

UNITATEA DE ÎNVĂȚARE 1: NOȚIUNI LEGATE DE ALGORITMICĂ, PROGRAMARE STRUCTURATĂ ȘI SCHEME LOGICE

1.5. Răspunsuri la testele de autoevaluare

Testul de autoevaluare 1:

1. Ce este un algoritm?

Un algoritm este o secvență finită de pași, care durează un interval de timp finit, ce rezolvă o problemă și întoarce un rezultat.

2. Care sunt părțile componente ale unui algoritm?

Părțile fundamentale ale unui algoritm sunt:

- a. Date de intrare;
- b. Comenzi;
- c. Date de ieșire.

3. Ce este o schemă logică?

O schemă logică este un desen care cuprinde poligoane și săgeți și care, citite în mod corect, duc la înțelegerea algoritmului pe care îl reprezintă.

Testul de autoevaluare 2:

1. Ce este o variabilă?

O variabilă este o zonă de memorie care a primit o denumire și ce reține una sau mai multe valori.

2. Ce este o variabilă simplă?

O variabilă simplă este o variabilă ce reține o singură valoare.

3. Ce este o variabilă structurată

O variabilă structurată este o variabilă ce reține două sau mai multe valori.

4. Care sunt operațiile care se pot face cu o variabilă?

Operațiile care se pot efectua asupra unei variabile sunt:

- a. Scrierea valorii unei variabile;
- b. Atribuirea unei valori către o variabilă;
- c. Citirea valorii unei variabile.

Testul de autoevaluare 3:

1. Ce este programarea structurată?

Programarea structurată este un stil de programare care se bazează pe împărțirea pașilor unui algoritm, respectiv a instrucțiunilor unui program în patru modele numite *structuri*.

2. Care sunt structurile de bază ale programării structurate?

Structurile de bază ale programării structurate sunt:

- a. Structura secvențială;
- b. Structura de decizie;
- c. Structura de selecție;
- d. Structura iterativă.

Testul de autoevaluare 4:

1. Care este semnificația blocului de intrare a datelor dintr-o schemă logică?

Semnificația blocului de intrare a datelor este aceea de a marca operațiile de introducere a datelor.

2. Care este semnificația blocului de decizie dintr-o schemă logică?

Semnificația blocului de decizie este de a permite ramificarea algoritmului în raport cu îndeplinirea unei condiții sau condițiilor scrise în interiorul rombului.

3. Care este blocul din care pleacă întotdeauna săgețile într-o schemă logică?

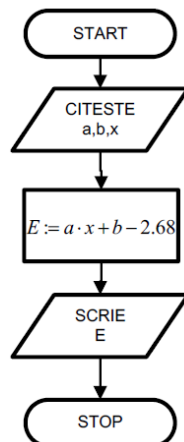
Blocul din care pleacă întotdeauna săgețile într-o schemă logică este blocul START.

Exemplul 1.

Fie $a, b, x \in \mathbb{R}$. Să se întocmească schema logică pentru evaluarea expresiei aritmetice:

$$E = a * x + b - 2,68$$

```
1 #include <iostream> // Include biblioteca standard pentru intrare/ieșire
2
3 int main() {
4     // Declarăm variabilele
5     double a, b, x, E;
6
7     // Citim datele de intrare
8     std::cout << "Introdu valoarea pentru a: ";
9     std::cin >> a;
10
11     std::cout << "Introdu valoarea pentru b: ";
12     std::cin >> b;
13
14     std::cout << "Introdu valoarea pentru x: ";
15     std::cin >> x;
16
17     // Calculăm expresia E = a * x + b - 2.68
18     E = a * x + b - 2.68;
19
20     // Afișăm rezultatul
21     std::cout << "Valoarea lui E este: " << E << std::endl;
22
23
24     return 0;
25 }
26
```



1.4. Rezumat

- Un program pe calculator, la bază, are rolul de a rezolva o problemă, de a satisface o nevoie;
- Orice problemă se rezolvă prin executarea unei serii de pași, care formează un *algorithm*;
- Un algorithm are trei părți componente fundamentale:
 - a) Datele de intrare;
 - b) Comenzile;
 - c) Datele de ieșire.
- O *variabilă* este un concept fundamental în programare și este utilizată foarte des în cadrul unui algorithm;
- O variabilă este o zonă de memorie ce reține o valoare și care are primit un nume;
- Există două tipuri de variabile:
 - a) Variabile simple; (cum ar fi int, float, double, char, bool)
 - b) Variabile structurate. (cum ar fi array, struct)
- *Programarea structurată* este un stil de programare ce se bazează pe utilizarea a patru structuri fundamentale pentru scrierea programelor;
- Structurile fundamentale utilizate de programarea structurată sunt:
 - a) Structura secvențială;
 - b) Structura de decizie;
 - c) Structura de selecție;
 - d) Structura iterativă;
- *Schema logică* este o modalitate grafică de reprezentare a unui algorithm, ce folosește o serie de poligoane și săgeți care au anumite semnificații.

Structurile de bază ale programării structurate

În programarea structurată, există patru structuri fundamentale care permit organizarea logică a fluxului de control al unui program: **structura secvențială**, **structura de decizie**, **structura de selecție** și **structura iterativă**. Iată exemple pentru fiecare dintre ele în C++:

1. Structura secvențială:

Instrucțiunile sunt executate una după alta, în ordine.

Explicație: Instrucțiunile sunt executate secvențial: inițializăm variabilele *a* și *b*, calculăm suma lor și apoi afișăm rezultatul.



```
1 #include <iostream>
2
3 int main() {
4     int a = 5;        // Inițializare variabilă
5     int b = 10;       // Inițializare variabilă
6     int suma = a + b; // Calculăm suma
7
8     std::cout << "Suma este: " << suma << std::endl; // Afișăm rezultatul
9
10    return 0;
11 }
12
```

2. Structura de decizie (if-else):

Permite alegerea unei ramuri de cod în funcție de o condiție.

Explicație: Dacă condiția `numar % 2 == 0` este adevărată, se execută ramura care afișează că numărul este par. Dacă nu, se execută ramura `else`, care afișează că numărul este impar.

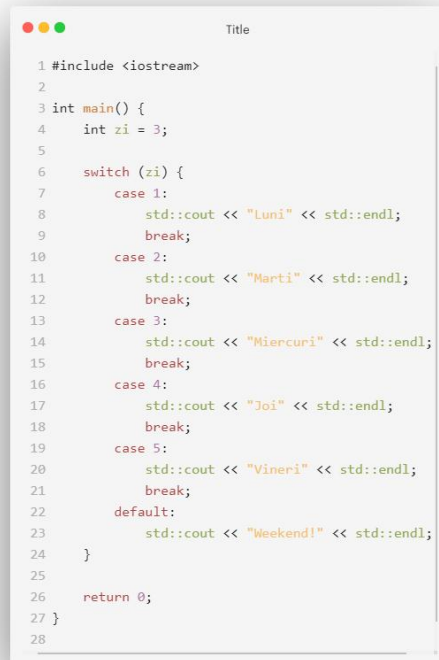


```
1 #include <iostream>
2
3 int main() {
4     int numar = 7;
5
6     if (numar % 2 == 0) { // Verificăm dacă numărul este par
7         std::cout << numar << " este par." << std::endl;
8     } else {
9         std::cout << numar << " este impar." << std::endl;
10    }
11
12    return 0;
13 }
14
```

3. Structura de selecție (switch-case):

Este utilizată pentru a selecta una dintre mai multe căi posibile de execuție pe baza valorii unei variabile.

Explicație: În funcție de valoarea variabilei `zi`, se execută unul dintre blocurile de cod corespunzătoare fiecărui `case`. Dacă niciun `case` nu se potrivește, se execută ramura `default`.



```
1 #include <iostream>
2
3 int main() {
4     int zi = 3;
5
6     switch (zi) {
7         case 1:
8             std::cout << "Luni" << std::endl;
9             break;
10        case 2:
11            std::cout << "Marti" << std::endl;
12            break;
13        case 3:
14            std::cout << "Miercuri" << std::endl;
15            break;
16        case 4:
17            std::cout << "Joi" << std::endl;
18            break;
19        case 5:
20            std::cout << "Vineri" << std::endl;
21            break;
22        default:
23            std::cout << "Weekend!" << std::endl;
24    }
25
26    return 0;
27 }
28
```

4. Structura iterativă (while, for):

Permite repetarea unei secvențe de instrucțiuni până când o condiție este îndeplinită.

Exemplu (while loop)



```
1 #include <iostream>
2
3 int main() {
4     int i = 1;
5
6     while (i <= 5) { // Repetăm cât timp i este mai mic sau egal cu 5
7         std::cout << "i = " << i << std::endl;
8         i++; // Incrementează i
9     }
10
11     return 0;
12 }
13
```

Exemplu (for loop):



```
1 #include <iostream>
2
3 int main() {
4     for (int i = 1; i <= 5; i++) { // Inițializare; Condiție; Incrementare
5         std::cout << "i = " << i << std::endl;
6     }
7
8     return 0;
9 }
10
```

Explicație:

- În primul exemplu, bucla `while` continuă să se execute cât timp condiția `i <= 5` este adevărată.
- În al doilea exemplu, bucla `for` iterează de la 1 la 5, afișând valoarea lui `i` la fiecare pas.