7. Aplicaţii folosind instrucţiuni în C/C++
(Applications using instructions in C/C++)

1. Obiective:

- Înţelegerea *categoriilor* de instrucţiuni din C/C++
- Înţelegerea *modului* în care lucrează instrucţiunile în C/C++
- Scrierea şi rularea de programe simple în care sunt *folosite* aceste instrucţiuni

1. Objectives:

- Understanding the categories of C/C++ instructions
- Understanding the C/C++ instructions' operating mode
- Writing and running some simple programs that use instructions

2. Breviar teoretic

Categorii de instrucţiuni:

- instrucţiunea compusă (secvenţială)
  - este o secvenţă de [declaraţii şi] instrucţiuni, scrisă între *acolade* {....}

```
{
        listă_de_declaraţii;
        listă_instrucţiuni;
}
```

- instrucţiunea condiţională (-de ramificaţie, -decizională, -alternativă)
  - *if (expresie)*
    instrucţiune
  - *if (expresie)*
    instrucţiune1
    else
    instrucţiune2
- instrucţiuni ciclice
  - repetitivă, codiţionată anterior: sunt două, *while()* şi *for()*

    - instrucţiunea ***while***( ) are următorul format:

```
while (expresie)          // antet
        {
                instrucţiuni;          // corpul buclei
        }
```

unde dacă:

- *expresie* $\neq 0$ (adevărat) atunci se execută *instrucţiuni*, după care se revine la punctul în care se evaluează din nou expresia, corpul executându-se atâta timp cât expresia este adevarată, adică *expresie* $\neq 0$; când *expresie* $= 0$ se trece la următoarea instrucţiune după ciclul ***while***;
- *expresie* $= 0$ de la început, atunci corpul nu se execută nici o dată.

- instrucţiunea ***for( )*** are următorul format:

```
for(exp1;exp2;exp3)
{
        instrucţiuni;
}
```

unde *exp1, exp2, exp3* sunt expresii cu următoarea semnificaţie:

- *exp1*, reprezintă partea de *iniţializare* a ciclului ***for;***

- *exp2*, reprezintă condiţia de *continuare* a ciclului ***for*** (are acelaşi rol cu *expresie* din ciclul ***while***.
- *exp3*, reprezintă partea de *reiniţializare* a ciclului ***for;***

- Biblioteca STL conţine o funcţie template *for_each( )*

- In *C++0x/1y/2z* există acum şi o instrucţiune *for* pentru parcurgerea colecţiilor cu iteratori *(for range based).* Se utilizează pentru prelucrarea tuturor elementelor dintr-un domeniu.

  for (declaratie : expresie_domeniu) instructiuni
  unde *:*
  declaratie*:* declaraţia unei variabile de tipul unui element din domeniu sau o referinţă de acel tip. Deseori utilizează specificatorul ***auto*** pentru deducerea automată a tipului
  expresie_domeniu: secvenţe de elemente cum sunt tablourile, containerele şi alte tipuri, pentru care domeniul este definit de funcţiile begin() şi end()

  - repetitivă condiţionată posterior:

```
do
    {
        instrucţiuni; // se exec. cel putin o data
    }
while(expresie);
```

- instrucţiunea selectivă
  - *switch (expresie)*

```
{
    case c1:    şir_1;
                [break;]
    case c2:    şir_2;
                [break;]
        ...
    case cn:    şir_n;
                [break;]
    [default:   şir]
}
```

  unde:
  - $c_1, \ldots, c_n$ sunt constante ce pot fi: întregi, caracter, element de enumerare.
  - *şir_1, …, şir_n, şir*, sunt secvenţe de instrucţiuni.
- funcţii şi instrucţiuni de salt
  - funcţia ***exit()*** , declarată în <*stdlib.h*>, cu următorul format:
    void exit(int cod); // termină programul în curs de execuţie

  - instrucţiunea ***continue***, cu următorul format:
    continue; //termină iteraţia curentă

  - instrucţiunea ***break***, are următorul format:
    break; //termină bucla curentă, sare la prima instr. după linia finală a buclei

- instrucţiunea ***return***, are următoarele formate:
  - *return;* //revine din funcţia curentă, fără să-i asigneze o valoare. Poate fi “early return”
  - *return expresie;*

- instrucţiunea ***goto***, are următorul format:
  - *goto nume;*

3. Exemple

Exemplul 1: program ce citeşte mai multe numere întregi, le memorează într-un tablou şi apoi le afişează.

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#define MAX     100

int main(void)
{
int i, n, tab[MAX];
     printf("\nCate numere vor fi? : ");
     scanf("%d", &n);
     if(n<=0) {
          printf("\n Date invalide ! (n <0)");
          return 0;
     } // end if

     printf("\nIntroduceti %d numere intregi: ", n);
     for(i=0; i<n; i++) {
          printf("\n\t Astept numarul %d : ", i+1);
          scanf("%d", &tab[i]);
     } // end for

     printf("\nAti introdus urmatoarele numere intregi :\n");
     for(i=0; i<n; i++)
          printf("\t%d\n", tab[i]);
    return 0;
}//end main
```

Exerciţiu: Rescrieţi acelaşi program folosind instrucţiunea do-while.

Exemplul 2: program ce citeşte o cifră din intervalul [1,7] şi afişează denumirea zilei din saptamână, corespunzătoare cifrei respective.

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int  main()
{
int i;
     puts("Tastati o cifra din intervalul [1,7], 1=luni, 2=marti  etc.: ");
     scanf("%d", &i);
     switch(i)
     {
          case 1:
                    puts("luni");
                    break;
          case 2:
                    puts("marti");
                    break;
          case 3:
                    puts("miercuri");
                    break;
          case 4:
                    puts("joi");
                    break;
```

```
        case 5:
                puts("vineri");
                break;
        case 6:
                puts("sambata");
                break;
        case 7:
                puts("duminica");
                break;
        default:
                puts("Cifra tastata nu este in intervalul [1,7]");
        }
  return 0;
}//end main
```

Exemplul 3:Adună valorile date de la tastatură cu confirmarea continuării după fiecare număr introdus

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include<conio.h>

int main()
{
        int c, suma=0;
        int key;

        do
        {
                printf("Introduceti un numar: ");
                scanf("%d",&c);
                suma += c;
                printf("Continuati ? (d/n) \n");
                key =_getch();
        }
        while (!((key == 'n')||(key=='N')));
        printf("Suma numerelor introduse este %d\n", suma);
        return 0;
}//end main
```

Exemplul 4: Programul evaluează dacă un număr introdus este prim sau nu. Optimizati algoritmul.

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int NrPrim(int);

int main(void)
{
int nr;
    printf("\nIntroduceti un numar : ");
    scanf("%d", &nr);
    if(NrPrim(nr) > 0)
        printf("\nNumarul %d este prim\n", nr);
    else
```

```
        printf("\nNumarul %d nu este prim\n", nr);
    return 0;
}// end main

int NrPrim(int n)
// Functia returneaza +1 daca numarul n este prim sau -1 in caz contrar

{
int i;
    if(n <= 1)
        return(-1);
    if(n == 2)
        return(1);
    if(n % 2 ==0)
        return(-1);
    for(i=3; i<n; i+=2) {
        if(n % i == 0)
            return(-1);
    }
    return(1);
}// end fnc. NrPrim
```

Exemplul 5: program care afiseaza valorile dintr-un tablou unidimensional, utilizand instructiunea for() pentru parcurgerea colectiilor.

```
#include <iostream>
using namespace std;
 int main()
{
   int x[10] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
   for( int y : x ) { // acces prin valoare, folosind o copie in varibila cu tip specificat
        cout << y << " ";
   }
   cout << endl;
   for( auto y : x ) { // deducerea tipului, acces prin valoare, copierea valorii
      cout << y << " ";
   }
   cout << endl;
   for( auto &y : x ) { // acces prin referinta, deducerea tipului, cand se doresc modif.
        cout << y << " ";
   }
   cout << endl;
   for( const auto &y : x ) { // acces prin referinta, deducerea tipului, fara modif.
        cout << y << " ";
   }
   cout << endl;
   return 0;
}
```

4. Întrebări:

- Evidenţiaţi diferenţa dintre structura *while* şi *do...while* printr-un exemplu.
- Care este rolul instrucţiunii *break* în cadrul instrucţiunii *switch* ?
- Care este rolul instrucţiunii *break* în cadrul instrucţiunilor ciclice ?
- Care este rolul instrucţunii *continue* în cadrul instrucţiunilor ciclice ?

5. Teme:

1. Se citesc trei numere de la tastatură (a, b şi c). Să se determine aria dreptunghiului ale cărui laturi sunt a şi b şi verificaţi dacă diagonala dreptunghiului este egală cu c.
2. Să se scrie un program care verifică dacă un număr citit de la tastatură este pătrat perfect.
3. Să se scrie un program care calculeaza a^n, unde n este citit de la consolă (a se defineşte în program).
4. Să se scrie un program care citeşte de la tastatură o valoare întreagă şi calculează n! (n-factorial).
5. Să se scrie un program care :
   - determină cel mai mare număr prim mai mic decât numărul dat
   - determină toate numerele prime mai mici decât numărul dat.
6. Să se scrie un program care determină cel mai mare divizor comun a doi întregi.
7. Să se scrie un program care determină toţi divizorii unui număr.
8. Calculaţi produsul a două numere întregi folosind numărul corespunzător de adunări.
9. Să se scrie un program care determină câtul împărţirii a doi întregi folosind scăderi succesive.
10. Să se scrie un program care determină numărul de cifre care compun un număr întreg citit de la tastatură.
11. Să se scrie un program care citeşte de la tastatură n numere întregi. Afişaţi toate numerele impare din şir.
12. Să se citească un număr întreg n de la tastatură.
    Se citesc apoi numere reale, până când suma lor depăşeşte valoarea lui n.
    Să se afişeze suma numerelor citite, cu o precizie de 2 zecimale şi numărul lor (câte s-au introdus).
13. Să se scrie un program care determină cmmmc a două numere citite de la tastatură.
14. Scrieţi un program care citeşte n numere întregi de la tastatură şi le afişează pe cele divizibile cu 3.
15. Să se scrie un program care citeşte de la tastatură un caracter, pe care îl afişează pe n rânduri, câte n caractere pe un rând.
16. Să se scrie o aplicaţie C/C++ în care se introduc de la tastatură numere întregi, până ce utilizatorul apasă tasta <Esc>. Să se determine şi să se afişeze media numerelor impare pozitive.

5'. Homework :

1. Read from the keyboard 3 integer numbers (a, b and c). Determine the area of the rectangle that has the sides equal to a and b and verify if the rectangle's diagonal is equal to c.
2. Please verify if a natural number introduced from the keyboard is a perfect square or not.
3. Write a program that calculates a^n, where a and n are read from the keyboard (a is "hard coded").
4. Write a program that reads from the keyboard an integer value n, and calculates n! (n-factorial).
5. Write a program that:
   - determines the greatest prime number that's smaller than a certain given number;
   - determines all the prime numbers smaller than a given number;
6. Write a program that determines the greatest common divider of 2 integer values read from the keyboard.
7. Write a program that determines all the divisors of a value introduced from the KBD.
8. Calculate the product of 2 integer numbers using additions.
9. Write a program that determines the integer quotient of 2 integer numbers using a series of subtractions.
10. Write a program that determines the number of figures that compose an integer number read from the keyboard.
11. Write a program that reads from the keyboard n integer numbers. Display all the odd numbers and store them in an array.
12. Read from the keyboard an integer number n. Read a series of real numbers, until their sum is greater than n. Display the sum with a 2 digits precision and how many numbers were introduced.
13. Determine the least common multiple of 2 integer numbers read from the keyboard.
14. Write a program that reads n integer numbers from the keyboard and displays those that can be divided by 3.
15. Write a program that reads from the keyboard a single character. The program should display that character on n rows, n times in a row.
16. Write a C/C++ application that reads from the keyboard a series of integer numbers, until the user presses on the <Esc>- key. Determine and display the average value of the odd positive numbers.