MINISTERUL EDUCAŢIEI și CERCETĂRII al REPUBLICII MOLDOVA UNIVERSITATEA TEHNICĂ a MOLDOVEI   
FACULTATEA CALCULATOARE, INFORMATICǍ şi MICROELECTRONICǍ Departamentul Informatică și Ingineria Sistemelor

**Lucrare de laborator nr. 1**

La Matematica Discretă

Tema: ***Păstrarea grafurilor în memoria calculatorului.***

***Matricea drumurilor***

A realizat Chistol Maxim, st. gr. IA-233  
A verificat Vladimir Melnic

**Chișinău 2024**

**Cuprins**

[**1. Scopul lucrării** 3](#_Toc158886713)

[**2. Sarcina** 4](#_Toc158886714)

[**3. Concluzii** 13](#_Toc158886715)

# **1. Scopul lucrării**

1. Studierea metodelor de definire a unui graf: matrice de incidenţă, matrice de adiacenţă, liste;

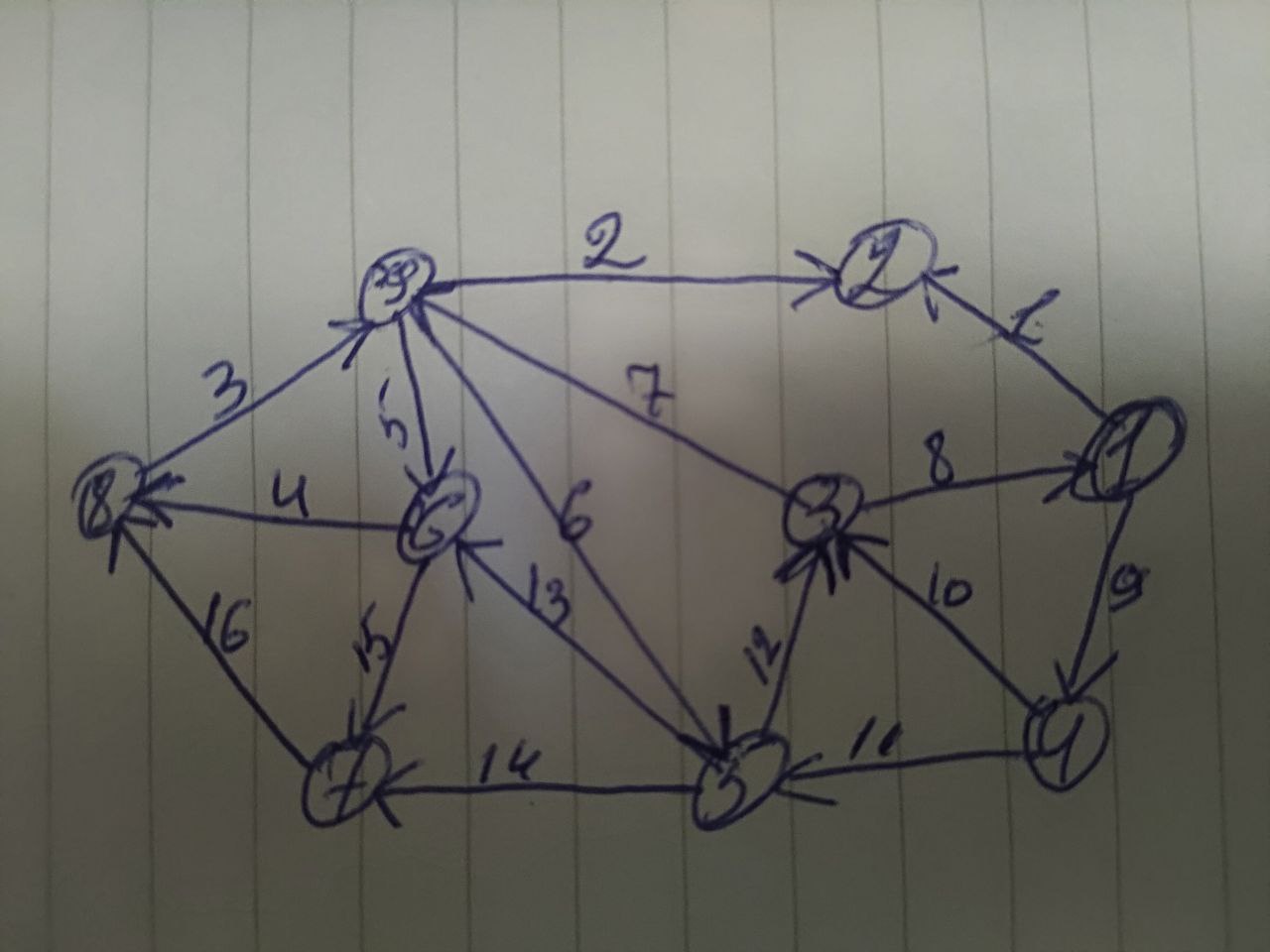
2. Elaborarea unor proceduri de introducere, extragere şi transformare a diferitor forme de reprezentare internă a grafurilor cu scoaterea rezultatelor la display şi imprimantă.

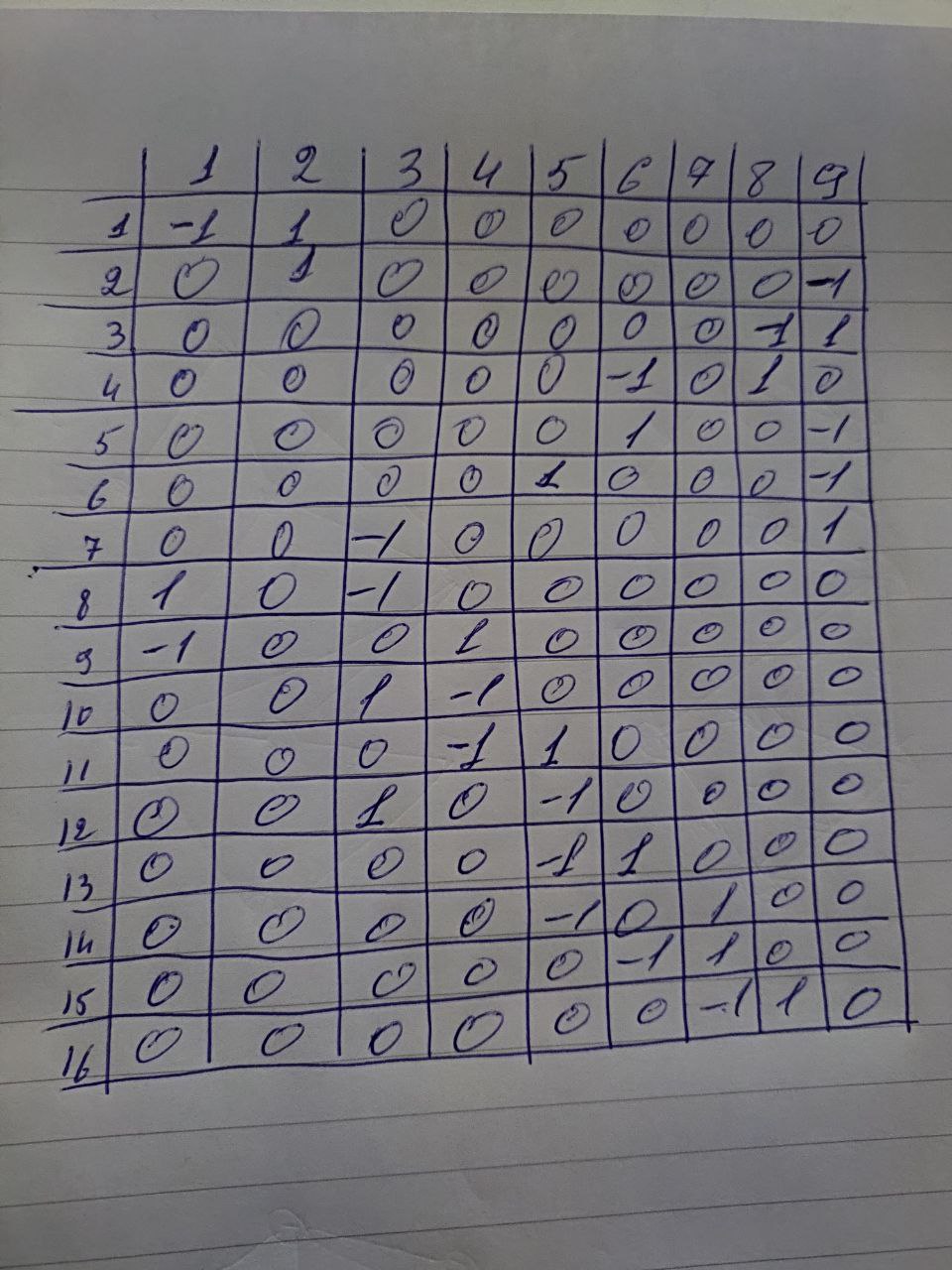
3. Elaborarea programelor de determinare a matricei drumurilor într-un graf orientat.

# **2. Sarcina**

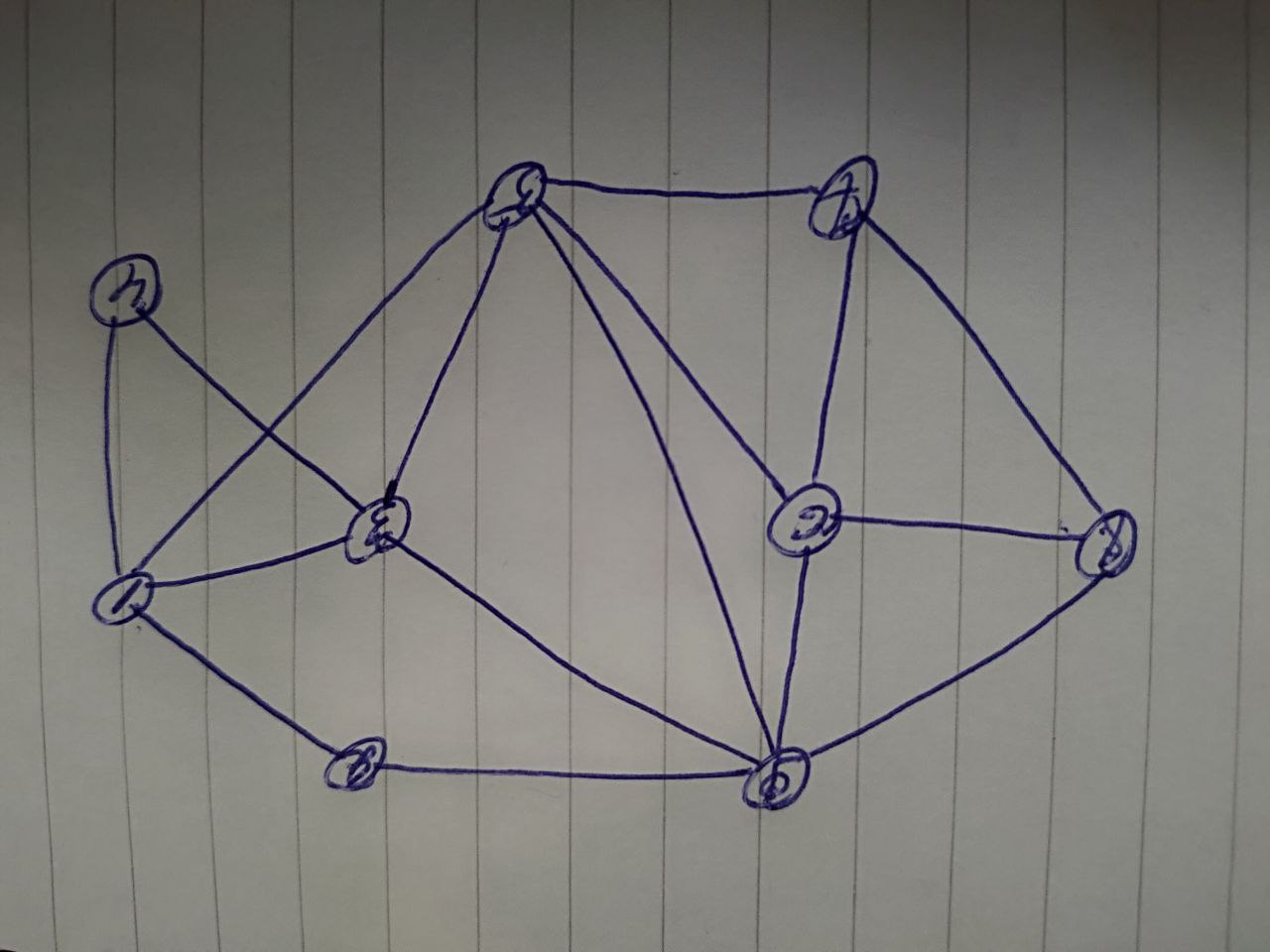
1. Elaboraţi procedura introducerii unui graf în memoria calculatorului în formă de:

• matrice de incidenţă (graf orientat)



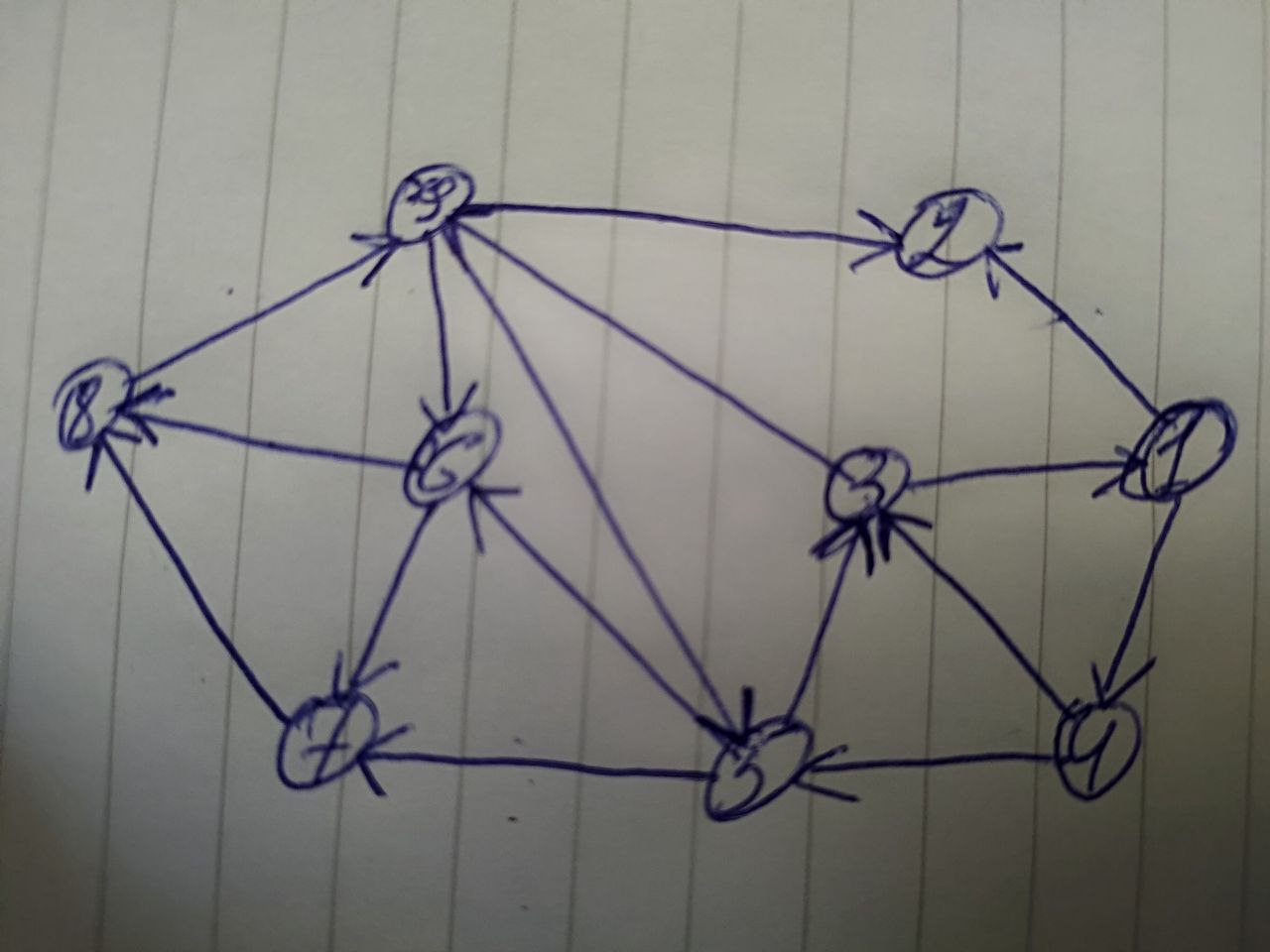


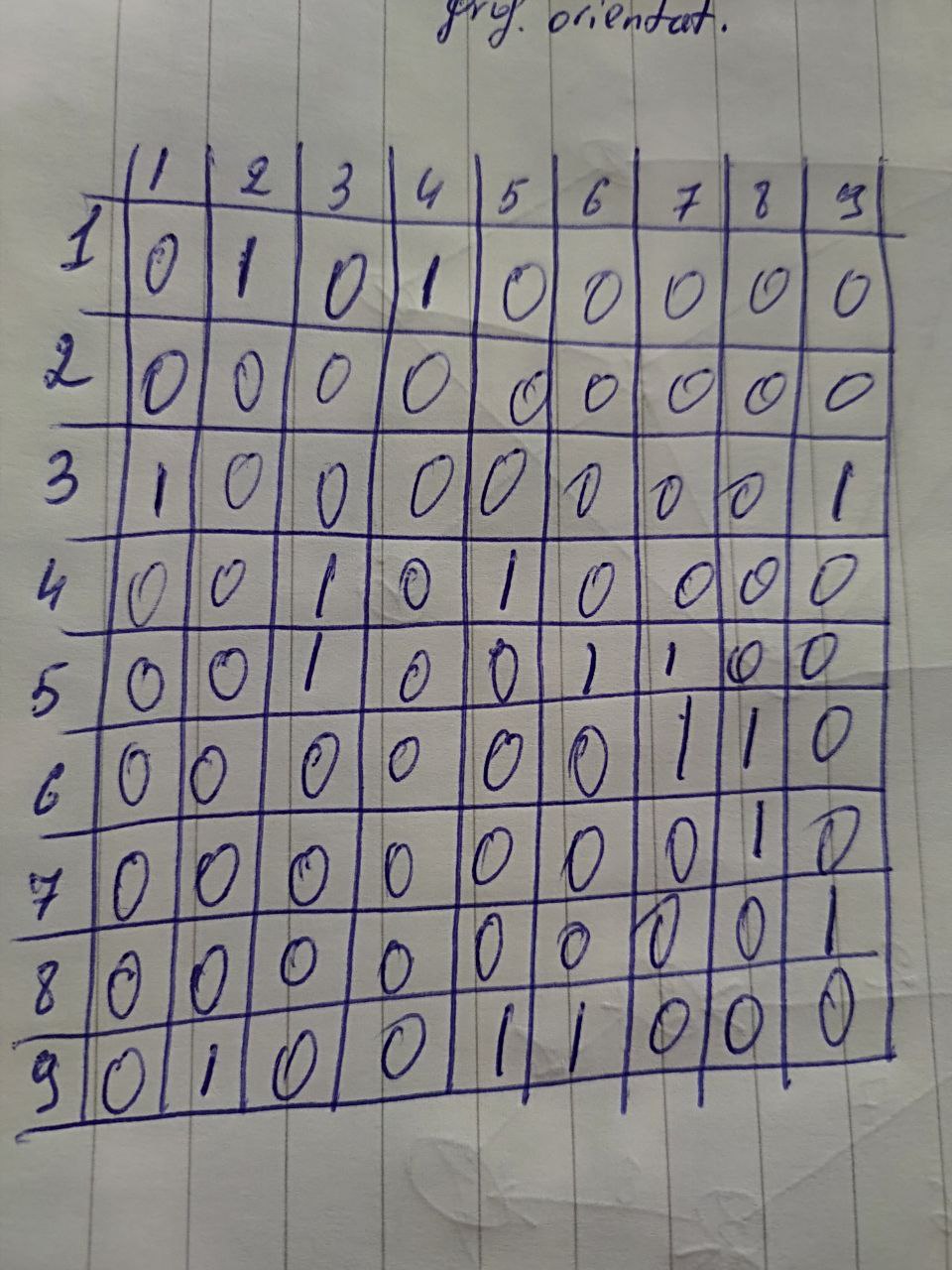
• matrice de adiacenţă (graf neorientat).





• matrice de adiacenţă (graf orientat).



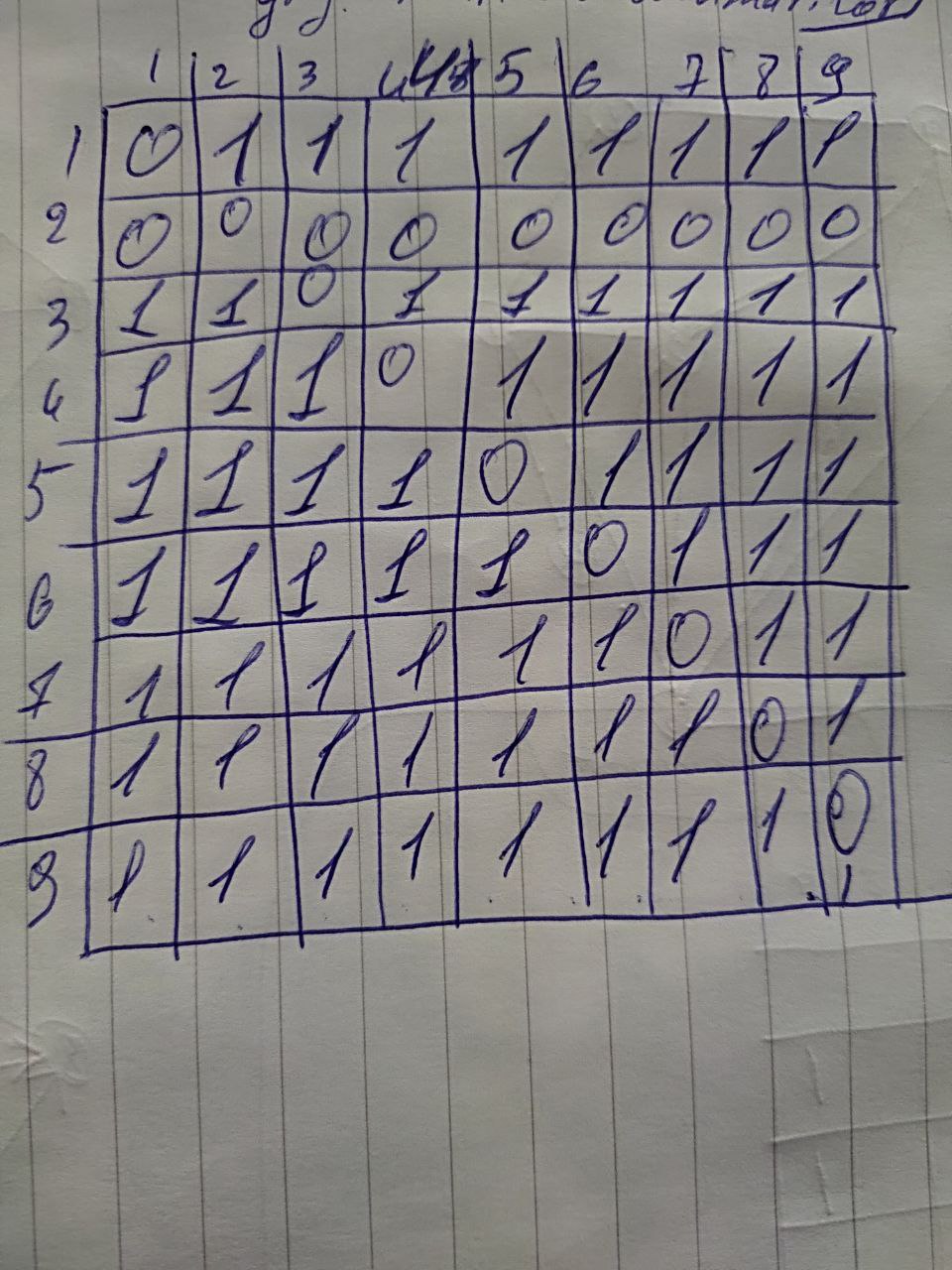


2. Folosind procedurile menţionate elaboraţi programul care va permite:

• introducerea grafului reprezentat sub oricare din cele trei forme cu posibilităţi de corecţie a datelor;

• extragerea informaţiei la display.

3. Elaborați procedura de determinare a matricei drumurilor într-un graf orientat.



4. Realizaţi un program cu următoarele funcţii:

• introducerea grafului ca matrice de adiacență (graf orientat) cu posibilităţi de corectare a informaţiei;

• determinarea matricei drumurilor;

• extragerea informaţiei la display;

#include <stdio.h>

void meniu(int arr[9][9], int Tab[16][9], int array[9][9]);

void Afisare\_Matrice\_mxn(int Tab[16][9]);

void Matrice\_orientat\_inciadenta(int Tab[16][9]);

void Matrice\_null\_nxn\_orientat(int arr[9][9]);

void Matrice\_null\_nxn\_neorientat(int array[9][9]);

void Matrice\_null\_mxn(int Tab[16][9]);

void Matrice\_orientat(int arr[9][9]);

void Matrice\_neorientat(int array[9][9]);

void Afisare\_matrice\_nxn\_orientat(int arr[9][9]);

void Afisare\_matrice\_nxn\_neorientat(int array[9][9]);

void Matrice\_Drumurilor(int arr[9][9]);

int main() {

    int array[9][9];

    int arr[9][9];

    int Tab[16][9];

    meniu(arr, Tab, array);

    return 0;

}

void meniu(int arr[9][9], int Tab[16][9], int array[9][9]) {

    int option;

    printf("\n--------Lista de optiuni--------\n"

           "1. Egalarea matriciei patratice cu zero\n"

           "2. Egalarea matriciei MxN cu zero\n"

           "3. Introducerea varfurilor adiacente neorientat\n"

           "4. Introducerea varfurilor adiacente orientat\n"

           "5. Introducerea varfurilor inciadente\n"

           "6. Afisarea matricielor Adiacente\n"

           "7. Afisarea matriciei Indiacenta\n"

           "8. Efectuarea determinarii matriciei drumurilor\n"

           "0. Iesire din program\n");

    printf("Selectati si introduceti nr optiunii dorite: ");

    scanf("%d", &option);

    switch (option) {

        case 0: {

            printf("Se efectueaza iesire din program...");

            break;

        }

        case 1: {

            Matrice\_null\_nxn\_orientat(arr);

            Matrice\_null\_nxn\_neorientat(array);

            meniu(arr, Tab, array);

            break;

        }

        case 2: {

            Matrice\_null\_mxn(Tab);

            meniu(arr, Tab, array);

            break;

        }

        case 3: {

            Matrice\_neorientat(array);

            meniu(array, Tab, arr);

            break;

        }

        case 4: {

            Matrice\_orientat(arr);

            meniu(arr, Tab, array);

            break;

        }

        case 5: {

            Matrice\_orientat\_inciadenta(Tab);

            meniu(arr, Tab, array);

            break;

        }

        case 6: {

            Afisare\_Matrice\_mxn(Tab);

            printf("\n\n");

            Afisare\_matrice\_nxn\_neorientat(array);

            meniu(arr, Tab, array);

            break;

        }

        case 7: {

            Afisare\_matrice\_nxn\_orientat(arr);

            printf("\n\n");

            Afisare\_matrice\_nxn\_neorientat(array);

            meniu(arr, Tab, array);

            break;

        }

        case 8: {

            Matrice\_Drumurilor(arr);

            meniu(arr, Tab, array);

            break;

        }

        default: {

            printf("\n-----------Nu exista optiunea!!!-----------\n");

            meniu(arr, Tab, array);

        }

    }

}

void Afisare\_Matrice\_mxn(int Tab[16][9]) {

    for (int i = 1; i <= 16; i++) {

        for (int j = 1; j <= 9; j++) {

            printf("%d ", Tab[i][j]);

        }

        printf("\n");

    }

}

    // Implementarea ta pentru introducerea varfurilor inciadente

    void Matrice\_orientat\_inciadenta(int Tab[16][9]) {

    int i, j, k;

    printf("Introduceti pozitia liniei: ");

    scanf("%d", &i);

    printf("Introduceti punctul initial si final: ");

    scanf("%d %d", &j, &k);

    Tab[i][j] = -1;

    Tab[i][k] = 1;

}

void Matrice\_null\_nxn\_orientat(int arr[9][9]) {

    for (int i = 1; i <=9; i++) {

        for (int j = 0; j < 9; j++) {

            arr[i][j] = 0;

        }

    }

}

void Matrice\_null\_nxn\_neorientat(int array[9][9]) {

    for (int i = 1; i <=9; i++) {

        for (int j = 1; j <= 9; j++) {

            array[i][j] = 0;

        }

    }

}

void Matrice\_null\_mxn(int Tab[16][9]) {

    for (int i = 1; i <= 16; i++) {

        for (int j = 1; j <= 9; j++) {

            Tab[i][j] = 0;

        }

    }

}

void Matrice\_orientat(int arr[9][9]) {

    int i, j;

    printf("Introduceti linia: ");

    scanf("%d", &i);

    printf("Introduceti coloana: ");

    scanf("%d", &j);

    arr[i][j] = 1;

}

void Matrice\_neorientat(int array[9][9]) {

    int i, j;

    printf("Introduceti pozitia varfurilor: ");

    scanf("%d %d", &i, &j);

    array[i][j] = 1;

    array[j][i] = 1;

}

void Afisare\_matrice\_nxn\_orientat(int arr[9][9]) {

    for (int i = 1; i <= 9; i++) {

        for (int j = 1; j <= 9; j++) {

            printf("%d ", arr[i][j]);

        }

        printf("\n");

    }

}

void Afisare\_matrice\_nxn\_neorientat(int array[9][9]) {

    for (int i = 1; i <= 9; i++) {

        for (int j = 1; j <= 9; j++) {

            printf("%d ", array[i][j]);

        }

        printf("\n");

    }

}

void Matrice\_Drumurilor(int arr[9][9]) {

    for (int i = 1; i <= 9; i++) {

            for (int j = 1; j <= 9; j++) {

if(i=j){

arr[i][j]=0;}

}

}

    for (int k = 1; k <= 9; k++) {

        for (int i = 1; i <= 9; i++) {

            for (int j = 1; j <= 9; j++) {

                if (i != j && arr[i][j] == 0 && arr[i][k] == 1 && arr[k][j] == 1)

                    arr[i][j] = 1;

            }

        }

    }

}

void Matrice\_Drumurilor(int arr[9][9]) {

    for (int k = 1; k <= 9; k++) {

        for (int i = 1; i <= 9; i++) {

            for (int j = 1; j <= 9; j++) {

                if (i != j && arr[i][j] == 0 && arr[i][k] == 1 && arr[k][j] == 1)

                    arr[i][j] = 1;

            }

        }

1. explicatia celor 3 for

for (int k = 1; k <= 9; k++)`: Primul for parcurge toate nodurile posibile (k) de la 1 la 9, reprezentând un nod intermediar în căutarea drumurilor.

for (int i = 1; i <= 9; i++)`: for 2 parcurge toate nodurile de la 1 la 9, reprezentând nodul sursă al unui posibil drum.

for (int j = 1; j <= 9; j++)`: al 3 for parcurge toate nodurile de la 1 la 9, reprezentând nodul destinație al aceluiași posibil drum.

2. Se verifica Contiile:

- `if (i != j && arr[i][j] == 0 && arr[i][k] == 1 && arr[k][j] == 1)`: Această condiție verifică dacă există un drum de la nodul `i` la nodul `j` trecând prin nodul intermediar `k`. Condițiile suplimentare `i != j` și `arr[i][j] == 0` asigură că nu se compară un nod cu el însuși și că nu există deja o legătură directă între `i` și `j`.

3. Se actualizeaza matricai Drumurilor

- `arr[i][j] = 1;`: Dacă condițiile sunt îndeplinite, atunci se marchează că există un drum de la nodul `i` la nodul `j` prin nodul intermediar `k`, astfel actualizând matricea drumurilor.

# **3. Concluzii**

În urma acestei experiențe de laborator, am acumulat cunoștințe solide în domeniul operațiilor cu grafuri și modalităților lor de reprezentare. Am căpătat înțelegere asupra modului în care grafurile sunt stocate în memoria calculatorului și am explorat proprietățile fiecărei forme de reprezentare, inclusiv impactul asupra utilizării memoriei și eficienței prelucrării datelor.

Pe durata lucrării, am revizuit și consolidat cunoștințele referitoare la manipularea tablourilor bidimensionale, dobândind o perspectivă mai clară asupra operațiilor și structurii acestora. Am dezvoltat o înțelegere mai profundă a conceptelor din matematica discretă și modul în care acestea se aplică în domeniul programării. Prin experimentarea cu diverse aspecte ale grafurilor, am dobândit abilități practice în manipularea și analiza datelor reprezentate sub formă de grafuri.