Facultatea: **Automatică şi Calculatoare**, Iaşi
Specializarea: **Calculatoare, Tehnologia informaţiei**

PROIECTAREA ALGORITMILOR

Lucrarea nr. 1
Recapitularea unor noţiuni C/C++

Breviar teoretic:

Structura unui program simplu in C/C++:

```
//includere de fisiere header
#include <stdio.h>

//functia main = punctul de „intrare” in program
int main (void) {
      //declarare de variabila
      int x;

      //apel de functie
      printf (”Hello world!\n”);

      //atribuire valoare variabila
      x = 10;

      printf (”x = %d\n”, x);
      //finalizarea unui program
      return 0;
}
```

Variabile:

- mecanism ce simplifică stocarea informaţiilor în cadrul unui program;
- **variabilă** este un nume simbolic asociat cu o anumită **locaţie de memorie**;
- **numele unei variabile** *începe* fie cu un caracter ***alfabetic*** fie cu caracterul _ (***underscore***) şi *nu trebuie să conţină spaţii albe sau alte caracter în afara celor alfa-numerice*;
- **variabilă** poate fi accesată (citită) şi modificată (scrisă);
- variabilă se *caracterizează* prin: **tip**, **nume** şi **valoare**;
- **variabile locale** – variabile definite în interiorul unei funcţii;
- **variabile globale** – variabile definite în exteriorul oricărei funcţii dintr-un fişier sursă;
- variabilă este ***vizibilă*** din momentul în care a fost definită până la sfârşitul unei funcţii (dacă este variabilă locală) sau până la sfârşitul fişierului sursă (dacă este variabilă globală).

Tipurile de date fundamentale:

INT:

- tip de dată utilizat pentru stocarea **numerelor întregi**, **pozitive** şi **negative**, **reprezentate în cod complementar faţă de 2**;
- dimensiune: **4 octeţi**.

CHAR:

- tip de dată utilizat pentru stocarea **caracterelor**:
- dimensiune: **1 octet**.

FLOAT:

- tip de dată utilizat pentru stocarea **numerelor reale**, **pozitive** şi **negative**, **reprezentate în format cu virgulă mobilă**, **în simplă precizie**;
- dimensiune: **4 octeţi**.

DOUBLE:

- tip de dată utilizat pentru stocarea **numerelor reale**, **pozitive** şi **negative**, **reprezentate în format cu virgulă mobilă**, **în dublă precizie**;
- dimensiune: **8 octeţi**.

Tipuri de date definite utilizator – STRUCTURI:

Declarare:

```
struct nume_structura {
      //lista membri structura
      tip_data1 membru1, membru2;
      tip_data2 membru3;
      …
};
```

Declarea unei variabile de tip structură:

```
struct nume_structura nume_variabila;
```

Funcţii:

- **funcţie** reprezintă o secvenţă de cod destinată să implementeze o anumită *sarcină*;
- **funcţie** se declară prin ***antet***:

```
tip_retur nume_funcţie (listă de parametri formali);
```

- **funcţie** se ***defineşte*** prin implementarea corpului funcţiei (marcat prin *{}*).

Pointeri:

- un **pointer** reprezintă în C/C++ un tip de variabilă ce *indică* o adresă;
- declarare unui pointer:

```
T * nume_pointer; // T reprezintă un tip de dată primar
                  // sau definit de utilizator
```

- ***iniţializarea*** pointerilor:

cu ***valoare 0***: `T* p = 0`; sau `T* p = NULL;`

cu ***adresa unei variabile*** declarate anterior: `T* p = &(var);`
cu ***valoarea*** unui alt pointer declarat şi iniţializat anterior: `T* p = ptr;`
prin ***alocare dinamică de memorie***:

```
T* p = (T*) malloc (no_elems * sizeof(T));
```

s***au***

```
T* p = (T*) calloc (no_elems, sizeof (T));
```

Sarcini de lucru:

1. Scrieți un program C/C++ care să permită inserarea unei valori pe o poziţie specificată, într-o *listă liniară dublu înlănţuită*.

2. Se dă o matrice pătratică a[n][n] cu elemente formate din caractere din alfabetul latin. Să se rotească elementele matricei în sens trigonometric, folosind spaţiu suplimentar de memorie constant.

Barem de notare:

1. P1: 4p
2. P2 5p
3. Baza: 1p