

7. Aplicații folosind instrucțiuni în C/C++ (Applications using instructions in C/C++)

1. Obiective:

- Înțelegerea *categoriilor* de instrucțiuni din C/C++
- Înțelegerea *modului* în care lucrează instrucțiunile în C/C++
- Scrierea și rularea de programe simple în care sunt *folosite* aceste instrucțiuni

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 instrucțiuni:

- instrucțiunea compusă (secvențială)
 - este o secvență de [declarații și] instrucțiuni, scrisă între *acolade* {....}

```
{  
    listă_de_declarații;  
    listă_instrucțiuni;  
}
```
- instrucțiunea condițională (-de ramificație, -decizională, -alternativă)
 - *if*(*expresie*)
 instrucțiune
 - *if*(*expresie*)
 instrucțiune1
 else
 instrucțiune2
- instrucțiuni ciclice
 - repetitivă, condiționată anterior: sunt două, *while*() și *for*()
 - instrucțiunea ***while***() are următorul format:

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

unde dacă:

- *expresie* $\neq 0$ (adevărat) atunci se execută *instrucțiuni*, după care se revine la punctul în care se evaluează din nou expresia, corpul executându-se atâta timp cât expresia este adevărată, adică *expresie* $\neq 0$; când *expresie* = 0 se trece la următoarea instrucțiune după ciclul ***while***;
- *expresie* = 0 de la început, atunci corpul nu se execută nici o dată.
- instrucțiunea ***for***() are următorul format:

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

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

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

- *exp2*, reprezintă condiția de *continua*re a ciclului **for** (are același rol cu *expresie* din ciclul **while**).
- *exp3*, reprezintă partea de *reinițializare* a ciclului **for**;
- Biblioteca STL conține o funcție template *for_each()*
- În C++0x/1y/2z există acum și o instrucțiune *for* pentru parcurgerea colecțiilor cu iteratori (*for range based*). Se utilizează pentru prelucrarea tuturor elementelor dintr-un domeniu.

for (declaratie : expresie_domeniu) instructiuni

unde :

declaratie: declarația unei variabile de tipul unui element din domeniu sau o referință de acel tip. Deseori utilizează specificatorul **auto** pentru deducerea automată a tipului

expresie_domeniu: secvențe de elemente cum sunt tablourile, containerele și alte tipuri, pentru care domeniul este definit de funcțiile *begin()* și *end()*

- repetitivă condiționată posterior:

```
do
{
    instructiuni; // se exec. cel puțin o dată
}
while(expresie);
```

- instrucțiunea selectivă

- *switch (expresie)*

```
{
    case c1:    șir_1;
               [break;]
    case c2:    șir_2;
               [break;]
    ...
    case cn:    șir_n;
               [break;]
    [default:   șir]
}
```

unde:

- *c1, ..., cn* sunt constante ce pot fi: întregi, caracter, element de enumerare.
- *șir_1, ..., șir_n, șir*, sunt secvențe de instrucțiuni.

- funcții și instrucțiuni de salt

- funcția **exit()** , declarată în *<stdlib.h>*, cu următorul format:

```
void exit(int cod); // termină programul în curs de execuție
```

- instrucțiunea **continue**, cu următorul format:

```
continue; //termină iterația curentă
```

- instrucțiunea **break**, are următorul format:

```
break; //termină bucla curentă, sare la prima instr. după linia finală a buclei
```

- instrucțiunea **return**, are următoarele formate:

- *return;* //revine din funcția curentă, fără să-i asigneze o valoare. Poate fi “early return”
- *return expresie;*

- instrucțiunea **goto**, are următorul format:

- *goto nume;*

3. Exemple

Exemplul 1: program ce citește mai multe numere întregi, le memorează într-un tablou și apoi le afișează.

```
#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 1;
    } // 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
```

Exercițiu: Rescrieți același program folosind instrucțiunile *while()* si *do-while()*.

Exemplul 2: program ce citește o cifră din intervalul [1,7] și afișează denumirea zilei din săptămână, 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( )
{
    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 si `for()` standard.

```

// #define _CRT_SECURE_NO_WARNINGS
#include <iostream>
using namespace std;

int main( ) {
    int arr[] = { -1, 20, 3, 4 }; // tablou unidim., init. la decl.
    printf("Tabloul original are %d elemente.\nEle sunt:\n", _countof(arr));

    for (int i : arr) { // acces prin valoare, folosind o copie in iteratorul i cu tip specificat
        cout << i << " "; // i -- iterator
    }

    cout << "\nModificam toate valorile!\n";
    for (auto& i : arr) { // acces prin referinta, deducerea tipului, cand se doresc modif.
        cout << "Valoare elementului actual i = " << i << "\t introdu noua val.: ";
        cin >> i; // scanf("%d", &i); // ambele merg la fel, scanf() portabil cere #define _CRT_SECURE....
    }

    cout << "\nAfisam din nou, cu for( ) standard, utilizand operatorul sizeof( ).\n";
    for (int i = 0; i < sizeof(arr) / sizeof(int); i++) // standard for
        cout << arr[i] << ' ';

    cout << "\n... si folosind functia de biblioteca _countof( )" << '\n';
    for (int i = 0; i < _countof(arr); i++) // standard for
        cout << arr[i] << ' ';
    cout << '\n';

    cout << "\nInitializare generalizata tablou\n";
    for (int i : { -1, 1, 3, 5, 7 }) // the initializer mai be a braced-init-list
        cout << i << ' ';
    cout << '\n';
}

```

```

cout << " \nputem sa nu utilizam valoarea variabilei de ciclare...\n";
for (int i : arr)
    cout << 7 << ' ';

cout << "\ntablou de pointeri, la siruri de caractere constante\n";
const char* cuvinte[ ] = { "alfa", "beta", "gama" };
for (const char* cuvint : cuvinte)
    cout << cuvint << '\t';
cout << "\nBy! \n";

} // main

```

4. Întrebări:

- Evidențiați diferența dintre structura *while* și *do...while* printr-un exemplu.
- Care este rolul instrucțiunii *break* în cadrul instrucțiunii *switch* ?
- Care este rolul instrucțiunii *break* în cadrul instrucțiunilor ciclice ?
- Care este rolul instrucțiunii *continue* în cadrul instrucțiunilor ciclice ?

5. Teme:

1. Se citesc trei numere de la tastatură (a, b și c). Să se determine aria dreptunghiului ale cărui laturi sunt a și b și verificați 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 calculează a^n , prin înmulțiri repetate, unde n este citit de la consolă (a se definește în program).
4. Să se scrie un program care citește de la tastatură o valoare întreagă n și 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 toți divizorii unui număr.
8. Calculați produsul a două numere întregi folosind numărul corespunzător de adunări.
9. Să se scrie un program care determină câtul împărțirii 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 citește de la tastatură n numere întregi. Afișați toate numerele impare din șir.
12. Să se citească un număr întreg n de la tastatură.
Se citesc apoi numere reale, până când suma lor depășește valoarea lui n.
Să se afișeze suma numerelor citite, cu o precizie de 2 zecimale și numărul lor (câte s-au introdus).
13. Să se scrie un program care determină cmmmc a două numere citite de la tastatură.
14. Scrieți un program care citește n numere întregi de la tastatură și le afișează pe cele divizibile cu 3.
15. Să se scrie un program care citește de la tastatură un caracter, pe care îl afișează pe n rânduri, câte n caractere pe un rând, n citit de la tastatură.
16. Să se scrie o aplicație C/C++ în care se introduc de la tastatură numere întregi, până ce utilizatorul apasă tasta <Esc>. Să se determine și să se afișeze 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 , by repeated multiplications where n is 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, n read from the keyboard.
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.