Sesiunea 2

Topicuri

- Recapitulare rapida
- Rezolvare exercitii anterioare
- Extra exercitii cu functii
- Functii care intorc valori prin parametrii
    - Descriere
    - Exemple
- Functii recursive
    - Descriere
    - Exemple
- Exercitii de antrenament
- Preview sedinta urmatoare

Recapitulare rapida

- Q&A

Rezolvare exercitii anterioare

1. Scrieti un program care sa contina o functie ce va calcula restul impartirii a doua numere primite ca si parametru.

- Date de intrare: 13, 7
- Date de iesire: 6
- Solutie:

```
#include <iostream>

using namespace std;

int calculeazaRest(int x, int y);


int main() {

    cout << calculeazaRest(13, 7);
    return 0;
}

int calculeazaRest(int x, int y) {
    return x % y;
}
```

2. Scrieti un program care sa contina un subprogram ce va primi ca parametru un numar n si o cifra k, unde n cuprins in intervalul [999, 999999] si k cuprins in intervalul [0, 9]. Functia va intoarce numarul de aparitii ale cifrei k in numarul n sau -1 daca nu se gaseste nici macar o aparitie.

- Date de intrare: n = 545343, k = 3
- Date de iesire: 2
- Solutie:

```
#include <iostream>

using namespace std;

int calculeazaAparitii(int n,  int k);

int main() {

    int n = 545343;
    int k = 3;
    cout << calculeazaAparitii(n, k);
    return 0;
}

int calculeazaAparitii(int n,  int k) {
    int contor = 0;
    while (n > 0) {
        int ultimaCifra = n % 10;
        if (ultimaCifra == k) {
            contor++;
        }
        n = n/10;
    }

    if (contor == 0) {
        return -1;
    } else {
        return contor;
    }
}
```

3. Subprogramul calcul are un singur parametru, n, prin care primeste un numar natural (n apartine intervalului [2, 1000]). Subprogramul returneaza suma divizorilor proprii lui n care sunt numere prime. Scrieti definitia completa a subprogramului. Exemplu: Daca n=15, dupa apel, subprogramul va return valoarea 8. Numerele prime care sunt divizori proprii ai lui 15 sunt 3 si 5.

- Solutie:https://modinfo.ro/bac/variante-test-2021/info/v2.pdf

```
#include <iostream>

using namespace std;
```

```
int estePrim(int n);
int calcul(int n);

int main() {

    int n = 15;
    cout << calcul(n);
    return 0;
}

int calcul(int n) {
    int suma = 0;
    for(int i = 2; i < n; i++) {
        if (n % i == 0 && estePrim(i)){
            suma += i;
        }
    }
    return suma;
}

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

    for (int i = 2; i * i <= n; i++) {
        if (n % i == 0) {
            return 0;
        }
    }

    return 1;
}
```

Extra exercitii cu functii

1. Subprogramul divX are doi parametri, n și x, prin care primește câte un număr natural din intervalul [2,50]. Subprogramul afișează pe ecran, în ordine descrescătoare, separate prin câte un spațiu, primele n numere naturale nenule divizibile cu x. Scrieți definiția completă a subprogramului. Exemplu: dacă n=4 și x=15 în urma apelului se afișează numerele 60 45 30 15

- Link: https://modinfo.ro/bac/variante-test-2021/info/v1.pdf
- Solutie:

```
#include <iostream>

using namespace std;

void divX(int n, int x);
```

```
int main() {

    int n = 4;
    int x = 15;
    divX(n, x);

    return 0;
}

void divX(int n, int x) {
    for(int i = 4; i > 0; i--) {
        cout <<  i * x << " ";
    }
}
```

2. Subprogramul factori are doi parametri, n și m, prin care primește câte un număr natural din intervalul [1,109]. Subprogramul returnează numărul valorilor prime care apar la aceeași putere atât în descompunerea în factori primi a lui n, cât și în descompunerea în factori primi a lui m. Scrieți definiția completă a subprogramului. Exemplu: dacă n=16500 și m=10780, atunci subprogramul returnează 2 (16500=2^2*35^3*11, 10780=2^2*57^2*11).

- Link: https://modinfo.ro/bac/variante-test-2021/info/v2.pdf
- Solutie:

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

int estePrim(int n);
int factori(int n, int m);

int main() {

    int n = 16500;
    int m = 10780;

    cout << factori(n, m);
    return 0;
}

int factori(int n, int m) {
    int contor = 0;
    for(int i = 2; i <= n ; i++) {
        // Punem conditia de estePrim la inceput ca sa scurtam
operatiile
        if (estePrim(i) && (n % i == 0) && (m % 2 == 0)) {
            int putere = 1;
            int puterePrimN = pow(i, putere);
            int puterePrimM = pow(i, putere);
            // luam puterea comuna dintre cele doua
            while ( n % puterePrimN == 0 && m % puterePrimM == 0)
```

```
{
                    putere++;
                    puterePrimN = pow(i, putere);
                    puterePrimM = pow(i, putere);
                }
                // Verificam ca pentru ambele, conditia este falsa.
                // Practim verificam cazuri de genul unde x la puterea
(n+1) este divizor prim pentru un numar (gen m) dar nu si pentru un
alt numar dat (gen n)
                if (n % puterePrimN != 0 && m % puterePrimM != 0) {
                    contor++;
                }
            }
        }

        return contor;
    }

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

        for (int i = 2; i * i <= n; i++) {
            if (n % i == 0) {
                return 0;
            }
        }

        return 1;
    }
```

- Ce putem face mai bine (Poti incerca acasa sa vezi ce iese sau daca iti place mai mult ideea asta):
  - Stim ca numaram doar divizorii primi comuni dintre cele doua numere deci putem scurta `for-ul` daca decidem sa ne uitam in divizorii celui mai mic dintre `m` si `n`. (noi am ales direct `n` fara sa luam in considerare acest lucru.)
  - In cazul in care nu am fi avut voie sa folosim functia `pow` din biblioteca `cmath` sau `math.h`, puteam face impartiri succesive atat cu `m` cat si cu `n` la `i` atunci cand acesta este prim. Si practic va trebuie sa vedem ca ele se impart ambele de exact atatea ori la `i`.

3. Subprogramul suma are un singur parametru, n, prin care primește un număr natural (n apartine[1,106]). Subprogramul returnează suma divizorilor pozitivi ai lui n care nu sunt primi. Scrieți definiția completă a subprogramului. Exemplu: pentru n=12 subprogramul returnează 23 (23=1+4+6+12).

- Link: https://modinfo.ro/bac/variante-test-2021/info/v3.pdf
- Solutie:

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

int estePrim(int n);
int suma(int n);

int main() {

    int n = 12;

    cout << suma(n);
    return 0;
}

int suma(int n) {
    int rezultat = 0;
    for (int i = 1; i <= n; i++) {
        if (n% i == 0 && !estePrim(i)){
            rezultat += i;
        }
    }
    return rezultat;
}

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

    for (int i = 2; i * i <= n; i++) {
        if (n % i == 0) {
            return 0;
        }
    }

    return 1;
}
```

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:

```cpp
#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);
    }
}
```

Exercitii de antrenament

- Nota:
  - O buna parte din exercitiile de aici 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
  - 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.

1. 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

2. 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`

3. 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□[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

4. 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

5. 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 *

6. 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.

7. 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.

8. 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`.

9. 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.

10. 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.

Preview sedinta urmatoare

- O buna parte din sedinta urmatoare, daca nu chiar toata, o vom dedica rezolvarii de exercitii in care focusul va fi pe functii.
- Deoarec functiile sunt un topic destul de important la bacalaureat (vor aparea cel putin la partea I si partea III) o sa dedicam mai mult timp lor
- Desi chiar si in sesiunea asta, multe exercitii sunt extrase din variante de bac, de sesiunea urmatoare, toate exercitiile vor fi doar exercitii extrase din variante de bac si aici vom acoperii tot ce am invatat despre functii:
    - functii normale (care intorc sau nu o valoare)
    - functii recursive
    - functii care intorc o valoare prin intermediul parametrilor.

SPORURI!