

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 № 1
«Представлення 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

ЗМІСТ

1. ВСТУП.....	3
2. ХІД РОБОТИ.....	4
2.1. Завдання.....	4
2.2. Алгоритм.....	4
2.3. Код.....	5
2.4. Результат.....	10
3. ВИСНОВКИ.....	11

ВСТУП

Тема: Представлення графів.

Мета: Розглянути та вивчити різні типи представлення графів в пам'яті обчислювальних пристроїв (комп'ютерів).

Обладнання: Персональні комп'ютери.

ХІД РОБОТИ

Завдання:

Реалізувати програмне застосування (програму), яке виконує наступні функції.

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

```
n m
v1 u1
v2 u2
. . . .
vm um
```

Тут n – кількість вершин графу (ціле число, більше нуля), m – кількість ребер графу (ціле число, більше нуля), v_i та u_i – початкова та кінцева вершина ребра i ($1 \leq v_i \leq n$, $1 \leq u_i \leq n$, цілі числа). Індксація вершин у файлі ведеться з 1. Вважається, що граф є орієнтованим.

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

2. Вивід матриць інцидентності та суміжності. За вимогою користувача програма повинна виводити матриці інцидентності та суміжності (окремі функції) на екран та/або у текстовий файл, який вказує користувач.

Алгоритм:

1. Отримати від користувача значення кількості вершин і ребер графа.
2. Обрати тип роботи програми: задавати граф рандомом чи вручну.
3. Задати граф списком ребер:
 - 3.1. Якщо тип роботи = «0», задати граф рандомом:
 - 3.1.1. Задати двовимірний масив рандомними значеннями в діапазоні від 1 до кількості вершин графу.
 - 3.2. Якщо тип роботи = «1», задати граф вручну:
 - 3.2.1. Вводити кожний елемент з клавіатури.
 - 3.3. Вивести список ребер графа.
4. Одержати матрицю інцидентності.
 - 3.1. Створити двовимірний масив для матриці інцидентності.
 - 3.2. У кожному рядку масиву списку ребер:
 - 3.2.1. Перевірити, чи елементи ряду співпадають.
 - 3.2.1.1. Якщо так, то присвоїти елементу матриці інцидентності [у рядку елемента матриці списку ребер – перший елемент] [відповідний номер ряду матриці списку ребер] = 2

3.2.1.2. Якщо ні, то присвоїти елементу матриці інцидентності [у рядку елемента матриці списку ребер – перший елемент] [відповідний номер ряду матриці списку ребер] = -1. Елементу матриці інцидентності [у рядку елемента матриці списку ребер – другий елемент] [відповідний номер ряду матриці списку ребер] = 1.

3.3. Вивести матрицю інцидентності.

5. Одержати матрицю суміжності.

4.1. Створити двовимірний масив для матриці суміжності.

4.2. У кожному рядку масиву списку ребер:

4.2.1. Кожному елементу матриці суміжності [у рядку елемента матриці списку ребер – перший елемент] [у рядку елемента матриці списку ребер – другий елемент] = 1.

4.3. Вивести матрицю суміжності.

Код:

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

float n, m;
int a[20][20], inz[20][20], sum[20][20];
string str;
char t;

void generate() {
    random_device random_device; // entropy source
    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 условию\n-----\n";
    exit(0);
}
cout << "\n";
// define work type
cout << "Сгенерировать 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++) {
            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 << "СПИСОК 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 inz_matr() {
    cout << "ИНЦИДЕНТНАЯ МАТРИЦА:\n";
    cout << "-----";
    for (int i = 0; i < m; i++) {
        cout << "-----";
        if (a[i][0] != a[i][1]) {
            inz[a[i][0]][i] = -1;
            inz[a[i][1]][i] = 1;
        }
    }
}

```

```

    else {
        inz[a[i][0]][i] = 2;
    }
}
cout << "\n";
// print inz matr
for (int t = 0; t < m; t++) {
    // first interpr
    if (t == 0) {
        cout << "    \|t";
    }
    // 10-99 numbs
    if (t + 1 > 9) {
        cout << "e" << t + 1 << " \|t";
    }
    // 1-9 numbs
    else {
        cout << "e" << t + 1 << " \|t";
    }
}
cout << "\n";
for (int s = 0; s < n; s++) {
    // skelet of the tabl
    for (int l = 0; l < m; l++) {
        if (l == 0) {
            cout << "-----";
            cout << "-----";
        }
        else {
            cout << "-----";
        }
    }
    cout << "\n";
    for (int k = 0; k < m; k++) {
        // first interpr
        if (k == 0) {
            // 10-99 numbs
            if (s + 1 > 9) {
                cout << " v" << s + 1 << " \|t";
            }
            // 1-9 numbs
            else {
                cout << " v" << s + 1 << " \|t";
            }
        }
    }
    // 10-99 numbs

```



```

        if (inz[s + 1][k] >= 0) {
            cout << inz[s + 1][k] << "  \t";
        }
        // 1-9 numbs
    else {
        cout << inz[s + 1][k] << "  \t";
    }
}
cout << "\n";
}
cout << "\n\n";
}

void sum_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 << "  \t";
                }
                // 10-99 numbs
                if (f >= 9) {
                    cout << "v" << f + 1 << "  \t";
                }
                // 1-9 numbs
                else {
                    cout << "v" << f + 1 << "  \t";
                }
            }
        }
    }
}

```

```

        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 << " \t";
            }
            // 1-9 numbs
            else {
                cout << " v" << s + 1 << " \t";
            }
        }
        cout << sum[s+1][k+1] << " \t";
    }
    cout << "\n";
}
}
int main() {
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);

    generate();
    output();
    inz_matr();
    sum_matr();
}

```

Результат:



```
Microsoft Visual Studio Debug Console
Вершин: 0
-----
N - не удовлетворяет условие
-----
```

Рис. 1. Результат выполнения программы при неправильно введенном числу вершин



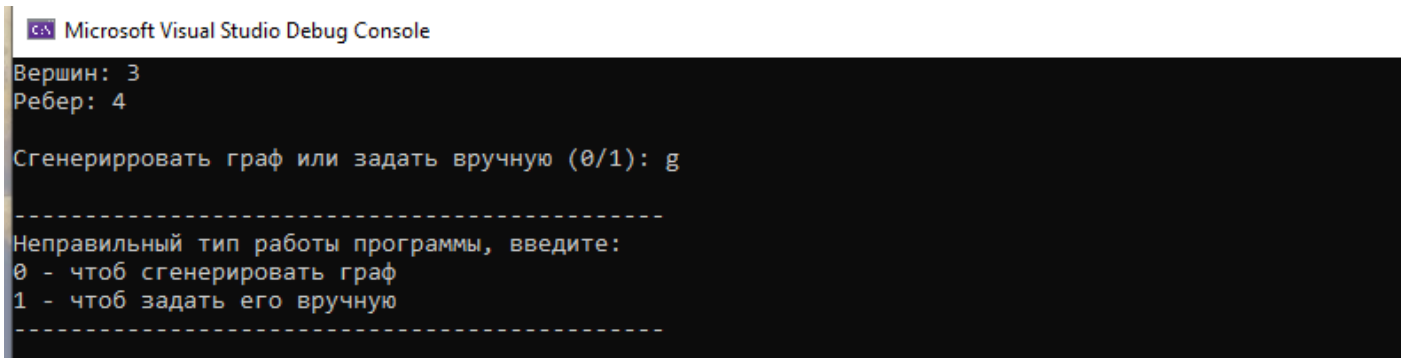
```
Microsoft Visual Studio Debug Console
Вершин: 3
Ребер: -3
-----
M - не удовлетворяет условие
-----
```

Рис. 2. Результат выполнения программы при неправильно введенном числу ребер



```
Microsoft Visual Studio Debug Console
Вершин: 3
Ребер: a
-----
M - не удовлетворяет условие
-----
```

Рис. 3. Результат выполнения программы при неправильно введенном значению числа ребер



```
Microsoft Visual Studio Debug Console
Вершин: 3
Ребер: 4
Сгенерировать граф или задать вручную (0/1): g
-----
Неправильный тип работы программы, введите:
0 - чтоб сгенерировать граф
1 - чтоб задать его вручную
-----
```

Рис. 4. Результат выполнения программы при неправильно введенном значению типу выполнения работы

```

Microsoft Visual Studio Debug Console

Вершин: 3
Ребер: 4

Сгенерировать граф или задать вручную (0/1): 0
СПИСОК РЕБЕР:
-----
e1 = (v3, v2)
e2 = (v2, v1)
e3 = (v3, v1)
e4 = (v1, v1)

ИНЦИДЕНТНАЯ МАТРИЦА:
-----
      | e1 | e2 | e3 | e4 |
-----
v1 | 0 | 1 | 1 | 2 |
-----
v2 | 1 | -1 | 0 | 0 |
-----
v3 | -1 | 0 | -1 | 0 |

СМЕЖНАЯ МАТРИЦА:
-----
      | v1 | v2 | v3 |
-----
v1 | 1 | 0 | 0 |
-----
v2 | 1 | 0 | 0 |
-----
v3 | 1 | 1 | 0 |

```

Рис. 5. Результат виконання програми при завданні графа рандомом

```

Microsoft Visual Studio Debug Console

Вершин: 3
Ребер: 4

Сгенерировать граф или задать вручную (0/1): 1
СПИСОК РЕБЕР:
-----
e1 = 3 2
e2 = 2 1
e3 = 3 1
e4 = 1 1

ИНЦИДЕНТНАЯ МАТРИЦА:
-----
      | e1 | e2 | e3 | e4 |
-----
v1 | 0 | 1 | 1 | 2 |
-----
v2 | 1 | -1 | 0 | 0 |
-----
v3 | -1 | 0 | -1 | 0 |

СМЕЖНАЯ МАТРИЦА:
-----
      | v1 | v2 | v3 |
-----
v1 | 1 | 0 | 0 |
-----
v2 | 1 | 0 | 0 |
-----
v3 | 1 | 1 | 0 |

```

Рис. 6. Результат виконання програми при завданні графа вручну

ВИСНОВКИ

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

Створено програмне забезпечення на мові програмування C++ для роботи з графами. Реалізована програма приймає від користувача граф, заданий списком ребер вручну: користувачем з клавіатури, або за допомогою рандомайзера і будує для заданого графа матриці інцидентності та суміжності.