Sesiunea 9

Agenda

- Rezolvare exercitii Sesiunea 8
- Introducere in functii
- Exercitii functii
- Tema

Rezolvare exercitii tema Sesiunea 8

1. Rezolvare

```
#include <iostream>
#include <fstream>

using namespace std;

int main(){
    int n;
    cin >> n;
    float matrice[n][n];
    float blurat[n][n];
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            cin >> matrice[i][j];
        }
    }
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            float medieVecini =0.0;

            float vecinNord = matrice[i-1][j];
            float vecinSud = matrice[i+1][j];
            float vecinVest = matrice[i][j-1];
            float vecinEst = matrice[i][j+1];
            if (i == 0) {
                if (j == 0) {
                    medieVecini = (vecinEst + vecinSud) / 2;
                } else if (j == n - 1) {
                    medieVecini = (vecinVest + vecinSud) / 2;
                } else {
                    medieVecini = (vecinVest + vecinEst + vecinSud) / 3;
                }
            } else if (i == n - 1) {
                if (j == 0) {
                    medieVecini = (vecinNord + vecinEst) / 2;
                } else if (j == n - 1) {
                    medieVecini = (vecinNord + vecinVest) / 2;
                } else {
```

```
                        medieVecini = (vecinNord + vecinVest + vecinEst) /
3;
                    }
                } else {
                    if (j == 0) {
                        medieVecini = (vecinNord + vecinSud + vecinEst) / 3;
                    } else if (j == n - 1) {
                        medieVecini = (vecinNord + vecinSud + vecinVest) /
3;
                    } else {
                        medieVecini = (vecinEst + vecinSud + vecinVest +
vecinNord) / 4;
                    }
                }

                blurat[i][j] = medieVecini;
            }
        }

        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                cout << blurat[i][j] << " ";
            }
            cout << endl;
        }

        return 0;
    }
```

2. Rezolvare:

```
    #include <iostream>
    #include <fstream>

    using namespace std;

    int main(){
        int n;
        cin >> n;
        int matrice[n][n];
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                cin >> matrice[i][j];
            }
        }
        int sumaZonaNord = 0, sumaZonaEst = 0, sumaZonaSud = 0, sumaZonaVest
= 0;
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                if (j > i && i+j < n - 1) {
                    sumaZonaNord += matrice[i][j];
                } else if (j > i && i + j > n - 1 ) {
```

```
                    sumaZonaEst += matrice[i][j];
                } else if (i > j && i + j > n - 1) {
                    sumaZonaSud += matrice[i][j];
                } else if (i > j && i+j < n-1) {
                    sumaZonaVest += matrice[i][j];
                }
            }
        }

        int sumaEV = sumaZonaEst + sumaZonaVest;
        int sumaNS = sumaZonaSud + sumaZonaNord;

        if (sumaEV > sumaNS) {
            cout << sumaEV - sumaNS << endl;
        } else {
            cout << sumaNS - sumaEV << endl;
        }


        return 0;
    }
```

3. Rezolvare:

```
        #include <iostream>

        using namespace std;
        /**
        * Observam ca pe diagonala principala avem multiplii lui k pe care
ii putem calcula astfel:
        * matrice[i][j] = k * (i+1). Folosim i+1 deoarece pornim de la 0 cu
indicele.
        * Dupa care observam ca pentru fiecare element din dreapta
diagonalei principale, scadem din valoarea
        * de pe diagonala principala, distanta dintre elementul curent si
diagonala principala, lucru pe care
        * il aflam cu j - i
        *
        * Iar pentru elementele din stanga diagonalei principale, adunam la
diagonala principala distanta dintre
        * elementul curent si diagonala principala, lucru pe care il aflam
cu i - j.
        */

        int main() {
            int k,n;
            cin >> k >> n;
            int matrice[n][n];
            for (int i = 0; i < n; i++) {
                for (int j = 0; j < n; j++) {
                    int valoareDiagonalaPrincipala = k * (i+1);
                    if (i == j) {
```

```
                        matrice[i][j] = valoareDiagonalaPrincipala;
                    } else if (j > i) {
                        matrice[i][j] = valoareDiagonalaPrincipala - (j-i);
                    } else {
                        matrice[i][j] = valoareDiagonalaPrincipala + (i -
j);
                    }
                }
            }

            for (int i = 0; i < n; i++) {
                for (int j = 0; j < n; j++) {
                    cout << matrice[i][j] << " ";
                }
                cout << endl;
            }

            return 0;

        }
```

4. Rezolvare:

```
    #include <iostream>

    using namespace std;

    int main() {
        int i, j;
        int a[9][9];
        // Pentru bac trebuie sa scrii doar ce este intre liniile hasurate
        //----------------------------------------
        for(i = 0; i < 9; i++) {
            for (j = 0; j< 9; j++) {
                if (i +j <= 3 || i+j >= 13 ) {
                    a[i][j] = 4;
                } else {
                    a[i][j] = 2;
                }
            }
        }
        //----------------------------------------

        for(i = 0; i < 9; i++) {
            for (j = 0; j< 9; j++) {
                cout << a[i][j] << " ";
            }
            cout << endl;
        }
        return 0;
    }
```

Introducere in functii / subprograme

- O funcție definește o anumită sarcină pe care dorim să o îndeplinim, cum ar fi:
    - calcularea pătratului unui număr-
    - afișarea unui mesaj gen: "Hello world"
    - afișarea numelui complet al unei persoane
    - etc.
- Daca luam de exemplu o functie care calculeaza media geometrica a 3 numere:
    - Te poti gandi ca in loc sa copiezi undeva la 4,5 linii de cod de fiecare data, o sa ai doar o linie de cod.
    - O sa fie ceva gen: `calculeazaMedie(x,y,z)` sau `calculeazaMedie(a,b,c)`
    - daca nu ai fi avut functii, trebuie sa copiezi peste tot, toata logica ce iti calcula media.
- Un avantaj mare al functiilor este ca daca avem o problema de logica in calculul/operatiile pe care il/le face functia noastra (si prespunem ca e chemata in muuuulte locuri) noi vom avea de corectat intr-un singur loc, anume unde am definit functia si solutia probleme se va propaga automat peste tot unde e folosita.
- Pentru definirea unei funcții, vom utiliza următorul șablon:

```
    <tip întoarcere> <nume funcție> (<listă de parametri>){
    <corpul funcției>
}
```

    - exemplu cu functie care intoarce o valoare:

```
int computeAverage(int x, int y) {
    int average = ( x + y ) / 2;

    return average;
}
```

    - exemplu cu functie care nu intoarce o valoare:

```
void displayAverage(int x, int y) {
    int average = ( x + y ) / 2;

    cout << "Media celor doua numere este: " << average;
}
```

- Acum, sa disecam fiecare parte din sablonul functiei:

- `<tip întoarcere>`
  - O funcție poate întoarce una sau zero valori.
  - Daca intoarce o valoare aceasta poate fi de tipul `int`, `float`, `double`, etc.
  - Dacă nu întoarce o valoare, ar trebui să folosim cuvântul cheie `void` înainte de a declara numele funcției.
- `<nume funcție>`
  - reprezintă numele pe care îl dăm acelei bucăți de cod care gestionează o sarcină specifică pentru noi. Acest lucru este util atunci când dorim să apelăm funcția ulterior.
    - Apelul unei funcții este ca o scurtătură, dorim doar să aducem funcționalitatea implementata in functie, pentru a o folosi în logica noastră, fără a o recrea.
      - De multe ori, functiile pe care le folosim, nici nu sunt scrise de noi. De exemplu, folosim funcția `cout`, dar aceasta este o bucată de cod care nu a fost creată de noi.
- `<listă de parametri>`
  - reprezintă datele de intrare folosite de funcția noastră pentru a-și îndeplini responsabilitățile.
  - Putem extrapola putin si sa vedem functia ca pe o reteta de prajituri
    - `parametrii` sunt ingredientele pentru rețeta noastră (de exemplu, zahăr, lapte, etc.)
    - rețeta este logica care amestecă ingredientele (a.k.a. parametrii)
    - prajitura va fi rezultatul final al funcției noastre atunci când este apelată.

Nota:

- Cu excepția funcției principale (adica `main`), fiecare funcție trebuie să fie definită înainte de a fi folosită. Atunci când definim o funcție, trebuie doar să specificăm tipul întoarcerii, numele și lista de parametri, de exemplu:
  - `void afișareAdresă();`
  - `int calculDistanță(int x, int y);`
- De retinut ca o functie poate sa cheme o alta functie (sau mai multe). A se vedea exercitiul `6`.

Exemple functii

1. Scrieti o functie care transforma secundele, primite ca si parametru, in ore, minute si secunde, dupa care afisaza rezultatul in formatul: `HH:MM:SS`.

- Solutie:

```
#include <iostream>
using namespace std;

void convertesteSecunde(int secunde);

int main() {
    int secunde;
    cout << "Introduceti numarul total de secunde: ";
    cin >> secunde;
    convertesteSecunde(secunde);
```

```
        return 0;
    }

    void convertesteSecunde(int secunde) {
        int ore = secunde / 3600;
        int minute = (secunde % 3600) / 60;
        int secundeRamase = secunde - (ore * 3600) - (minute * 60);

        cout << ore<<":"<<minute<<":"<<secundeRamase;
    }
```

2. Scrieti o functie care primeste ca si parametru un numar natural n. unde n > 100 si n <= 99999. Functia va returna cate cifre ale numarului sunt impare, -1 in cazul in care niciuna nu indeplineste conditia.

- Date de intrare: n = 123456
- Date de iesire: 3
- Date de intrare: n = 2468
- Date de iesire: -1
- Solutie:

```
#include <iostream>
using namespace std;

int numaraImpare(int n);

int main() {
    int n;
    cout << "Introduceti n: ";
    cin >> n;
    cout << numaraImpare(n);

    return 0;
}

int numaraImpare(int n) {
    int contor = 0;
    while (n > 0) {
        int ultimaCifra = n % 10;
        if (ultimaCifra % 2 != 0) {
            contor++;
        }
        n = n / 10;
    }
    if (contor == 0) {
        return -1;
    } else {
        return contor;
    }
}
```

3. Subprogramul `DoiTrei` are un parametru, `n`, prin care primeste un numar natural (n apartine intervalului [0, 10^9]). Subprogramul returneaza valoarea 1 daca toate cifrele lui `n` sunt din multimea `{2,3}` sau valoarea 0 in caz contrar. Scrieti definitia completa a programului.

- Date de intrare: `n` = 22323
- Date de iesire: 1
- Date de intrare: `n` = 2023
- Date de iesire: 0
- Solutie:

```
#include <iostream>
using namespace std;

int DoiTrei(int n);

int main() {
    int n;
    cout << "Introduceti n: ";
    cin >> n;
    cout << DoiTrei(n);

    return 0;
}

int DoiTrei(int n) {
    int rezultat = 1;
    while (n > 0) {
        int ultimaCifra = n % 10;
        if (ultimaCifra != 2 && ultimaCifra != 3) {
            rezultat = 0;
            break;
        }

        n = n / 10;
    }

    return rezultat;
}
```

4. Scrieti un program care sa contina un subprogram ce va primi ca si parametru un numar `n`. Subprogramul va afisa toti divizorii numarului, separati prin spatiu.

- Date de intrare: `n` = 15
- Date de iesire: 1,3,5,15
- Solutie:

```
#include <iostream>
using namespace std;
```

```
void afiseazaDivizori(int n);
int main() {
    int n;
    cout << "Introduceti n: ";
    cin >> n;
    afiseazaDivizori(n);

    return 0;
}

void afiseazaDivizori(int n) {
    for (int i = 1; i <= n; i++) {
        if ( n % i == 0) {
            cout << i << " ";
        }
    }
}
```

5. Scrieti un program care sa contina un subprogram ce va primi ca si parametru un numar `n`. Subprogramul va returna `1` daca numarul este `prim` sau `0` in caz contrar.

- Date de intrare: `n` = 21
- Date de iesire: 0
- Date de intrare: `n` = 13
- Date de iesire: 1
- Solutie:

```
    #include <iostream>
    using namespace std;

    int estePrim(int n);

    int main() {
        int n;
        cout << "Introduceti n: ";
        cin >> n;
        cout << estePrim(n);

        return 0;
    }

    int estePrim(int n) {
        if (n < 2) return 0;
        if (n == 2) return 1;
        if (n % 2 == 0) return 0;

        int rezultat = 1;
        for (int i = 3; i < n/2; i++) {
            if (n % i == 0) {
                rezultat = 0;
                break;
```

```
            }
        }

        return rezultat;
    }
```

6. Scrieti un program C++ care citeste de la tastatura un sir de caractere de maximum 50 de caractere. Programul va numara cate consoane si cate vocale sunt in sir. Atentie la faptul ca sirul de caractere poate contine caractere alfanumerice si literele vor fi toate litere mici ale alfabetului.

- Date de intrare: "ana are ac cu ata de 2 metri."
- Date de iesire:
  - `Consoane: 9`:
  - `Vocale: 11`
- Solutie:

```
#include <iostream>
#include <string.h>

using namespace std;

int esteVocala(char ch);
int esteConsoana(char ch);

int main() {
char propozitie[50];
cout << "Introduceti maximum 50 de caractere de la tastatura: ";
cin.getline(propozitie, 50);

int totalConsoane = 0;
int totalVocale = 0;
for (int i = 0; i < strlen(propozitie); i ++) {
    if (esteVocala(propozitie[i]) == 1) {
        totalVocale++;
    } else if (esteConsoana(propozitie[i])) {
        totalConsoane++;
    }
}

cout << "Consoane: " << totalConsoane << endl;
cout << "Vocale: " << totalVocale;

    return 0;
}

int esteVocala(char ch) {
    return strchr("aeiou", ch) != NULL;
}

int esteConsoana(char ch) {
    if (ch >= 'a' && ch <= 'z' && !esteVocala(ch)){
```

```
            return 1;
        } else {
            return 0;
        }
    }
```

Functii care intorc valori prin parametrii

Descriere

- Dupa cum vazut in exercitiile anterioare o functie poate sa intoarca o valoare (folosind cuvantul cheie `return` + tipul valoarii intoarse in antetul functiei) sau sa nu o faca, caz in care vom folosi cuvantul cheie `void` in antetul functiei.
- Pe langa aceste doua categorii de functii, mai exista si o a 3-a care este o combinatie intre cele doua anume, sunt functiile care intorc o valoare (sau mai multe) insa o fac prin intermediul parametrilor.
    - De cele mai multe ori, aceste functii contin `void` in antetul lor insa nimeni nu ne opreste din a intoarce atat prin parametri cat si folosind cuvantul cheie `return` + specificarea tipului de data intors, in antet, insa..stim si noi faptul ca daca putem, nu inseamna neaparat ca e ok 
    - Ce poate parea ciudat la inceput, este sintaxa prin care marcam faptul ca un parametru va fi folosit pentru a intoarce o valoare. Anume, acesta va fi precedat de `&` in antetul functiei. (`&` este operatorul de adresa:D vom vorbi mai mult despre el cand vom repeta pointerii)

Exemple

- Haide sa ne incalzim cu cateva exemple simple:

1. Să se scrie o funcție C++ care să determine suma cifrelor unui număr natural transmis ca parametru. Funcția întoarce rezultatul prin intermediul unui parametru de ieşire.

- Pentru `n` = 123456, vom intoarce 21 printr-un alt parametru
- Solutie:

```
#include <iostream>
using namespace std;

void sumaCifre(int n, int &y);

int main() {

    int n = 123456;
    int numeParametru;
    sumaCifre(n, numeParametru);

    cout << numeParametru;
    return 0;
}

void sumaCifre(int n, int &rezultat) {
    int suma = 0;
```

```
    while (n > 0) {
        int ultimaCifra = n % 10;
        suma += ultimaCifra;
        n = n /10;
    }
    rezultat = suma;
}
```

- Acum haide sa vorbim putin despre ce avem mai sus
    - Nu conteaza ce nume dai la parametru cand creezi functia. Poate sa fie, poate sa nu fie la fel cu numele pe care il vei da variabile pasate ca si argument (argument sau parametru, poteto-potato, tometo-tomato)
    - Dupa cum stim, atunci cand pasam o variabila ca si parametru, unei functii, orice modificari va suferi aceasta, ele nu vor fi vizible in afara functiei decat daca vom folosi operatorul de adresa `&`. Mai jos avem 2 exemple care evidentiaza acest lucru:

```
#include <iostream>
using namespace std;

void dubleazaValoareaPrimita(int n);
void dubleazaValoareaPrimitaCuAdresa(int &n);


int main() {

    int x = 4;
    cout << "Valoare x inainte de a apela functia ce dubleaza: " <<
x << endl;
    dubleazaValoareaPrimita(x);
    cout << "Valoare x dupa ce am apelat functia ce dubleaza: " <<
x << endl;

    cout << "-------------------------------------\n";
    cout << "Valoare x inainte de a apela functia ce dubleaza si
foloseste operatorul adresa: " << x << endl;
    dubleazaValoareaPrimitaCuAdresa(x);
    cout << "Valoare x dupa ce am apelat functia ce dubleaza: " <<
x << endl;
    return 0;
}

void dubleazaValoareaPrimita(int n) {
    n = n * 2;
    cout << "\n====In interiorul functiei care dubleaza valoarea
parametrului primit: ====" << endl;
    cout << "Dupa ce l-am dublat n = " << n;
    cout << "\n ==== Sfarsit functie care dubleaza valoarea
parametrului primit ====" << endl;
}
```

```
    void dubleazaValoareaPrimitaCuAdresa(int &n) {
        n = n * 2;
        cout << "\n====In interiorul functiei care dubleaza valoarea
 parametrului primit cu &: ====" << endl;
        cout << "Dupa ce l-am dublat n = " << n << endl;
        cout << "\n ==== Sfarsit functie care dubleaza valoarea
 parametrului primit cu & ====" << endl;
    }
```

2. Să se scrie o funcție C++ care să determine suma cifrelor pare și suma cifrelor impare pentru un număr natural transmis ca parametru. Funcția va întoarce rezultatele prin intermediul unor parametri de ieşire.

- Solutie:

```
#include <iostream>
using namespace std;

void calculSumaCifre(int n, int &sumaPare, int &sumaImpare);

int main() {

    int n = 2749;
    int sumaPare, sumaImpare;
    calculSumaCifre(n, sumaPare, sumaImpare);
    cout << sumaPare << endl << sumaImpare;
    return 0;
}

void calculSumaCifre(int n, int &sumaPare, int &sumaImpare) {
    // Asta e doar un exemplu prin care arat cum ne putem proteja de
valoarea cu care vin parametri
    sumaPare = 0;
    sumaImpare = 0;

    while (n > 0) {
        int ultimaCifra = n%10;
        if (ultimaCifra % 2 == 0) {
            sumaPare += ultimaCifra;
        } else {
            sumaImpare += ultimaCifra;
        }
        n = n/10;
    }
    // Observam faptul ca nu mai setam la final valoarea pentru cele doua
sume deoarece o facem din mers.
}
```

3. Să se scrie o funcție C++ care să determine prima și ultima cifră a unui număr natural transmis ca parametru. Funcția va întoarce rezultatele prin intermediul unor parametri de ieşire.

- Solutie:

```
#include <iostream>
using namespace std;

void determinaCapeteNumar(int n, int &primaCifra, int &ultimaCifra);

int main() {

    int n = 2749;
    int prima, ultima;
    determinaCapeteNumar(n, prima, ultima);
    cout << prima << " " << ultima;
    return 0;
}

void determinaCapeteNumar(int n, int &primaCifra, int &ultimaCifra) {
    primaCifra = ultimaCifra = n % 10;
    n = n/10;
    while (n > 0) {
        primaCifra = n % 10;
        n = n/10;
    }
}
```

Functii recursive

Descriere

- In programare, o `functie recursiva` este o functie care se cheama pe ea insasi.
- Trebuie sa retinem faptul ca orice functie recursiva se poate rescrie folosind o instructiune repetitiva (`for`, `while`, `do-while`)
- Orice functie recursiva are 2 parti importante:
  - Conditia de oprire (anume instructiunea care va face functia sa se opreasca)
  - Partea recursiva.
    - Aici trebuie retinut faptul ca, cu fiecare apel recursiv, trebuie sa ne apropiem de conditia de oprire, altfel, intram in loop infinit si bum 😄. Glumesc, sistemul de operare o sa se prinda ca programul o ia pe aratura si o sa ii taie macaroana.

Exemple

1. Sa consideram urmatorul exemplu in care scriem o functie recursiva ce calculeaza factorialul primelor `n` numere.

```
int factorial(int n) {
    if (n < 1) {
        return 1;
    }
```

```
    return n * factorial (n - 1);
}
```

- Aici, conditia de oprire este verificare pentru `n < 1` si partea recursiva este apelul functiei `factorial` unde la fiecare pas, `n` are o valoare cu 1 mai mica decat valoarea anterioara, deci cu fiecare pas ne apropiem de conditia de oprire.
- Acum, poate parea putin greu de inteles insa hai sa vedem cum arata apelurile functiei noastre, in cazul in care facem apelul `factorial (4)`:

1. `factorial(4)`: deoarece `4 > 1` functia noastra va intoarce `4 * factorial(3)`
2. `4 * factorial(3)` = `4 * ( 3 * factorial (2))`
3. `4 * (3 * factorial (2))` = `4 * ( 3 * (2 * factorial(1)))`
4. `4 * ( 3 * (2 * factorial(1)))` = `4 * ( 3 * (2 * ( 1 * factorial (0))))`
5. Acum pentru ca `factorial(0)` este egal cu 1, expresia anterioara devine: `4 * 3 * 2 * 1` = `24`

2. Urmatorul exemplu, va contine o functie recursiva pentru a calcula `cel mai mic divizor comun` folosind algoritmul lui Euclid.

```
int gcd(int a, int b) {
    if (b == 0){
        return a;
    }
    int reminder = a % b;
    return gcd(b,  reminder);
}
```

- Aici conditia de oprire este atunci cand `b = 0` si apelul recursiv, paseaza la fiecare apel `a % b` ca si valoare pentru parametrul `b`, deci cu fiecare apel ne apropiem de conditia de oprire

3. Scrieți funcția recursivă cu antetul `long long SumProdRec(int n)` care primind ca parametru un număr natural nenul n, returnează valoarea sumei 1•2 + 2•3 + ... (n-1)•n.

- Input: 4
- Output: 20
- Solution:

```
#include <iostream>
using namespace std;

long long SumProdRec(int n);

int main() {
    int n;
    cout << "Enter n:";
    cout << SumProdRec(n);
    return 0;
}
```

```
long long SumProdRec(int n)
{
    if (n <= 1) return 0;
    return (n - 1) * n + SumProdRec(n - 1);
}
```

4. Scrieți funcția recursivă DivImpRec care primind ca parametru un număr natural nenul n, returnează cel mai mare divizor impar al său.

- Input: 24
- Output: 3
- Solution:

```
#include <iostream>
using namespace std;

int DivImpRec(int n);

int main() {
    int n;
    cout << "Enter n:";
    cout << DivImpRec(n);
    return 0;
}

int DivImpRec(int n)
{
    if (n % 2 == 1) return n;
    return DivImpRec(n / 2);
}
```

5. Să se scrie o funcție C++ recursivă care să returneze numărul cifrelor divizibile cu 3 ale unui număr natural n transmis ca parametru.

- Input: 2009376
- Output: 5
- Solution:

```
#include <iostream>
using namespace std;

int CifDiv3Rec(int n);

int main() {
    int n;
    cout << "Enter n:";
    cout << CifDiv3Rec(n);
    return 0;
```

```
}

int CifDiv3Rec(int n) {
    if (n == 0) {
        return 0;
    }
    int lastDigit = n % 10;
    if (lastDigit % 3 == 0) {
        return 1 + CifDiv3Rec(n / 10);
    }
    else {
        return 0 + CifDiv3Rec(n / 10);
    }
}
```

Tema

1. Scrie o funcție `void salut()` care afișează „Salut, lume!”. Apeleaz-o din main.
2. Scrie o funcție `int patrat(int n)` care primește un număr și returnează pătratul lui.
3. Scrie o funcție `float suma(float a, float b)` care primește două numere reale și returnează suma lor.
4. Scrie o funcție `int estePar(int n)` care verifică dacă un număr este par și returnează true/false.
5. Scrie o funcție `void afiseazaCaracter(char c, int n)` care afișează caracterul c de n ori pe ecran.
6. Scrie o funcție `void schimba(int &a, int &b)` care interschimbă valorile a două variabile.
7. Scrie o funcție `void dubleaza(int &x)` care dublează valoarea variabilei primite.
8. Scrie o funcție `void citesteNumar(int &x)` care citește de la tastatură un număr și îl returnează prin parametrul de referință.
9. Scrie o funcție `float medie(float a, float b, float c)` care calculează media aritmetică a 3 numere.
10. Scrie o funcție `int max3(int a, int b, int c)` care returnează maximul dintre 3 valori.
11. Scrie o funcție void citireVector(int v[], int n) care citește de la tastatură un vector de n elemente.
12. Scrie o funcție void afisareVector(int v[], int n) care afișează vectorul.
13. Scrie o funcție int sumaVector(int v[], int n) care returnează suma elementelor unui vector.
14. Scrie o funcție int maximVector(int v[], int n) care returnează cel mai mare element.
15. Scrie o funcție void sorteazaVector(int v[], int n) care sortează vectorul crescător (poți folosi metoda „bubble sort”).
16. Definim o structură `Student { char nume[30]; float nota; }`

    - Scrie o funcție `void citireStudent(Student &s)` care citește datele unui student.
    - Scrie o funcție `void afisareStudent(Student s)` care afișează datele.
    - Scrie o funcție `float medieStudenti(Student v[], int n)` care returnează media notelor unui grup de studenți.

- Nota:
    - O buna parte din exercitiile urmatoare sunt exercitii extrase din variante de bac.
    - Nu te stresa daca nu le intelegi sau daca nu le poti face, orice ar fi, o sa le discutam sedinta urmatoare cand vei primi si solutia la ele dar te rog, incearca-le, sparge-le logica in bucatele mici pe care sigur le gasesti in ceea ce am facut.
    - Partea buna daca le rezolvi, este ca o sa ai cel putin 2 solutii pentru exercitiile pe care le faci. Asta deoarece sunt sanse foarte mari ca fiecare dintre noi sa o rezolvam in felul nostru (complet diferit) si totusi sa fie corect.
    - De asemenea, daca un exercitiu nu e clar, is sanse ca eu sa-l fi scris gresit, poti oricand sa ma cauti pe whatsapp sa imi zici si o sa corectez.

17. Scrieti un subprogram care elimina toate cifrele impare dintr-un numar `n` primit ca parametru. Se garanteaza faptul ca `n` >= 10; Programul va return numarul modificat.

- Exemplu: Pentru `n` = 123456, subprogramul va intoarce numarul: `246`

18. Scrieti un subprogram care primeste ca si parametru un numar `n`. Subprogramul va inlocui fiecare cifra impara cu dublul acesteia. In cazul in care dublul cifrei impare va fi mai mare decat 10, se va lua ultima cifra.

- Exemplu pentru `n` = 123456, subprogramul va intoarce numarul: `226406`

19. Un joc online cu n jetoane poate fi jucat de un grup de k (k≥2) jucători, numai dacă toate cele n jetoane pot fi distribuite în mod egal celor k jucători. Subprogramul joc are un singur parametru, n, prin care primește un număr natural (n apartine [2,104]), reprezentând numărul de jetoane ale unui joc de tipul precizat. Subprogramul returnează numărul valorilor distincte pe care le poate avea k pentru acest joc. Scrieți definiția completă a subprogramului. Exemplu: dacă n=12, atunci subprogramul returnează numărul 5 (cele 12 jetoane se pot distribui în mod egal pentru o grupă de 2 jucători, de 3 jucători, de 4 jucători, de 6 jucători sau de 12 jucători)
    - Link: https://modinfo.ro/bac/variante-test-2021/info/v4.pdf

20. Subprogramul identice are un singur parametru, n, prin care primește un număr natural (n apartine intervalului [10,109]). Subprogramul returnează valoarea 1, dacă numărul n are toate cifrele egale, sau valoarea 0 în caz contrar. Scrieți definiția completă a subprogramului. Exemplu: dacă n=2222, subprogramul returnează valoarea 1, iar dacă n=212, subprogramul returnează valoarea 0
    - Link: https://modinfo.ro/bac/variante-test-2021/info/v5.pdf

21. Subprogramul `afisare` are trei parametri:
    - x și y, prin care primește câte un număr natural din intervalul [0,106] (x <= y);
    - k, prin care primește un număr natural (k□[2,102]).

Subprogramul afișează pe ecran, în ordine strict crescătoare, numerele din intervalul [x,y], în secvențe de câte k, cu excepția ultimei secvențe care poate conține mai puțin de k numere.

- Fiecare secvență se încheie cu câte un simbol `*`, iar numerele și simbolurile sunt separate prin câte un spațiu, ca în exemplu. Scrieți definiția completă a subprogramului. Exemplu: dacă x=11, y=21 și k=4 se afișează pe ecran numerele de mai jos, în acest format. 11 12 13 14 * 15 16 17 18 * 19 20 21 *

22. Să se scrie un subprogram C++ prin care se dublează prima cifră a unui număr natural n transmis ca parametru. Funcția întoarce rezultatul prin intermediul aceluiași parametru n.

- Link:
  - Problema este luata de aici: https://www.pbinfo.ro/probleme/1633/dublare1. Poti vedea la pagina asta mai multe detalii, cum ar fi un exemplu de date de intrare dar desigur, te poti si verifica la ei pe site.

23. Să se scrie o funcție C++ care primește ca parametri două numere n și k și determină numărul format din primele k cifre ale lui n. Funcția va întoarce rezultatul prin intermediul unui parametru de ieşire.

- Link:
  - Problema este luata de aici: https://www.pbinfo.ro/probleme/910/kprefix. Poti vedea la pagina asta mai multe detalii, cum ar fi un exemplu de date de intrare dar desigur, te poti si verifica la ei pe site.

24. Subprogramul `numar` are trei parametri:

- `n` și `c`, prin care primește câte un număr natural (n apartine intervalului [0,109], `c` apartine intervalului [0,9]);
- `m`, prin care furnizează numărul obținut din `n`, prin eliminarea din acesta a tuturor cifrelor egale cu c, sau -1 dacă toate cifrele lui n sunt egale cu c. Cifrele nule nesemnificative sunt ignorate, ca în exemplu. Scrieți definiția completă a subprogramului. Exemplu: dacă `n=50752` sau `n=72` și `c=5`, după apel `m=72`, dacă `n=500` și `c=5`, după apel m=0, iar dacă `n=55` și `c=5`, după apel `m=-1`.

25. Să se scrie o funcție C++ recursivă care să returneze suma cifrelor unui număr natural transmis ca parametru.

- Link: https://www.pbinfo.ro/probleme/823/sumcifrec Poti vedea la pagina asta mai multe detalii, cum ar fi un exemplu de date de intrare dar desigur, te poti si verifica la ei pe site.

26. Să se scrie o funcție C++ recursivă care determină cel mai mare divizor comun a două numere transmise ca parametri și întoarce rezultatul prin intermediul unui parametru de ieșire.

- Link: https://www.pbinfo.ro/probleme/917/cmmdcrec1 Poti vedea la pagina asta mai multe detalii, cum ar fi un exemplu de date de intrare dar desigur, te poti si verifica la ei pe site.
- De asemenea, problema asta e mai tricky putin, deoarece combina doua concepte pe care le-am invatat in sesiunea asta, anume functiile recursive si functiile care intorc valori prin intermediul parametrilor de intrare.