

Introducere în C/C++ ,tipuri variabile , conversii, switch

1. Introducere în Bazele Limbajului C/C++

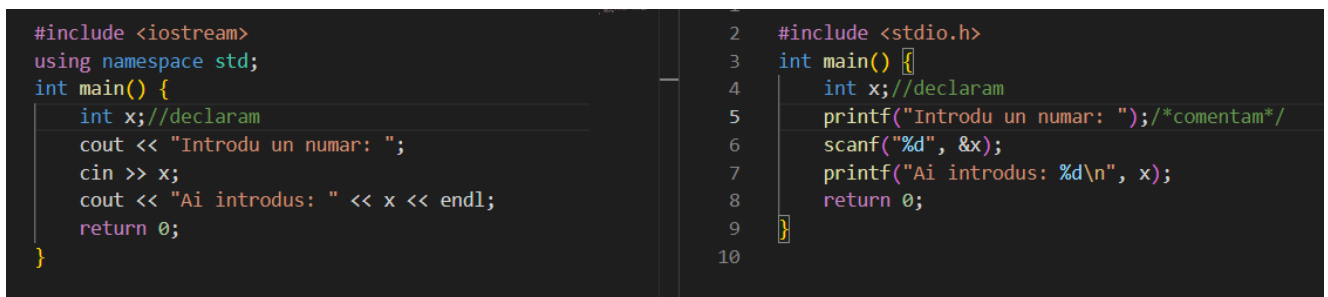
Ce este C/C++?

C este un limbaj de programare procedural, creat în anii '70, folosit pentru dezvoltarea de sisteme de operare și aplicații cu performanță ridicată.

- Este utilizat în dezvoltarea nucleului sistemului de operare Linux, în programarea microcontrolerelor și în aplicații critice de performanță.
- Limbajul C rămâne esențial în dezvoltarea de drivere, sisteme integrate și aplicații unde controlul direct al memoriei este necesar.

C++ este o extensie a limbajului C, care introduce programarea orientată pe obiecte (OOP) și alte caracteristici moderne.

- Este utilizat pe scară largă în dezvoltarea de aplicații complexe, cum ar fi jocuri video, aplicații pentru desktop, software de sistem și platforme financiare.
- În prezent, C++ este esențial în dezvoltarea aplicațiilor de realitate virtuală (VR), a motoarelor de jocuri precum Unreal Engine și a sistemelor de înaltă performanță în industria financiară.



```
#include <iostream>
using namespace std;
int main() {
    int x; //declaram
    cout << "Introdu un numar: ";
    cin >> x;
    cout << "Ai introdus: " << x << endl;
    return 0;
}
```

```
2  #include <stdio.h>
3  int main() {
4      int x; //declaram
5      printf("Introdu un numar: "); /*comentam*/
6      scanf("%d", &x);
7      printf("Ai introdus: %d\n", x);
8      return 0;
9  }
10
```

Figura 1. C++ vs C

2. Tipuri de variabile

Tip de variabilă	Descriere	Exemplu
int	Numere întregi	int varsta = 25;
float	Numere reale cu precizie simplă	float pi = 3.14;
double	Numere reale cu precizie dublă	double e = 2.71;
char	Caractere individuale	char litera = 'A';
bool	Valori logice true/false	bool esteAdevarat = true;
string (C++)	Șiruri de caractere	string mesaj = "Salut";
void	Fără tip de date (funcții fără return)	void afisare();
short	Numere întregi scurte	short x = 32767;
long	Numere întregi lungi	long y = 9223372036854775807;
long long	Numere întregi foarte lungi	long long z = 9223372036854775807LL;
Tabelul 1. Variabilele utilizate		

```
3  int varsta = 25;
4  float inaltime = 1.75;
5  char initiala = 'A';
6  bool esteStudent = true;
7  string mesaj = "Salut";
```

Figura 2. Exemplu de declarație

- Specificatorii de Format în C

În limbajul C, specificatorii de format sunt folosiți în principal în funcțiile de intrare-ieșire, cum ar fi printf(), scanf(), sprintf(), fprintf(), etc., pentru a afișa sau citi date de diferite tipuri. Acești specificatori sunt precedați de caracterul % și sunt urmați de una sau mai multe litere sau simboluri care indică tipul datelor.

%[flags][width][.precision][length]specifier -SINTAXA GENERALĂ

☐ **Flags (Opțiuni)**

Modifică modul de afișare a valorilor.

- - : Aliniere la stânga (left-align)
- + : Afișează semnul pentru numerele pozitive (+ sau -)
- space : Lasă un spațiu înaintea numerelor pozitive
- 0 : Completează cu zero-uri în loc de spații
- # : Activează formatarea alternativă (ex. 0x pentru hexazecimal)

☐ **Width (Lățimea câmpului)**

Specifică lățimea minimă a câmpului în care este afișată valoarea.

Ex.: %5d → va afișa numărul în minim 5 caractere.

☐ **Precision (Precizie)**

- Pentru numere întregi: definește numărul minim de cifre afișate.

- Pentru numere în virgulă mobilă: definește numărul de zecimale.
- Pentru șiruri: definește numărul maxim de caractere afișate.

Ex.: %6.2f → număr cu 6 caractere în total și 2 zecimale.

☐ **Length (Modificatori de lungime)**

- h : short
- l : long
- ll : long long
- L : long double

☐ **Specifiers (Specificieri de tip)**

Specificator	Descriere	Exemplu
%d/%i	întreg (int)	printf("%d", 123);
%u	unsigned int	printf("%u", 123U);
%f	float/double	printf("%.2f", 3.14159);
%e/%E	notație științifică	printf("%e", 123.45);
%g/%G	formă compactă (f sau e)	printf("%g", 123.45);
%x/%X	hexazecimal (lower/upper)	printf("%x", 255);
%o	octal	printf("%o", 255);
%c	caracter	printf("%c", 'A');
%s	șir de caractere	printf("%s", "Hello");
%p	pointer	printf("%p", ptr);
%%	caracterul %	printf("%%");

Tabel 2. Specificatorii din c

3. Biblioteci Standard în C și C++

Bibliotecile standard sunt colecții de funcții și clase care facilitează scrierea codului, oferind funcționalități predefinite pentru diverse operațiuni comune. Acestea se includ în cod cu ajutorul directivelor `#include`.

Bibliotecă	Limbaaj	Funcționalitate	Exemplu
stdio.h	C	Operații de intrare/ieșire standard (printf, scanf)	<code>#include <stdio.h></code>
iostream	C++	Operații de intrare/ieșire standard (cin, cout)	<code>#include <iostream></code>
math.h	C/C++	Funcții matematice (sqrt, pow, sin, cos)	<code>#include <math.h></code>
string.h	C	Funcții pentru manipularea șirurilor de caractere (strlen, strcpy)	<code>#include <string.h></code>
string	C++	Clasa și metodele pentru șiruri de caractere (length, append)	<code>#include <string></code>
stdlib.h	C/C++	Funcții diverse: alocare memorie (malloc, free), conversii numerice	<code>#include <stdlib.h></code>
vector	C++	Clasa vector pentru liste dinamice	<code>#include <vector></code>

Tabelul 3. Biblioteci utilizate.

De lucru:

1. Conversii între date

În C și C++ conversiile între tipuri se pot realiza fie implicit, când compilatorul convertește automat, de exemplu, un int într-un float în condițiile în care nu apare o pierdere semnificativă de informație, fie explicit, prin cast (de exemplu, (int)valoare_float), necesar atunci când transformarea de la un tip cu o reprezentare mai mare (precum double) la unul cu o reprezentare mai mică (precum float sau int) poate conduce la trunchierea valorii sau pierderea părții fracționale.

```
#include <stdio.h>

int main() {
    int a = 10;
    float b = 5.5;
    char c = 'A';

    printf("Int: %d, Float: %f, Char: %c\n", a, b, c);

    // Conversie float -> int
    int d = (int) b;
    printf("Conversie float -> int: %d\n", d);

    return 0;
}
```

Figura 3. Exemplu conversia explicită

1.1. Scrie un program care citește un caracter și îl transformă în cod ASCII.

```
// Citirea caracterului
printf("Introduceți un caracter: ");
scanf(" %c", &caracter); |

// Afișarea codului ASCII
printf("Codul ASCII al caracterului '%c' este: %d\n", caracter, (int)caracter);

return 0;
```

Figura 4. Conversie cod ASCII

- *SizeOf*

Operatorul sizeof este un instrument esențial în C și C++ care determină mărimea, în bytes, a unui tip de date sau a unei expresii. Acesta returnează numărul de bytes pe care o variabilă, un tip sau o structură îl ocupă în memorie. De exemplu, sizeof(int) poate returna 4, indicând că un int ocupă 4 bytes (deși aceasta poate varia în funcție de arhitectură).

- **Sintaxă:**

- Pentru un tip de date: *sizeof(type)* - Parantezele sunt obligatorii când specifici un tip.
- Pentru o variabilă sau o expresie: *sizeof variabila*

1.2 To Do (C++)

- **Declarați și inițializați 5 variabile** de tipuri diferite și afișați-le.
- **Conversii între tipuri** (ex: double -> int, char -> int, int -> char).
- **Afișați dimensiunea fiecărei variabile** folosind sizeof().


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

int main() {
    // Declaraire și inițializare de variabile de tipuri diferite
    int i = 65;
    float f = 3.14f;
    double d = 9.99;
    char c = 'Z';
    bool b = true;

    cout << "Valori inițiale:" << endl;
    cout << "int i = " << i << endl;
    cout << "float f = " << f << endl;
    cout << "double d = " << d << endl;
    cout << "char c = " << c << endl;
    cout << "bool b = " << boolalpha << b << endl; // boolalpha pentru a afișa
true/false

    // 1. Conversie double -> int (trunchierea valorii)
    int conv_d = (int)d;
    cout << "\nConversie double -> int: " << conv_d
        << " (din double " << d << ")" << endl;

    // 2. Conversie char -> int (obținerea codului ASCII)
    int conv_c = (int)c;
    cout << "Conversie char -> int: " << conv_c
        << " (din char '" << c << "')" << endl;

    // 3. Conversie int -> char (obținerea caracterului corespunzător)
    char conv_i = (char)i;
    cout << "Conversie int -> char: " << conv_i
        << " (din int " << i << ")" << endl;

    // Afișarea dimensiunii fiecărei variabile folosind sizeof()
    cout << "\nDimensiuni:" << endl;
    cout << "Dimensiunea unui int: " << sizeof(i) << " bytes" << endl;
    cout << "Dimensiunea unui float: " << sizeof(f) << " bytes" << endl;
    cout << "Dimensiunea unui double: " << sizeof(d) << " bytes" << endl;
```

```
cout << "Dimensiunea unui char: " << sizeof(c) << " bytes" << endl;
cout << "Dimensiunea unui bool: " << sizeof(b) << " bytes" << endl;

return 0;
}
```

2. Scrieți un program care cere utilizatorului să introducă o valoare întreagă pentru lungimea unui dreptunghi și o variabilă pentru latimea acestuia și apoi afișează aria și perimetrul dreptunghiului corespunzător în C.

Rezolvare:

- Se vor declara variabilele necesare: *lungime*, *latime* pentru cele două dimensiuni și *a*, *p* pentru aria și respectiv perimetrul dreptunghiului
- Se vor citi celor două variabile, *lungime* și *latime*
- Se va calcula aria și perimetrul dreptunghiului

```
#include <stdio.h>

int main() {
    int lungime, latime, a, p;

    printf("Introduceti lungimea dreptunghiului: ");
    scanf("%d", &lungime);
    printf("Introduceti latimea dreptunghiului: ");
    scanf("%d", &latime);

    a = lungime * latime;
    p = 2 * (lungime + latime);

    printf("Aria dreptunghiului: %d\n", a);
    printf("Perimetrul dreptunghiului: %d\n", p);

    return 0;
}
```

Figura 8. Rezolvare în C

3. Scrieți un program în C++ care cere utilizatorului să introducă o valoare pentru temperatura măsurată în grade Fahrenheit (F) și apoi realizează conversia către grade Celsius (C) folosind formula:

$$C = \frac{5}{9} * (F - 32)$$

Rezolvare:

- Se vor declara variabilele necesare: f si c de tip float;
- Se va citi numarul de grade in Fahrenheit;
- Se va face conversia in Celsius conform formulei

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

int main() {
    // Declararea variabilelor necesare
    float f, c;

    // Citirea temperaturii în grade Fahrenheit
    cout << "Introduceti temperatura in grade Fahrenheit: ";
    cin >> f;

    // Conversia în grade Celsius
    c = (5.0 / 9.0) * (f - 32);

    // Afișarea rezultatului
    cout << "Temperatura în grade Celsius este: " << c << "°C" << endl;

    return 0;
}
```

Figura 9. Rezolvare în C++

Structura switch

În limbajele de programare C și C++, structura switch este o instrucțiune de control utilizată pentru a executa diferite blocuri de cod în funcție de valoarea unei expresii. Este o alternativă mai clară și mai eficientă decât utilizarea mai multor instrucțiuni if-else-if, atunci când trebuie să comparăm o variabilă cu mai multe valori posibile.

```
switch (expresie) {  
    case valoare1:  
        // cod de executat dacă expresia == valoare1  
        break;  
    case valoare2:  
        // cod de executat dacă expresia == valoare2  
        break;  
    // mai multe cazuri după nevoie  
    default:  
        // cod de executat dacă expresia nu se potrivește cu niciun caz  
        break;  
}
```

4. Programul va solicita două numere întregi, va permite alegerea unei operații de la tastatură și va afișa rezultatul în funcție de alegerea utilizatorului în C.

Operațiile disponibile sunt:

+ pentru sumă

- pentru diferență

*** pentru produs**

/ pentru catul împărțirii (doar dacă $b > 0$)

% pentru restul împărțirii (doar dacă $b > 0$)

M pentru maximul dintre cele două numere

```
#include <stdio.h>

int main() {
    int a, b;
    char op;

    // Citirea celor două numere întregi
    printf("Introduceti doua numere intregi (a si b): ");
    scanf("%d %d", &a, &b);

    // Afișarea opțiunilor disponibile
    printf("Alegeti operatia dorita:\n");
    printf("+ : Suma\n");
    printf("- : Diferenta\n");
    printf("* : Produsul\n");
    printf("/ : Catul impartirii (daca b > 0)\n");
    printf("%% : Restul impartirii (daca b > 0)\n");
    printf("M : Maximul dintre cele doua numere\n");

    // Citirea operatorului dorit
    printf("Introduceti operatorul: ");
    scanf(" %c", &op);

    // Structura switch pentru efectuarea calculelor
    switch (op) {
        case '+':
            printf("Suma: %d\n", a + b);
            break;

        case '-':
            printf("Diferenta: %d\n", a - b);
            break;
```

```
case '*':  
    printf("Produsul: %d\n", a * b);  
    break;  
  
case '/':  
    if (b != 0) {  
        printf("Catul impartirii: %d\n", a / b);  
    } else {  
        printf("Impartirea la zero nu este permisa.\n");  
    }  
    break;  
  
case '%':  
    if (b != 0) {  
        printf("Restul impartirii: %d\n", a % b);  
    } else {  
        printf("Impartirea la zero nu este permisa.\n");  
    }  
    break;  
  
case 'M':  
    if (a > b) {  
        printf("Maximul este: %d\n", a);  
    } else if (b > a) {  
        printf("Maximul este: %d\n", b);  
    } else {  
        printf("Numerele sunt egale: %d\n", a);  
    }  
    break;  
  
default:
```

```
printf("Operator invalid!\n");  
break;  
}  
  
return 0;  
}
```

Figura 10. Switch în C

5. Scrie un program care cere de la utilizator o **notă (1-10)** și afișează un mesaj corespunzător (ex. 10 = "Excelent", 5 = "Satisfăcător", etc.) în C++.

6. Calculator de reduceri în C

Scrie un program care cere de la utilizator **prețul unui produs** și un **cod de reducere (1, 2 sau 3)**.

- **Cod 1:** 10% reducere
- **Cod 2:** 20% reducere
- **Cod 3:** 30% reducere
- Dacă introducerea este invalidă, afișează un mesaj de eroare.

7. Scrie un program în C care cere de la utilizator **salariul brut** și tipul contractului:

- **Cod 1:** Salariu pentru angajat full-time (**25% impozit**)
- **Cod 2:** Salariu pentru contract de colaborare (**10% impozit**)
- **Cod 3:** Salariu pentru PFA (**5% impozit**)

- Dacă codul introdus este invalid, afișează un mesaj de eroare.

Formula: $Salariu\ net = Salariu\ brut - (Salariu\ brut \times impozit)$