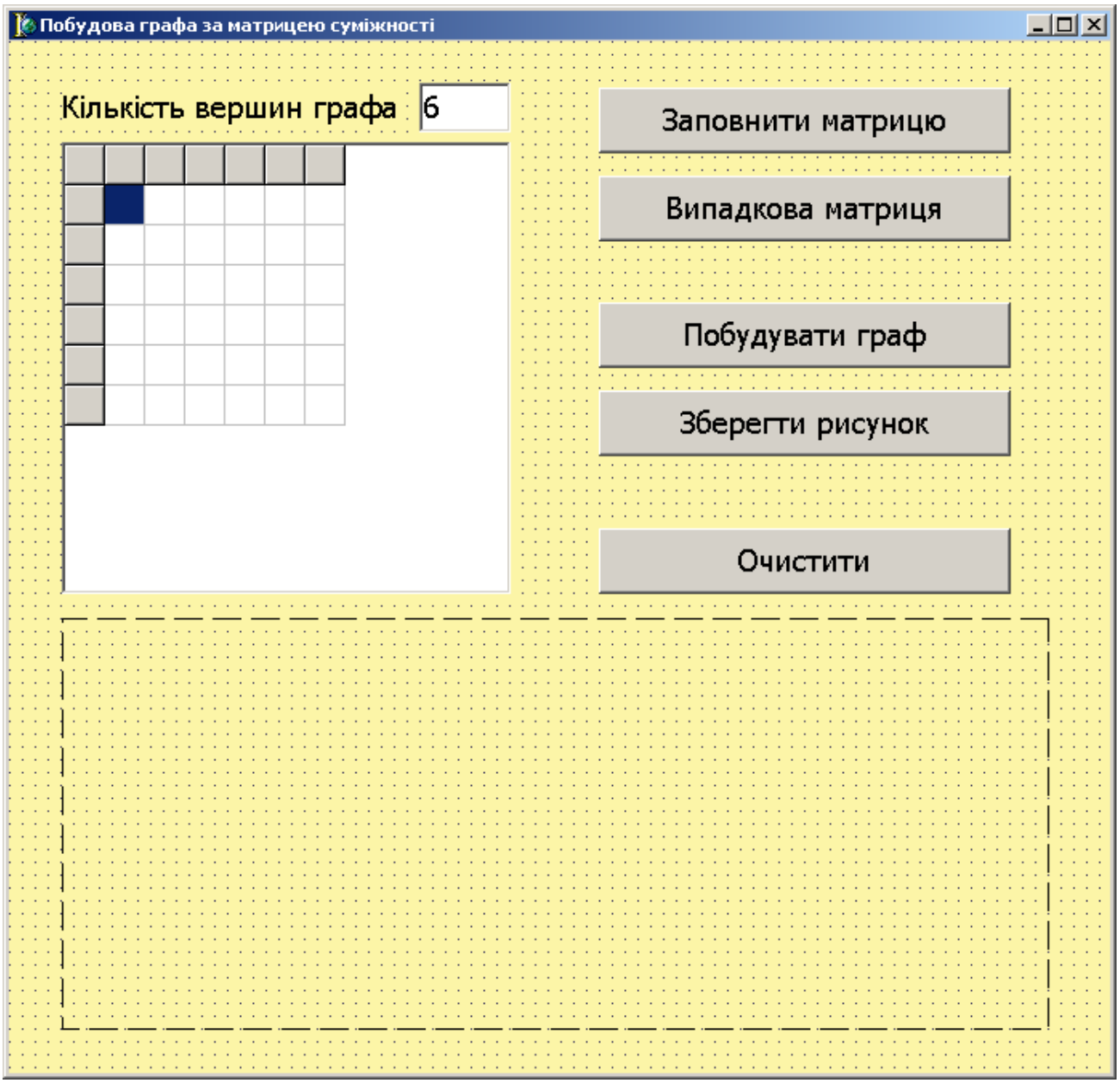


Рисунок 4.1 – Форма програми

Для форми було налаштовані такі властивості (табл. 4.1):

Таблиця 4.1 – Властивості компонента Form1 (форми)

№	Властивість	Значення властивості
1	Caption (заголовок)	«Побудова графа за матицею суміжності»
2	Width (ширина)	695
3	Height (висота)	671
4	Position (розташування на екрані)	poScreenCenter (по центру екрана)
5	Color (колір фону)	$00A7F5FC (жовтий)
6	Font (шрифт)	шрифт – «Tahoma», накреслення – «обычний», розмір – 14

На формі були розташовані компоненти (рис. 3.2):

- підпис Label1;
- текстове поле Edit1;
- таблиця StringGrid1;
- графічний компонент Image1;
- п’ять кнопок Button1, Button2, Button3, Button4, Button5.

Для компонентів було змінено властивості (див. табл. 4.2).

Далі був написаний код, який згруповано по підпрограмам. Огляд підпрограм представлено у таблиці 4.3.

Програма для побудови графа за матрицею інцидентності створювалась аналогічно.

Програма має обмеження – намалювати можна граф не більше ніж з 10 вершинами. Обмеження легко усувається виправленнями значення константи Max_nodes.

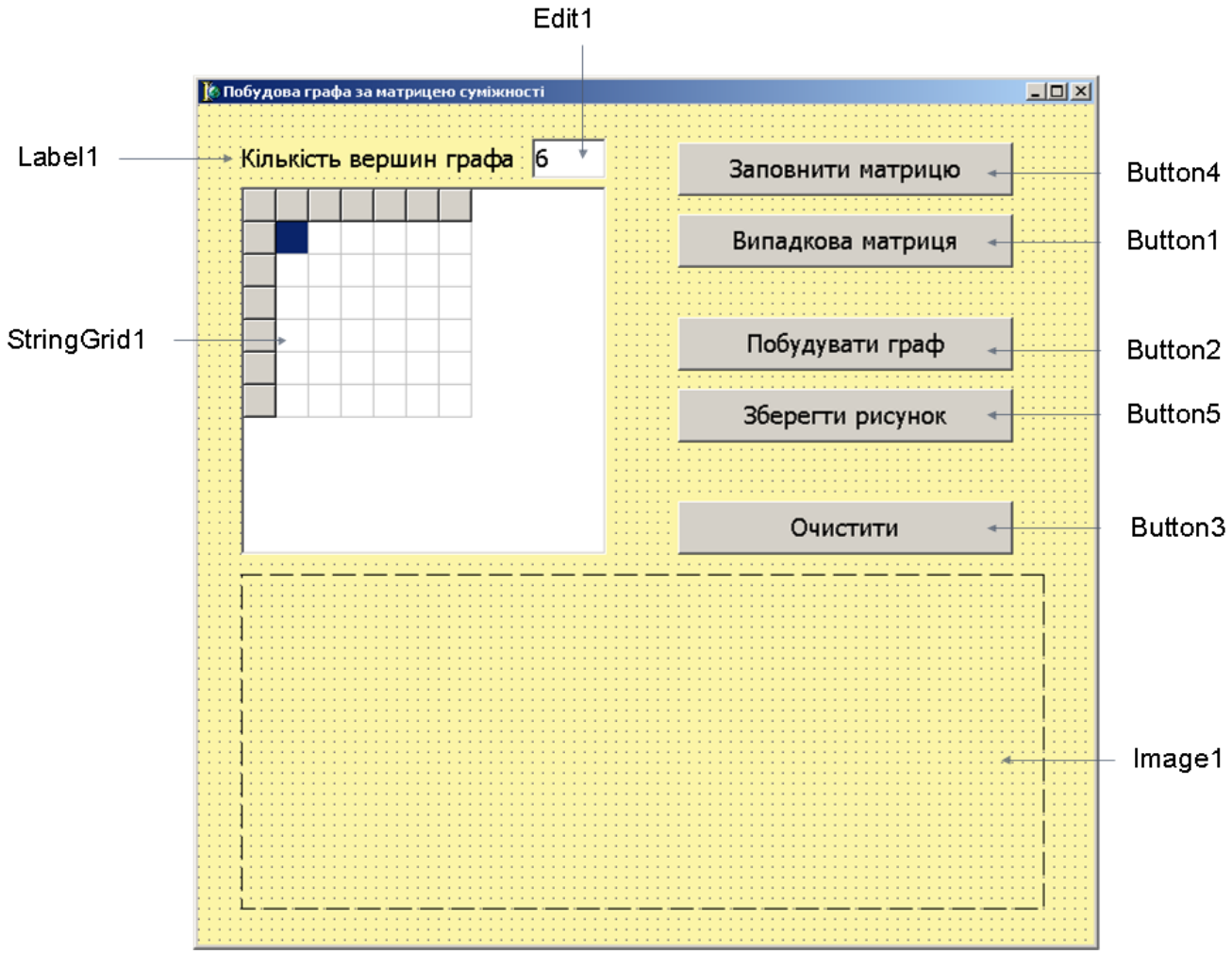


Рисунок 4.2 – Структура форми

Таблиця 4.2 – Властивості компонентів

№	Компонент	Властивість	Значення властивості
1	Label1	Caption (підпис)	«Кількість вершин графа»
2	Edit1	Text (текст)	«6»
3	Button1	Caption (заголовок)	«Випадкова матриця»
4	Button2	Caption (заголовок)	«Побудувати граф»
5	Button3	Caption (заголовок)	«Очистити»
6	Button4	Caption (заголовок)	«Заповнити матрицю»
7	Button5	Caption (заголовок)	«Зберегти рисунок»

Продовження таблиці 4.2 – Властивості компонентів

№	Компонент	Властивість	Значення властивості
8	StringGrid1	FixedCols (кількість фіксованих колонок)	1
9		FixedRows (кількість фіксованих рядків)	1
10		ColCount (кількість колонок)	7
11		RowCount (кількість рядків)	7
12		Width (ширина)	280
13		Height (висота)	282
14		Font (шрифт)	шрифт – «Tahoma», накреслення – «обычний», розмір – 12
15		DefaultColWidth (ширина колонок по замовченню)	24
16		DefaultRowHeight (висота рядків по замовченню)	24
17		Options.goEditing (можливість редагування)	true
18		Options.goTabs (можливість переходу по клітинкам за допомогою клавіші «Tab»)	true
19		Options.goAlwaysShowEditor (завжди показувати курсор)	true

Код програми першого візуалізатора розташовано у додатку А.

Таблиця 4.3 – Структура програми

№	Заголовок підпрограми	Призначення підпрограми
1	`procedure TForm1.Button4Click (Sender: TObject);`	Відповідає кнопці «Заповнити матрицю». Відбувається при клацанні по кнопці. Встановлюються розміри таблиці. Головна діагональ матриці заповнюється нульовими елементами.
2	`procedure TForm1.Button1Click (Sender: TObject);`	Відповідає кнопці «Випадкова матриця». Відбувається при клацанні по кнопці. Матриця суміжності заповнюється елементами (нулями та одиницями) у випадковому порядку.
3	`procedure TForm1.Button2Click (Sender: TObject);`	Відповідає кнопці «Побудувати граф». Відбувається при клацанні по кнопці. За заданою матрицею суміжності будується граф.
4	`procedure TForm1.Button5Click (Sender: TObject);`	Відповідає кнопці «Зберегти граф». Відбувається при клацанні по кнопці. Зберегає граф до файлу *«graph.jpg»*.
5	`procedure TForm1.Button3Click (Sender: TObject);`	Відповідає кнопці «Очистити». Відбувається при клацанні по кнопці. Очищає текстове поле (кількість вершин), таблицю (матрицю) та поле виводу графу.
6	`procedure TForm1.FormCreate (Sender: TObject);`	Відбувається при створенні форми. Включається датчик випадкових чисел.
7	`procedure TForm1.FormShow (Sender: TObject);`	Відбувається при показі форми. Ставиться курсор у поле «Кількість вузлів».

4.2. Інструкція по роботі з візуалізатором

Розглянемо роботу візуалізатора. Спочатку з’являється меню (рис. 4.3):

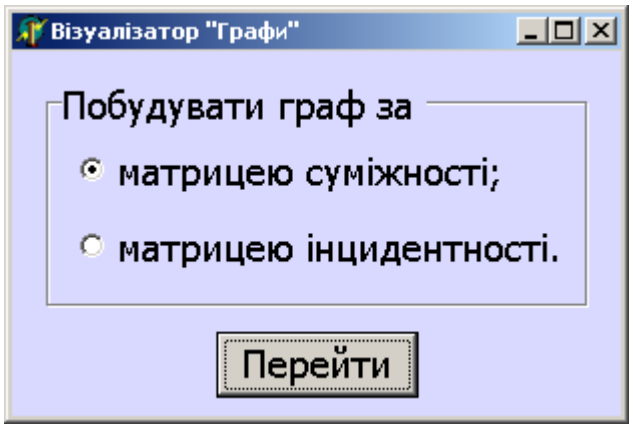


Рисунок 4.3 – Вигляд програми після запуску

Розглянемо роботу візуалізатора для побудови графа за матрицею суміжності.

Програма буде працювати за наступною схемою:

1) Користувач вводить кількість вузлів графа (рис. 4.4), по замовченню пропонується шість вершин.

2) Якщо користувач хоче ввести матрицю з клавіатури, то натискає кнопку *«Заповнити матрицю»*. При цьому встановлюється правильна вимірність матриці, а по головній діагоналі ставлять нулі (рис. 4.5). І далі матриця заповнюється користувачем нулями та одиницями.

3) Якщо користувач хоче згенерувати матрицю суміжності випадково, то натискає кнопку *«Випадкова матрицю»* (рис. 4.6).

4) Після заповнення матриці (з клавіатури або випадковим чином) натискається кнопка *«Побудувати граф»* – на поверхні компонента *Image* з’являється намальований граф (рис. 4.7), при цьому координати точок на поверхні беруться випадково. Тому при наступному натисненні цієї кнопки розташування вузлів буде іншим (рис. 4.8). Так можна прийнятний варіант зображення графа.

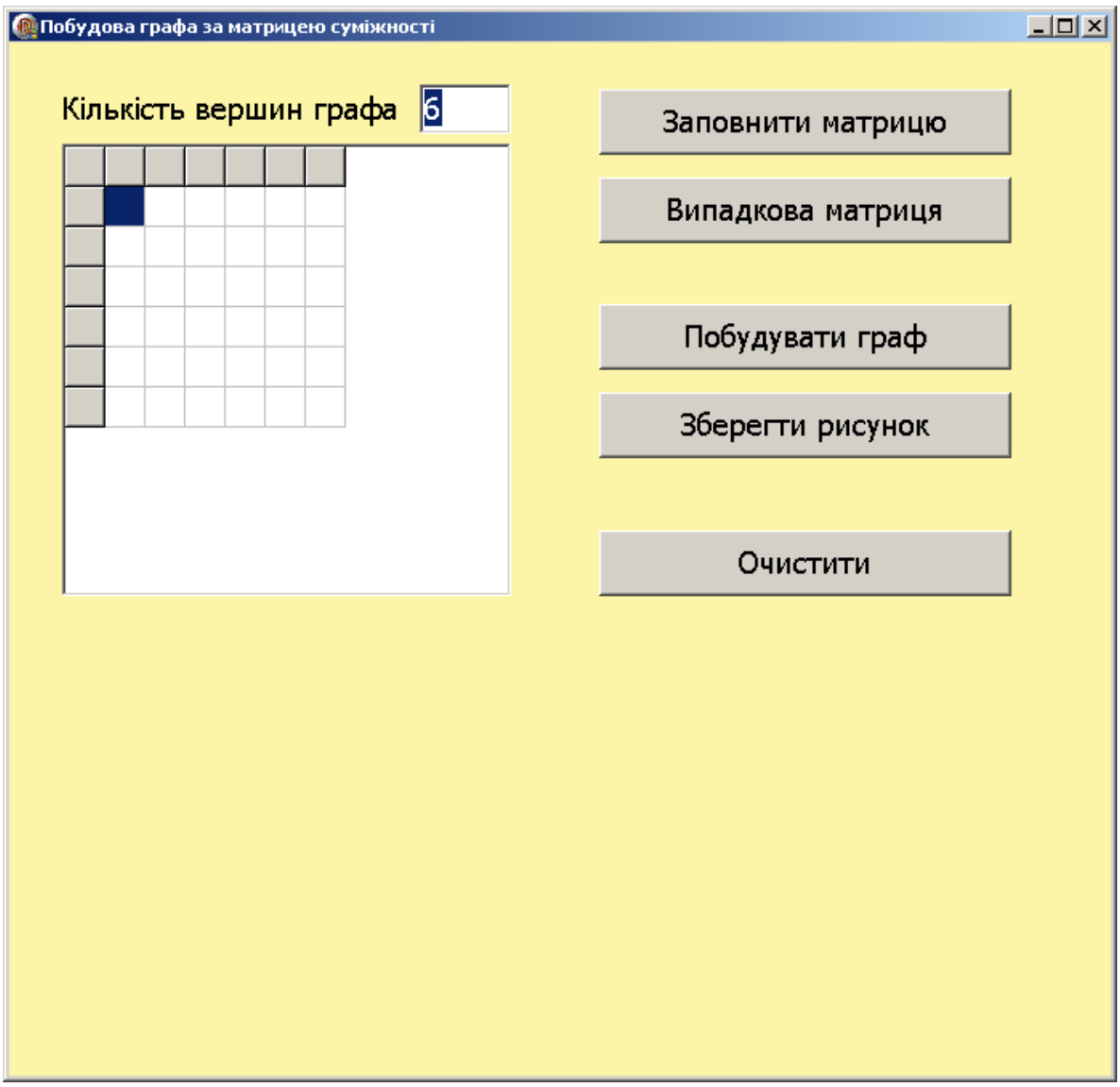


Рисунок 4.4 – Вигляд програми після запуску

5) Кнопка *«Зберегти рисунок»* дозволяє зберегти граф до файлу *«graph.jpg»* (рис. 4.9). Файл буде створений в папці з програмою. У разі повторного натиснення кнопки – файл перезаписується.

6) Кнопка *«Очистити»* повертає вікно у початковий стан (рис. 4.3).

Розглянемо роботу візуалізатора для побудови графа за матрицею інцидентності (рис. 4.10).

Цей візуалізатор (рис. 4.11-4.14) працює аналогічно до попереднього (рис. 4.4-4.9). Відмінність полягає у можливості підписувати ребра (рис. 4.13-4.14).

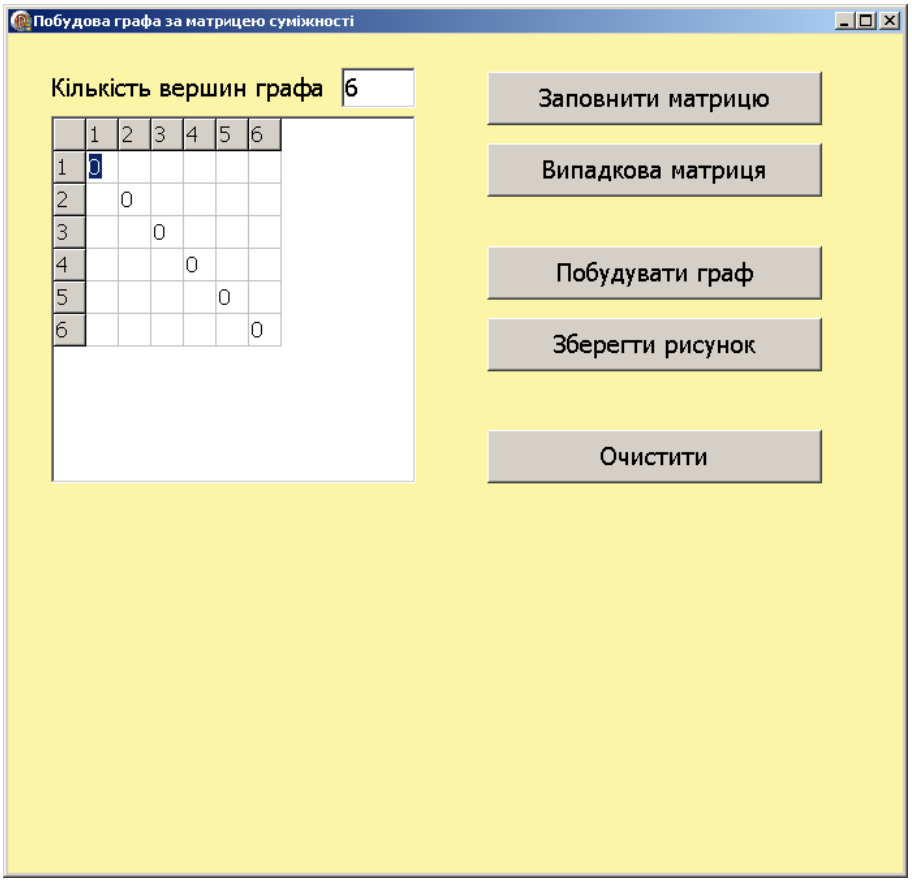


Рисунок 4.5 – Вигляд програми після натиснення кнопки «Заповнити матрицю»

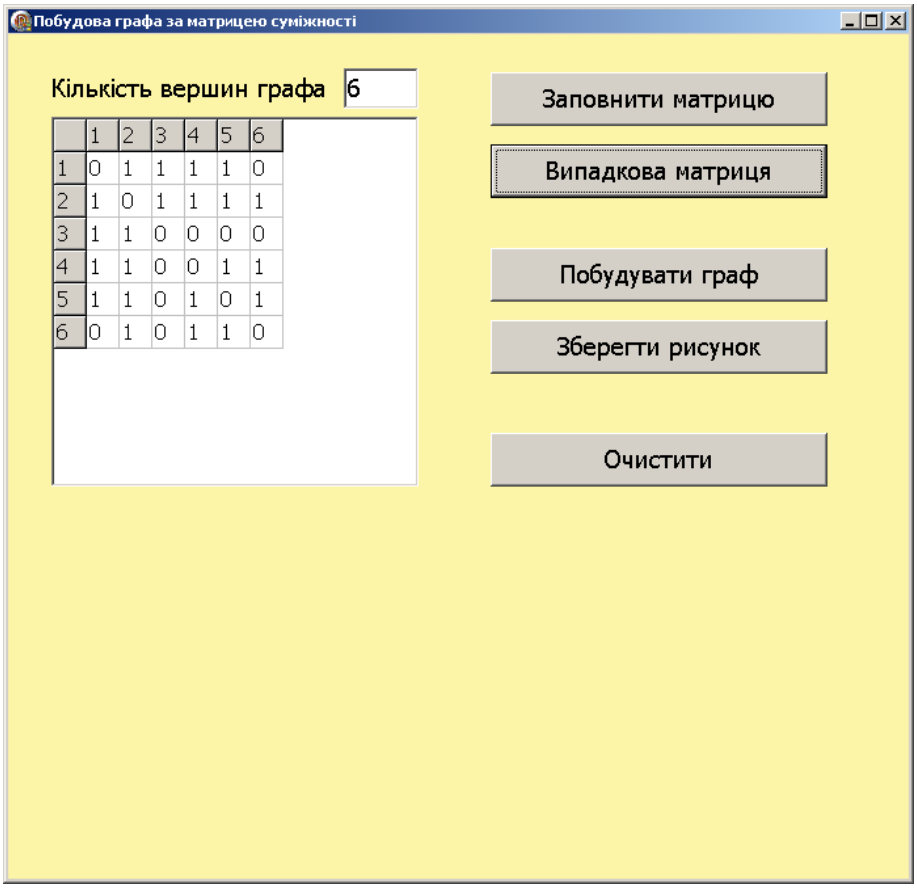


Рисунок 4.6 – Вигляд програми після натиснення кнопки «Випадкова матрицю»

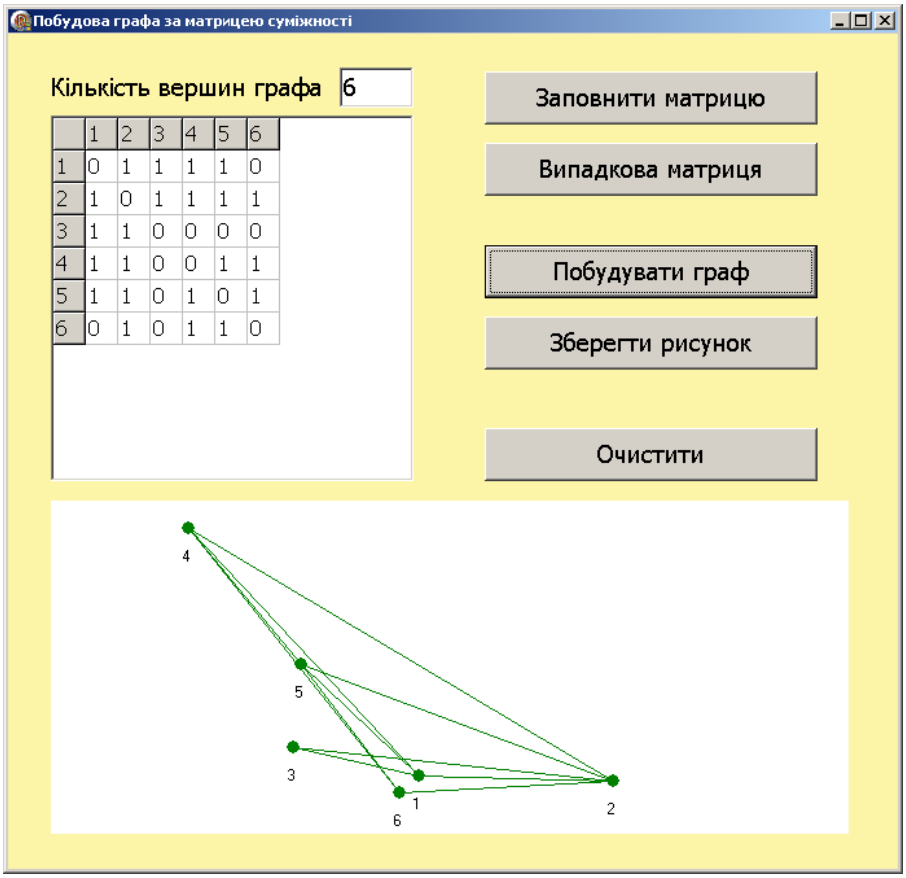


Рисунок 4.7 – Вигляд програми після натиснення кнопки «Побудувати граф»

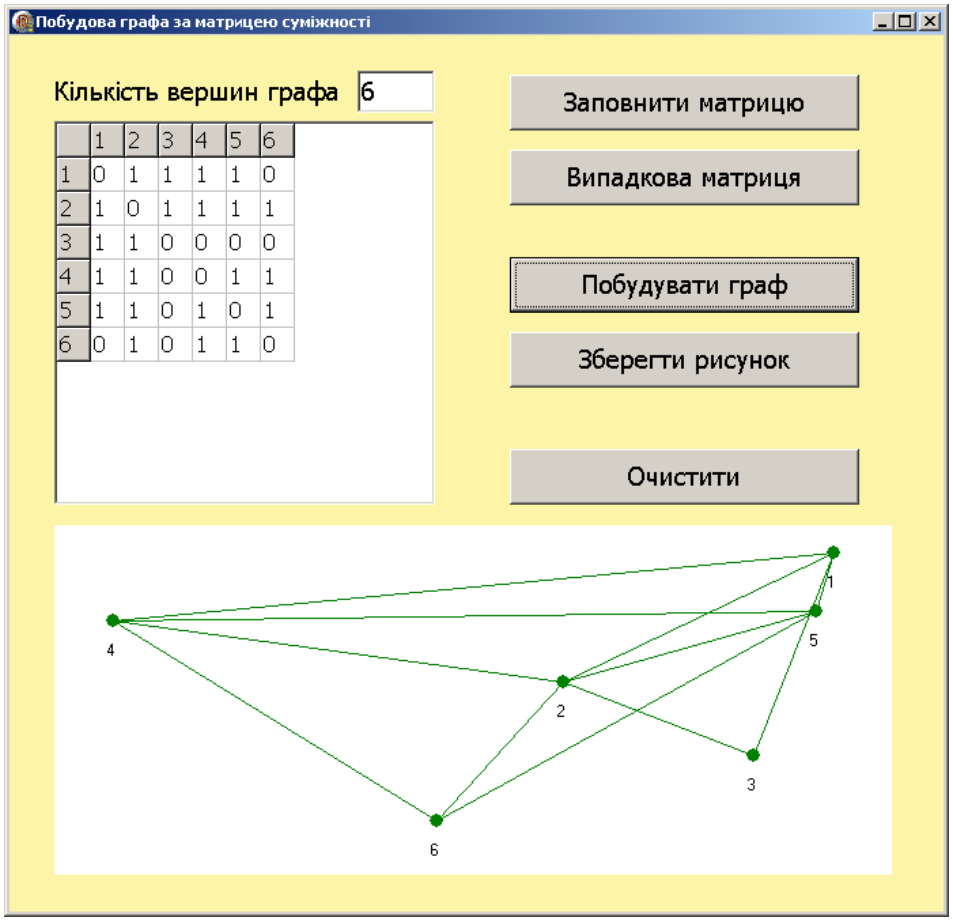


Рисунок 4.8 – Програма після чергового натиснення кнопки «Побудувати граф»

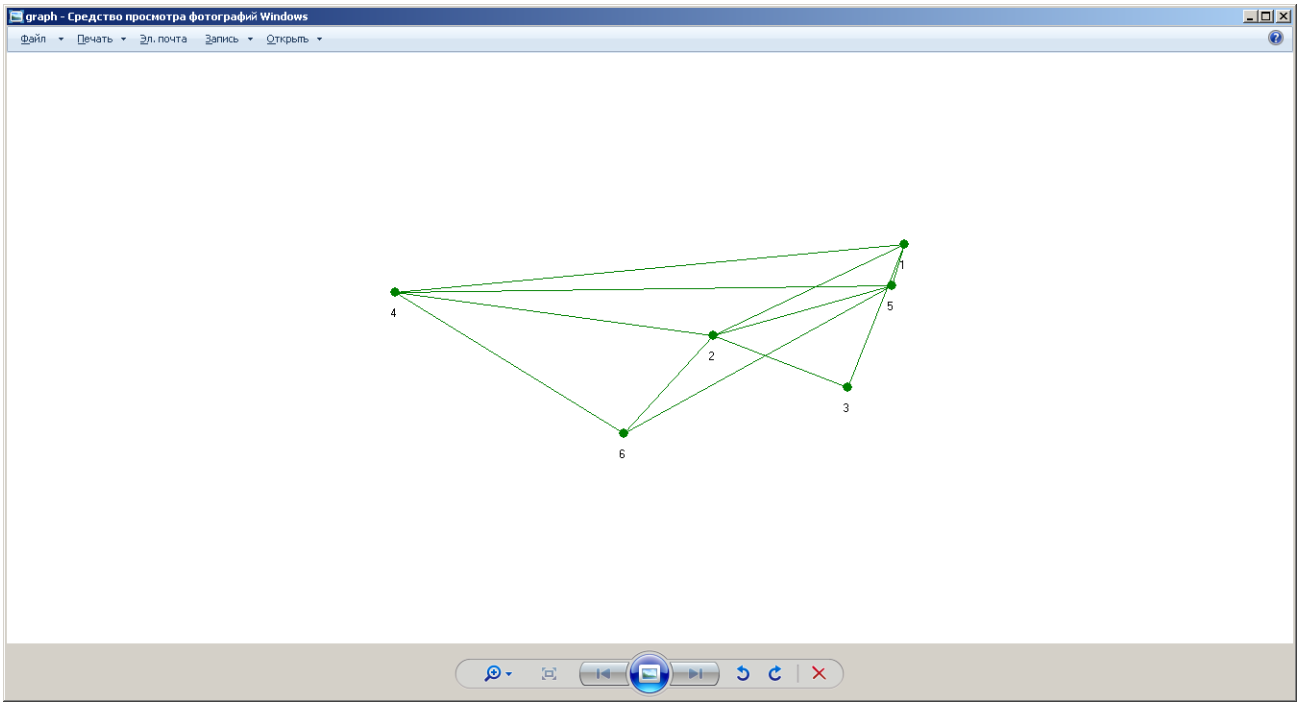


Рисунок 4.9 – Графічний файл, до якого збережено рисунок побудованого графа

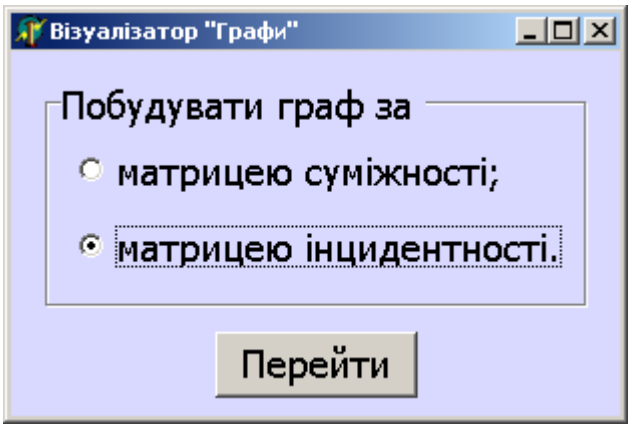


Рисунок 4.10 – Вигляд програми після запуску

Для переносу програми на інші комп’ютери слід переносити в одній папці 3 файли «Візуалізатор.exe», «S.exe», «I.exe». Назви останніх двох файлів не змінювати.

Така схема пов’язана з тим, що є дві окремі програми, представлені файлами «S.exe», «I.exe», а третя програма – «Візуалізатор.exe» – утворює меню вибору та запускає відповідний візуалізатор («S.exe» – для матриць суміжності; «I.exe» – для матриць, інцидентності).

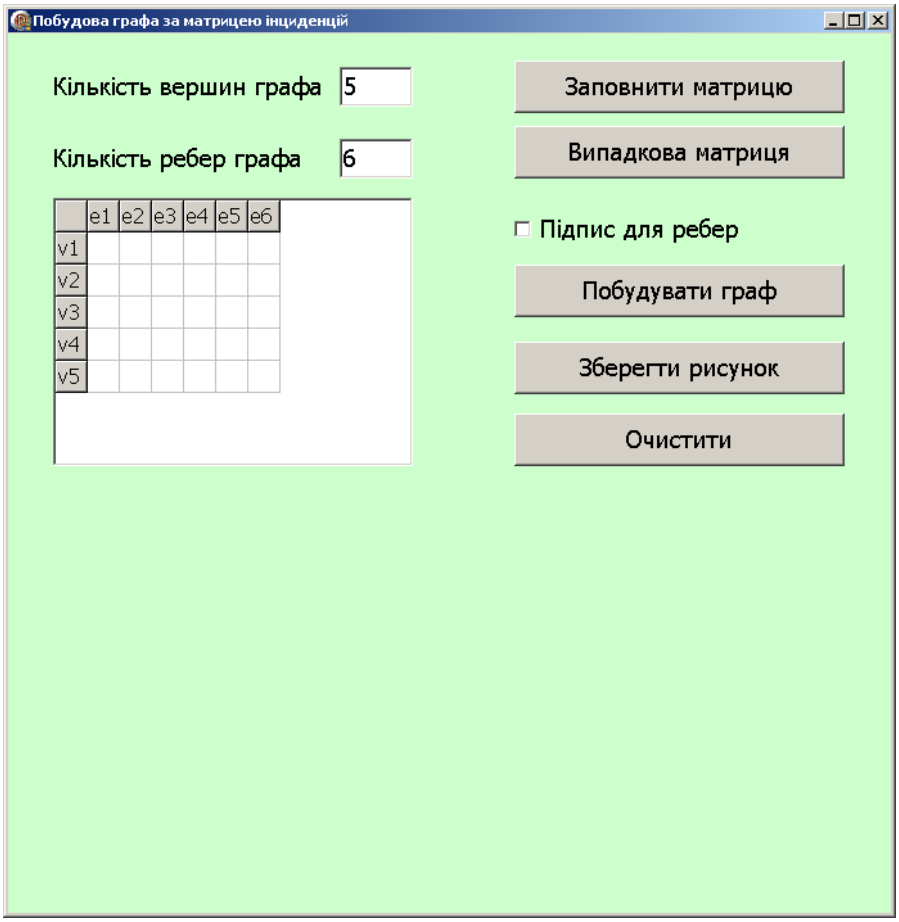


Рисунок 4.11 – Візуалізатор з побудови графа за матрицею інцидентності

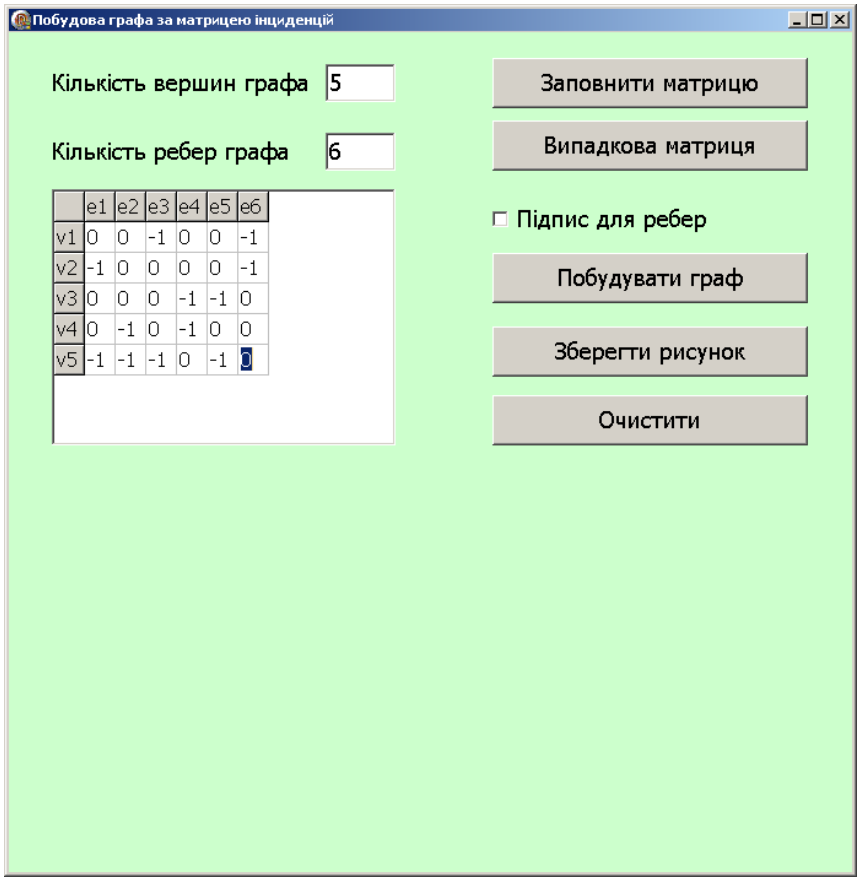


Рисунок 4.12 – Вигляд програми із заданою матрицею

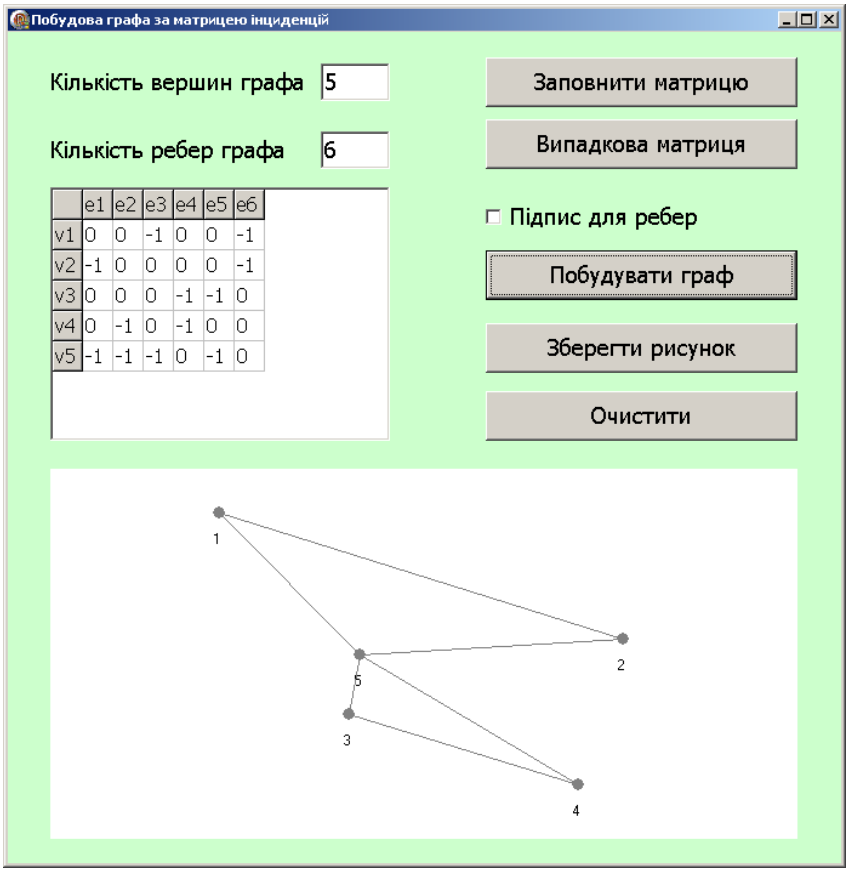


Рисунок 4.13 – Вигляд програми після натиснення кнопки «Побудувати граф»

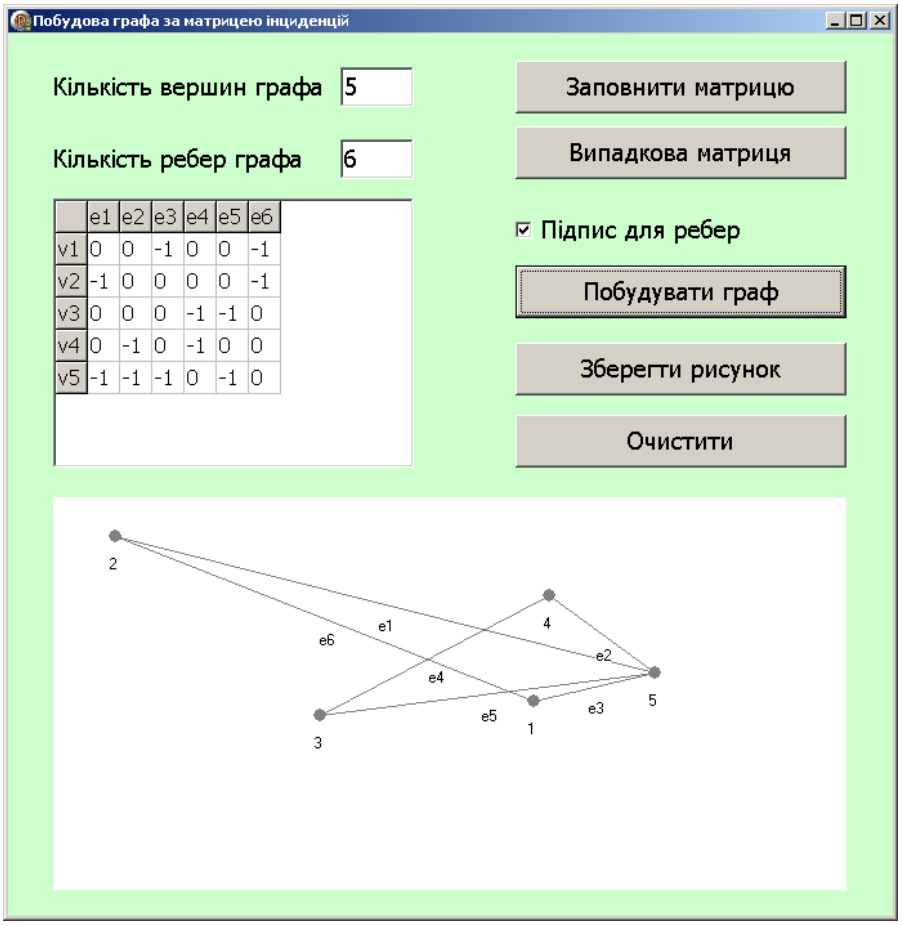


Рисунок 4.14 – Вигляд програми при поставленому прапорці «Підпис для ребер»

ВИСНОВКИ

В бакалаврській роботі було досліджено теми «Графи. Основні означення», «Графи. Способи представлення у комп’ютері». За основне джерело інформації був взятий дистанційний курс «Алгоритми і структури даних» Полтавського університету економки і торгівлі.

Було переглянуто деякі з наявних у мережі Інтернет візуалізаторів, в яких будуються матриці суміжності та матриці інциденцій за графом і навпаки, за заданими матрицями рисуються графи.

Було створено алгоритми візуалізаторів, в яких за заданою матрицею суміжності або інцидентності будується неорієнтований граф без петель.

Було накреслено блок-схеми алгоритмів.

Була написана програма візуалізатора на мові об’єктно-орієнтованого програмування Object Pascal в середовищі візуального програмування Delphi.

Програма перевірена та описана.

В подальшому доцільно запрограмувати інші візуалізатори: рисування графа за матрицею ваг, списків суміжності тощо.

СПИСОК ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Ємець О. О. Дискретна математика: Навчальний посібник. Вид. 2-ге, допов. / О. О. Ємець, Т. О. Парфьонова. – Полтава: РВВ ПУСКУ, 2009. – 287 с.

2. Ємець О.О. Дистанційний курс Полтавського університету економіки та торгівлі «Алгоритми та структури даних» для студентів спеціальностей «Комп'ютерні науки та інформаційні технології», «Комп'ютерні науки» / О. О. Ємець. – [Електронний ресурс].

3. Ємець О. О. Методичні рекомендації до виконання бакалаврської роботи для студентів спеціальності 122 «Комп`ютерні науки та інформаційні технології» освітня програма «Комп`ютерні науки» галузь знань – 12 «Інформаційні технології» / О. О. Ємець. – Полтава: ПУЕТ, 2017. – 71 с.

4. Ємець О. О. Методичні рекомендації щодо оформлення пояснювальних записок до курсових проектів (робіт) для студентів спеціальності 122 «Комп`ютерні науки та інформаційні технології» освітня програма «Комп`ютерні науки» галузь знань – 12 «Інформаційні технології» / О. О. Ємець. – Полтава: ПУЕТ, 2017. – 69 с.

5. Кормен Т. Алгоритмы: построение и анализ / Т. Кормен, Ч. Лейзерсон, Р. Риверст. – М.: МНЦО, 2000. – 960 с.

6. Красиков И. В. Алгоритмы. Просто как дважды два / И. В. Красиков, И. Е. Красикова. – М.: Эксмо, 2006. – 256 с.

7. Осипов Д. Л. Delphi. Программирование для Windows, OS X, iOS и Android / Л. Д. Осипов. – СПб.: БХВ-Петербург, 2014. – 464 с.

8. Соколов О. Ю. Інформатика для інженерів / О. Ю. Соколов, І. Т. Зарецька, Г. М. Жолткевич, О. В. Ярова. – Харків: Факт, 2006. – 424 с.

9. Стивенс Р. Delphi. Готовые алгоритмы / Р. Стивенс. – М.: ДМК Пресс; СПб.: Питер, 2004. – 384 с.

10. Фленов М. Е. Библия программиста (Delphi) / М. Е. Фленов. – БХВ-Петербург, 2011. – 674 с.