МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
Національний технічний університет України
Факультет інформатики та обчислювальної техніки
«Київський політехнічний інститут імені Ігоря Сікорського»
Кафедра інформаційних систем та технологій

Звіт
з лабораторної роботи № 3
«Операції над матрицями графів»
з дисципліни
«Дискретна математика»

Варіант № 25

Перевірила:
доц. Рибачук Людмила Віталіївна

Виконала: Павлова Софія
Студентка гр. ІС-12 , ФІОТ
1 курс,
залікова книжка № ІС-1224

Київ 2021

ЗМІСТ

ВСТУП

Тема: Операції над матрицями графів**.**

Мета: Дослідити властивості операції над матрицями графів та їх використання щодо визначення типів зв’язності.

Обладнання: Персональні комп’ютери.

ХІД РОБОТИ

Завдання:

Завдання

Реалізувати програмне застосування (програму), яке виконує наступні функції. Причому на вхід програми подається вхідний файл з описом графу, зі структурою, яка вказана у практичному завданні №1 «Представлення графів».

1. Визначити матриці відстаней та досяжності графу. Програма за запитом користувача виводить на екран та/або у файл матрицю відстаней *D* та матрицю досяжності *R* графу.
2. Визначити наявність простих циклів у графі. Програма визначає чи наявні у графі прості цикли та, в разі позитивної відповіді, виводить деякі цикли на екран.
3. Визначити тип зв'язності графу. Програма виводить на екран тип зв'язності графу.

Код:

```
#include <iostream>
#include <windows.h>
#include <random>
#include <string>
using namespace std;

float n, m;
int a[20][20], sum[20][20], r[20][20], r1[20][20], temp[20][20], temp1[20][20],
vidst[20][20], dos[20][20], i_matr[20][20];
int d = 1;
bool varik = TRUE;
string str;
char t;

void generate() {
    random_device random_device; // entropy sourse
    mt19937 generator(random_device()); // initialise randimizer
```

```
    cout << "Вершин: ";
    cin >> n;
    if (n <= 0 || (n - int(n)) != 0) {
        cout << "\n---------------------------\nN - не удовлетворяет условие\n----------------------
------\n";
        exit(0);
    }
    uniform_int_distribution<> distribution(1, n);
    cout << "Ребер: ";
    cin >> m;
    if (m <= 0 || (m - int(m)) != 0) {
        cout << "\n---------------------------\nM - не удовлетворяет условие\n---------------------
-------\n";
        exit(0);
    }
    cout << "\n";
    // define work type
    cout << "Сгенерирровать граф или задать вручную (0/1): ";
    cin >> t;
    switch (t) {
    case '0':
        t = '0';
        break;
    case '1':
        cout << "\n\nСПИСОК РЕБЕР:";
        cout << "\n-------------\n";
        for (int i = 0; i < m; i++) {
            if (i + 1 > 9) {
                cout << "e" << i + 1 << " = ";
            }
            else {
```

```
            cout << "e" << i + 1 << "  = ";
        }
        for (int j = 0; j < 2; j++) {
            cin >> a[i][j];
            if (a[i][j] >= 1 && a[i][j] <= n) {
                continue;
            }
            else {
                cout << "\n-----------------------------\nV" << a[i][j] << " - НЕ является точкой графа\n-----------------------------\n";
                exit(0);
            }
        }
    }
    cout << "\n\n";
    break;
  default:
    cerr << "\n-----------------------------------------------\n";
    cerr << "Неправильный тип работы программы, введите:\n0 - чтоб сгенерировать граф\n1 - чтоб задать его вручную";
    cerr << "\n-----------------------------------------------\n";
    exit(0);
  }
  if (t == '0') {
    // generate
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < 2; j++) {
            a[i][j] = distribution(generator);
        }
    }
  }
```

```
}
void output() {
    if (t == '0') {
        cout << "\nСПИСОК РЕБЕР:";
        cout << "\n----------------\n";
        for (int i = 0; i < m; i++) {
            str = "   ";
            // 10-99 numbs
            if (i >= 9) {
                cout << "e" << i + 1 << " = (";
            }
            // 1-9 numbs
            else {
                cout << "e" << i + 1 << "  = (";
            }
            for (int j = 0; j < 2; j++) {
                if (j < 1) {
                    // 10-99 numbs
                    if (((a[i][j] / 10) != 0) && (a[i][j] / 10) <= 9) {
                        str = "  ";
                    }
                    // 100-999 numbs
                    else if (((a[i][j] / 10) != 0) && (a[i][j] / 10) > 9 && (a[i][j] / 10) <= 99) {
                        str = " ";
                    }
                    cout << "v" << a[i][j] << "," << str;
                }
                else {
                    cout << "v" << a[i][j] << ")\n";
                }
            }
```

```
        }
        cout << "\n\n";
    }
}
void sumizhna_matr() {
    cout << "СМЕЖНАЯ МАТРИЦА:\n";
    for (int i = 0; i < m; i++) {
        sum[a[i][0]][a[i][1]] = 1;
    }
    // print
    for (int b = 0; b < n; b++) {
        if (b == 0) {
            cout << "----------";
            cout << "----------";
        }
        else {
            cout << "----------";
        }
    }
    cout << "\n";
    for (int s = 0; s < n; s++) {
        // first interpr
        if (s == 0) {
            // names of tables
            for (int f = 0; f < n; f++) {
                // first interpr
                if (f == 0) {
                    cout << "         |";
                }
                // 10-99 numbs
                if (f >= 9) {
```

```
            cout << "    v" << f + 1 << "  |";
        }
        // 1-9 numbs
        else {
            cout << "    v" << f + 1 << "   |";
        }
    }
    cout << "\n";
}
// skelet of the tabl
for (int l = 0; l < n; l++) {
    if (l == 0) {
        cout << "----------";
        cout << "----------";
    }
    else {
        cout << "----------";
    }
}
cout << "\n";
for (int k = 0; k < n; k++) {
    // first interpr
    if (k == 0) {
        // 10-99 numbs
        if (s + 1 > 9) {
            cout << "   v" << s + 1 << "   |";
        }
        // 1-9 numbs
        else {
            cout << "    v" << s + 1 << "   |";
        }
```

```
            }
            cout << "    " << sum[s + 1][k + 1] << "    |";
        }
        cout << "\n";
    }
}
void vidst_matr() {
    // define vidst from first degree
    for (int p = 1; p <= n; p++) {
        for (int q = 1; q <= n; q++) {
            vidst[p][q] = sum[p][q];
        }
    }
    // (r = sum), (temp = sum)
    for (int b = 1; b <= n; b++) {
        for (int s = 1; s <= n; s++) {
            r[b][s] = sum[b][s];
        }
    }
    for (int i = 2; i <= n; i++) {
        int sum_ch = 0;
        // temp = r * sum
        for (int b = 1; b <= n; b++) {
            for (int s = 1; s <= n; s++) {
                for (int k = 1; k <= n; k++) {
                    sum_ch += sum[b][k] * r[k][s];
                }
                temp[b][s] = sum_ch;
                sum_ch = 0;
            }
        }
```

```
    // r = temp
    for (int b = 1; b <= n; b++) {
        for (int s = 1; s <= n; s++) {
            r[b][s] = temp[b][s];
        }
    }
    // print temp
    cout << "\n\nСМЕЖНАЯ МАТРИЦА^"<<i<<":\n";
    for (int c = 1; c <= n; c++) {
        for (int l = 1; l <= n; l++) {
            cout << r[c][l] << "\t";
            if (c != l && r[c][l] == 1 && vidst[c][l] == 0) {
                vidst[c][l] = i;
            }
        }
        cout << "\n";
    }
}
// print final
cout << "\n\nМАТРИЦА РАССТОЯНИЙ:\n";
for (int z = 0; z < n; z++) {
    if (z == 0) {
        cout << "----------";
        cout << "----------";
    }
    else {
        cout << "----------";
    }
}
cout << "\n";
for (int x = 0; x < n; x++) {
```

```
// first interpr
if (x == 0) {
    // names of tables
    for (int f = 0; f < n; f++) {
        // first interpr
        if (f == 0) {
            cout << "        |";
        }
        // 10-99 numbs
        if (f >= 9) {
            cout << "    v" << f + 1 << "  |";
        }
        // 1-9 numbs
        else {
            cout << "    v" << f + 1 << "   |";
        }
    }
    cout << "\n";
}
// skelet of the tabl
for (int c = 0; c < n; c++) {
    if (c == 0) {
        cout << "----------";
        cout << "----------";
    }
    else {
        cout << "----------";
    }
}
cout << "\n";
for (int y = 0; y < n; y++) {
```

```
            // first interpr
            if (y == 0) {
                // 10-99 numbs
                if (x + 1 > 9) {
                    cout << "   v" << x + 1 << "   |";
                }
                // 1-9 numbs
                else {
                    cout << "    v" << x + 1 << "   |";
                }
            }
            if (vidst[x + 1][y + 1] == 0 && x + 1 != y + 1) {
                cout << "    *    |";
            }
            else {
                cout << "    " << vidst[x + 1][y + 1] << "    |";
            }
        }
        cout << "\n";
    }
}
void dosyjnist_matr() {
    // to create matrix with "1" on the main diagonal
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n; j++) {
            if (i == j) {
                i_matr[i][j] = 1;
            }
            else {
                i_matr[i][j] = 0;
            }
```

```
    }
}
// sum (i_matr + sum)
for (int i = 1; i <= n; i++) {
    for (int j = 1; j <= n; j++) {
        dos[i][j] = i_matr[i][j] + sum[i][j];
    }
}
// print step 1
cout << "\n\nЕДИНИЧНАЯ + СМЕЖНАЯ МАТРИЦА:\n";
for (int i = 1; i <= n; i++) {
    for (int j = 1; j <= n; j++) {
        cout<<dos[i][j]<<"\t";
    }
    cout << "\n";
}
// (r1 = dos), (temp1 = dos)
for (int b = 1; b <= n; b++) {
    for (int s = 1; s <= n; s++) {
        r1[b][s] = dos[b][s];
    }
}
for (int i = 2; i <= n - 1; i++) {
    int sum_ch = 0;
    // temp1 = r1 * dos
    for (int b = 1; b <= n; b++) {
        for (int s = 1; s <= n; s++) {
            for (int k = 1; k <= n; k++) {
                sum_ch += dos[b][k] * r1[k][s];
            }
            temp1[b][s] = sum_ch;
```

```
            sum_ch = 0;
        }
    }
    // r1 = temp1
    for (int b = 1; b <= n; b++) {
        for (int s = 1; s <= n; s++) {
            r1[b][s] = temp1[b][s];
            // convert to 1
            if (r1[b][s] != 0) {
                dos[b][s] = 1;
            }
        }
    }
}
// print step 2
cout << "\n\n(ЕДИНИЧНАЯ + СМЕЖНАЯ МАТРИЦА)^"<<n-1<<":\n";
for (int c = 1; c <= n; c++) {
    for (int l = 1; l <= n; l++) {
        cout << r1[c][l] << "\t";
    }
    cout << "\n";
}
// print final
cout << "\n\nМАТРИЦА ДОСЯГАЕМОСТИ:\n";
for (int z = 0; z < n; z++) {
    if (z == 0) {
        cout << "----------";
        cout << "----------";
    }
    else {
        cout << "----------";
```

```
    }
}
cout << "\n";
for (int x = 0; x < n; x++) {
    // first interpr
    if (x == 0) {
        // names of tables
        for (int f = 0; f < n; f++) {
            // first interpr
            if (f == 0) {
                cout << "         |";
            }
            // 10-99 numbs
            if (f >= 9) {
                cout << "    v" << f + 1 << "  |";
            }
            // 1-9 numbs
            else {
                cout << "    v" << f + 1 << "   |";
            }
        }
        cout << "\n";
    }
    // skelet of the tabl
    for (int c = 0; c < n; c++) {
        if (c == 0) {
            cout << "----------";
            cout << "----------";
        }
        else {
            cout << "----------";
```

```
            }
        }
        cout << "\n";
        for (int y = 0; y < n; y++) {
            // first interpr
            if (y == 0) {
                // 10-99 numbs
                if (x + 1 > 9) {
                    cout << "   v" << x + 1 << "   |";
                }
                // 1-9 numbs
                else {
                    cout << "    v" << x + 1 << "   |";
                }
            }
            cout << "    " << dos[x + 1][y + 1] << "    |";
        }
        cout << "\n";
    }
}
void defType() {
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n; j++) {
            if (dos[i][j] != 1) {
                varik = FALSE;
            }
        }
    }
    if (varik == TRUE) {
        cout << "\n\nГРАФ - СИЛЬНО-СВЯЗАННЫЙ\n";
    }
```

```
else {
    varik = TRUE;
    // sum (transpon matr dosygnosti + matr dos)
    for (int x = 1; x <= n; x++) {
        for (int y = 1; y <= n; y++) {
            temp[x][y] = dos[x][y] + dos[y][x];
            if (temp[x][y] == 0) {
                varik = FALSE;
            }
        }
    }
    if (varik == TRUE) {
        cout << "\n\nГРАФ - ОДНОБОЧНО-СВЯЗАННЫЙ\n";
    }
    else {
        varik = TRUE;
        // sum (transpon matr sumizhnosti + matr sum + I)
        for (int a = 1; a <= n; a++) {
            for (int b = 1; b <= n; b++) {
                temp1[a][b] = sum[a][b] + sum[b][a] + i_matr[a][b];
            }
        }
        // (r1 = sum), (temp1 = sum)
        for (int s = 1; s <= n; s++) {
            for (int k = 1; k <= n; k++) {
                r1[s][k] = temp1[s][k];
            }
        }
        for (int i = 2; i <= n; i++) {
            int sum_ch = 0;
            // temp1 = r1 * sum
```

```
            for (int b = 1; b <= n; b++) {
                for (int s = 1; s <= n; s++) {
                    for (int k = 1; k <= n; k++) {
                        sum_ch += temp1[b][k] * r1[k][s];
                    }
                    temp1[b][s] = sum_ch;
                    sum_ch = 0;
                }
            }
            // r1 = temp1
            for (int b = 1; b <= n; b++) {
                for (int s = 1; s <= n; s++) {
                    r1[b][s] = temp1[b][s];
                }
            }
        }
        for (int b = 1; b <= n; b++) {
            for (int s = 1; s <= n; s++) {
                if (temp1[b][s] == 0) {
                    varik = FALSE;
                }
            }
        }
        if (varik == TRUE) {
            cout << "\n\nГРАФ - СЛАБО-СВЯЗАННЫЙ\n";
        }
        else {
            cout << "\n\nГРАФ - НЕСВЯЗАННЫЙ\n";
        }
    }
}
```

```
}
int main() {
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);

    generate();
    output();
    sumizhna_matr();
    vidst_matr();
    dosyjnist_matr();
    defType();
}
```

Результат:

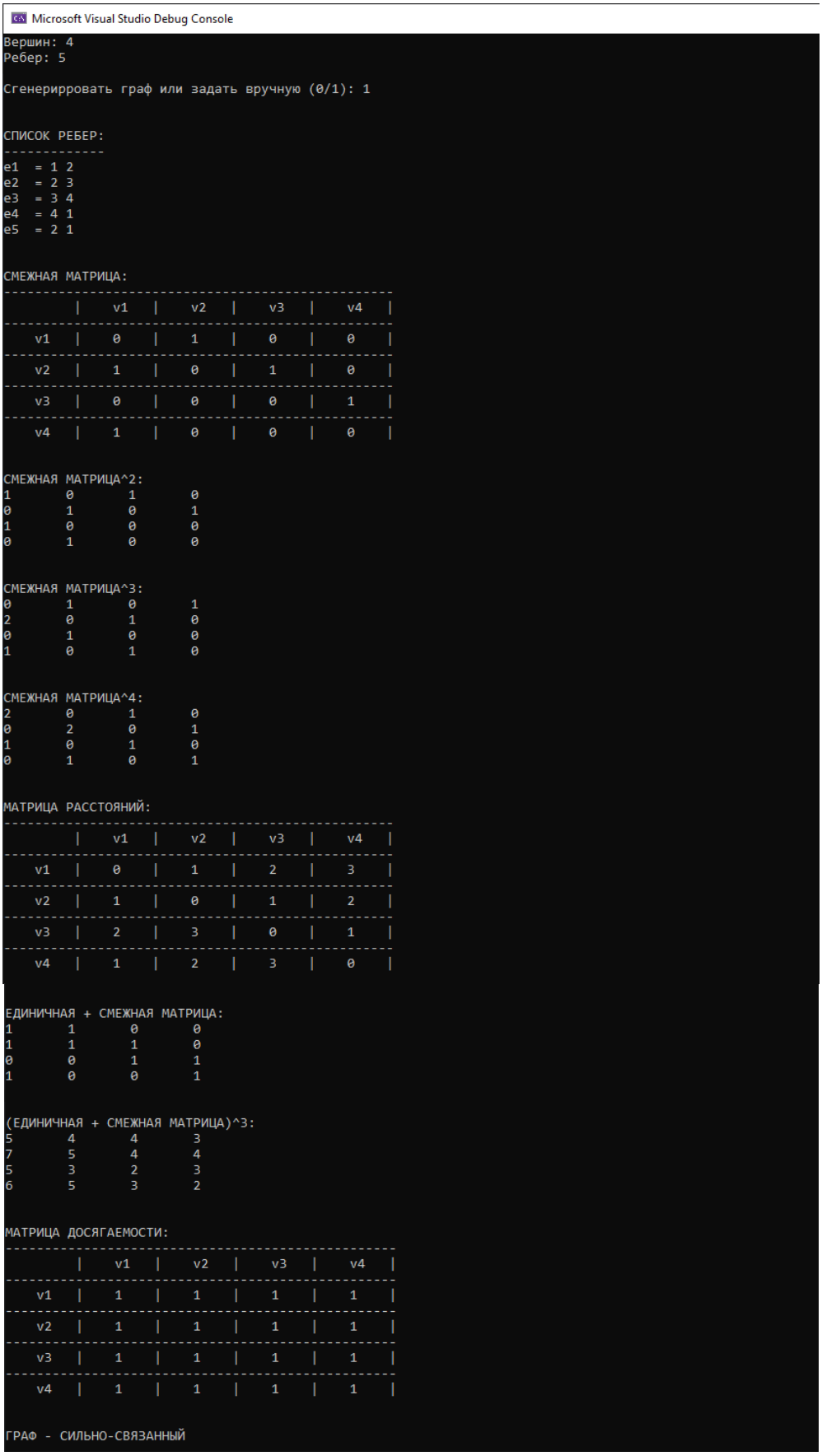

Рис. 1. Результат виконання програми (пункт 1, 3) для сильно-зв’язного графа

Microsoft Visual Studio Debug Console

```
Вершин: 6
Ребер: 6

Сгенеирровать граф или задать вручную (0/1): 1


СПИСОК РЕБЕР:
------------
e1  = 6 1
e2  = 1 5
e3  = 1 4
e4  = 5 4
e5  = 3 4
e6  = 2 3


СМЕЖНАЯ МАТРИЦА:
-----------------------------------------------------------------
        |   v1   |   v2   |   v3   |   v4   |   v5   |   v6   |
-----------------------------------------------------------------
   v1   |   0    |   0    |   0    |   1    |   1    |   0    |
-----------------------------------------------------------------
   v2   |   0    |   0    |   1    |   0    |   0    |   0    |
-----------------------------------------------------------------
   v3   |   0    |   0    |   0    |   1    |   0    |   0    |
-----------------------------------------------------------------
   v4   |   0    |   0    |   0    |   0    |   0    |   0    |
-----------------------------------------------------------------
   v5   |   0    |   0    |   0    |   1    |   0    |   0    |
-----------------------------------------------------------------
   v6   |   1    |   0    |   0    |   0    |   0    |   0    |


СМЕЖНАЯ МАТРИЦА^2:
0       0       0       1       0       0
0       0       0       1       0       0
0       0       0       0       0       0
0       0       0       0       0       0
0       0       0       0       0       0
0       0       0       1       1       0


СМЕЖНАЯ МАТРИЦА^3:
0       0       0       0       0       0
0       0       0       0       0       0
0       0       0       0       0       0
0       0       0       0       0       0
0       0       0       0       0       0
0       0       0       1       0       0


СМЕЖНАЯ МАТРИЦА^4:
0       0       0       0       0       0
0       0       0       0       0       0
0       0       0       0       0       0
0       0       0       0       0       0
0       0       0       0       0       0
0       0       0       0       0       0
```

```
СМЕЖНАЯ МАТРИЦА^5:
0       0       0       0       0       0
0       0       0       0       0       0
0       0       0       0       0       0
0       0       0       0       0       0
0       0       0       0       0       0
0       0       0       0       0       0

СМЕЖНАЯ МАТРИЦА^6:
0       0       0       0       0       0
0       0       0       0       0       0
0       0       0       0       0       0
0       0       0       0       0       0
0       0       0       0       0       0
0       0       0       0       0       0

МАТРИЦА РАССТОЯНИЙ:
-------------------------------------------------------------------
        |   v1  |   v2  |   v3  |   v4  |   v5  |   v6  |
-------------------------------------------------------------------
   v1   |   0   |   *   |   *   |   1   |   1   |   *   |
-------------------------------------------------------------------
   v2   |   *   |   0   |   1   |   2   |   *   |   *   |
-------------------------------------------------------------------
   v3   |   *   |   *   |   0   |   1   |   *   |   *   |
-------------------------------------------------------------------
   v4   |   *   |   *   |   *   |   0   |   *   |   *   |
-------------------------------------------------------------------
   v5   |   *   |   *   |   *   |   1   |   0   |   *   |
-------------------------------------------------------------------
   v6   |   1   |   *   |   *   |   2   |   2   |   0   |

ЕДИНИЧНАЯ + СМЕЖНАЯ МАТРИЦА:
1       0       0       1       1       0
0       1       1       0       0       0
0       0       1       1       0       0
0       0       0       1       0       0
0       0       0       1       1       0
1       0       0       0       0       1

(ЕДИНИЧНАЯ + СМЕЖНАЯ МАТРИЦА)^5:
1       0       0       15      5       0
0       1       5       13      0       0
0       0       1       5       0       0
0       0       0       1       0       0
0       0       0       5       1       0
5       0       0       32      13      1

МАТРИЦА ДОСЯГАЕМОСТИ:
-------------------------------------------------------------------
        |   v1  |   v2  |   v3  |   v4  |   v5  |   v6  |
-------------------------------------------------------------------
   v1   |   1   |   0   |   0   |   1   |   1   |   0   |
-------------------------------------------------------------------
   v2   |   0   |   1   |   1   |   1   |   0   |   0   |
-------------------------------------------------------------------
   v3   |   0   |   0   |   1   |   1   |   0   |   0   |
-------------------------------------------------------------------
   v4   |   0   |   0   |   0   |   1   |   0   |   0   |
-------------------------------------------------------------------
   v5   |   0   |   0   |   0   |   1   |   1   |   0   |
-------------------------------------------------------------------
   v6   |   1   |   0   |   0   |   1   |   1   |   1   |

ГРАФ - СЛАБО-СВЯЗАННЫЙ
```

Рис. 2. Результат виконання програми (пункт 1, 3) для слабко-зв’язного графа

Microsoft Visual Studio Debug Console

```
Вершин: 6
Ребер: 9

Сгенерирровать граф или задать вручную (0/1): 1


СПИСОК РЕБЕР:
-------------
e1  = 1 6
e2  = 6 1
e3  = 2 1
e4  = 2 3
e5  = 3 4
e6  = 5 4
e7  = 4 6
e8  = 1 5
e9  = 5 6


СМЕЖНАЯ МАТРИЦА:
-------------------------------------------------------------------------
        |    v1   |    v2   |    v3   |    v4   |    v5   |    v6   |
-------------------------------------------------------------------------
    v1  |    0    |    0    |    0    |    0    |    1    |    1    |
-------------------------------------------------------------------------
    v2  |    1    |    0    |    1    |    0    |    0    |    0    |
-------------------------------------------------------------------------
    v3  |    0    |    0    |    0    |    1    |    0    |    0    |
-------------------------------------------------------------------------
    v4  |    0    |    0    |    0    |    0    |    0    |    1    |
-------------------------------------------------------------------------
    v5  |    0    |    0    |    0    |    1    |    0    |    1    |
-------------------------------------------------------------------------
    v6  |    1    |    0    |    0    |    0    |    0    |    0    |


СМЕЖНАЯ МАТРИЦА^2:
1       0       0       1       0       1
0       0       0       1       1       1
0       0       0       0       0       1
1       0       0       0       0       0
1       0       0       0       0       1
0       0       0       0       1       1


СМЕЖНАЯ МАТРИЦА^3:
1       0       0       0       1       2
1       0       0       1       0       2
1       0       0       0       0       0
0       0       0       0       1       1
1       0       0       0       1       1
1       0       0       1       0       1


СМЕЖНАЯ МАТРИЦА^4:
2       0       0       1       1       2
2       0       0       0       1       2
0       0       0       0       1       1
1       0       0       1       0       1
1       0       0       1       1       2
1       0       0       0       1       2
```

```
СМЕЖНАЯ МАТРИЦА^5:
2       0       0       1       2       4
2       0       0       1       2       3
1       0       0       1       0       1
1       0       0       0       1       2
2       0       0       1       1       3
2       0       0       1       1       2


СМЕЖНАЯ МАТРИЦА^6:
4       0       0       2       2       5
3       0       0       2       2       5
1       0       0       0       1       2
2       0       0       1       1       2
3       0       0       1       2       4
2       0       0       1       2       4


МАТРИЦА РАССТОЯНИЙ:
-----------------------------------------------------------------------
        |    v1   |    v2   |    v3   |    v4   |    v5   |    v6   |
-----------------------------------------------------------------------
    v1  |    0    |    *    |    *    |    2    |    1    |    1    |
-----------------------------------------------------------------------
    v2  |    1    |    0    |    1    |    2    |    2    |    2    |
-----------------------------------------------------------------------
    v3  |    3    |    *    |    0    |    1    |    4    |    2    |
-----------------------------------------------------------------------
    v4  |    2    |    *    |    *    |    0    |    3    |    1    |
-----------------------------------------------------------------------
    v5  |    2    |    *    |    *    |    1    |    0    |    1    |
-----------------------------------------------------------------------
    v6  |    1    |    *    |    *    |    3    |    2    |    0    |


ЕДИНИЧНАЯ + СМЕЖНАЯ МАТРИЦА:
1       0       0       0       1       1
1       1       1       0       0       0
0       0       1       1       0       0
0       0       0       1       0       1
0       0       0       1       1       1
1       0       0       0       0       1


(ЕДИНИЧНАЯ + СМЕЖНАЯ МАТРИЦА)^5:
88      0       0       52      60      140
146     1       5       91      97      232
113     0       1       68      76      180
88      0       0       52      60      140
88      0       0       52      60      140
88      0       0       52      60      140


МАТРИЦА ДОСЯГАЕМОСТИ:
-----------------------------------------------------------------------
        |    v1   |    v2   |    v3   |    v4   |    v5   |    v6   |
-----------------------------------------------------------------------
    v1  |    1    |    0    |    0    |    1    |    1    |    1    |
-----------------------------------------------------------------------
    v2  |    1    |    1    |    1    |    1    |    1    |    1    |
-----------------------------------------------------------------------
    v3  |    1    |    0    |    1    |    1    |    1    |    1    |
-----------------------------------------------------------------------
    v4  |    1    |    0    |    0    |    1    |    1    |    1    |
-----------------------------------------------------------------------
    v5  |    1    |    0    |    0    |    1    |    1    |    1    |
-----------------------------------------------------------------------
    v6  |    1    |    0    |    0    |    1    |    1    |    1    |


ГРАФ - ОДНОБОЧНО-СВЯЗАННЫЙ
```

Рис. 3. Результат виконання програми (пункт 1, 3) для однобічно-зв’язного графа

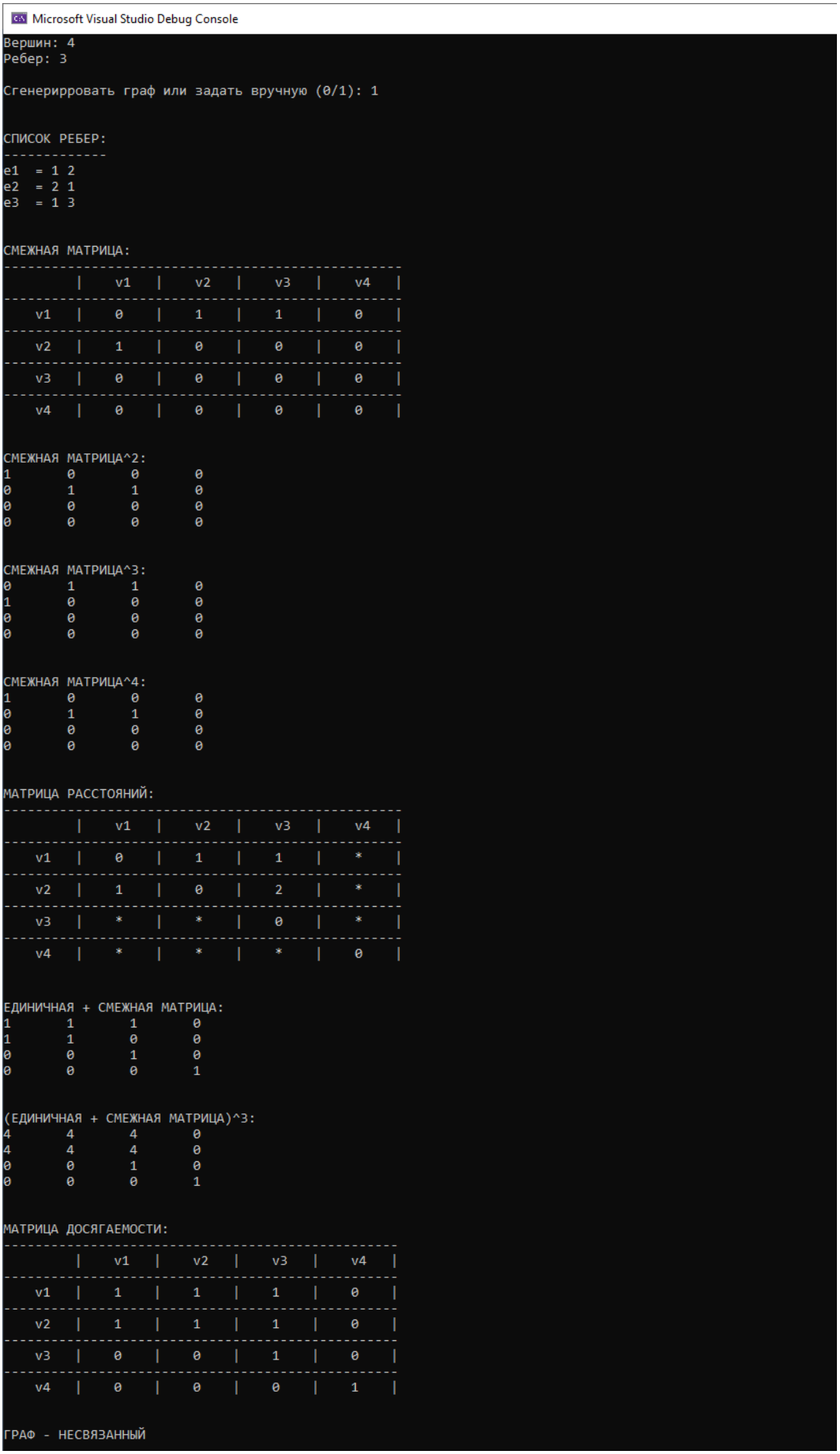

Рис. 4. Результат виконання програми (пункт 1, 3) для незв’язного графа

Для виконання пункту 2 завдання, було взято за основу однобічно-зв’язний граф, характеристики якого зазначено на рис. 2. На рисунку 5 зображено виконання пункту 2 лабораторної роботи №3 вручну.

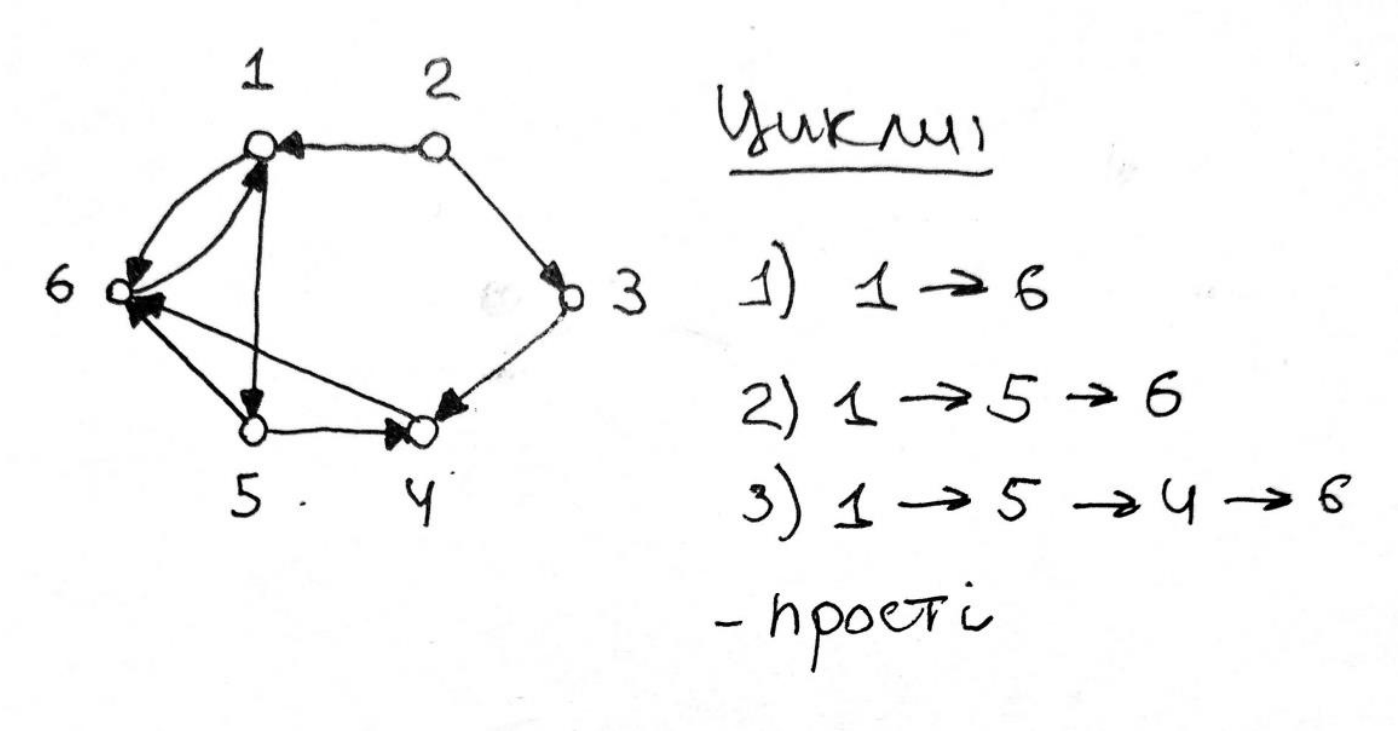

Рис. 5. Результат виконання пункту 2 завдання

ВИСНОВКИ

У ході виконання лабораторної роботи було розглянуто та вивчено різні типи зв’язності графів та їх представлення в пам’яті обчислювальних пристроїв (комп’ютерів).

Створено програмне забезпечення на мові програмування C++ для роботи з графами та визначення їх характеристик та типу зв’язності. Реалізована програма приймає від користувача граф, заданий списком ребер вручну: користувачем з клавіатури, або за допомогою рандомайзеру і будує для заданого графа матрицю суміжності, матрицю відстаней, матрицю досяжностей та визначає тип зв’язності графа. Пункт 2 лабораторної роботи №3 виконано вручну, без застосування програмного забезпечення.