Fundamentele Programării

Structura unui program C++

Structura unui program C++

- **Componentele de baza**: cuvintele cheie, identificatori, semnele de punctuatie, sintaxa
- **Directivele preprocesare** (sunt introduse de “#”)
- Functia principala ***`main()`*** si alte functii
- **Spatiile de nume**
- **Comentariile**
- Operatiile de **citire/ scriere** (sau **intrare/iesire** sau **input/output**). Identificatiorii **cin** si **cout.**

Cuvintele cheie ale C++

- Cuvintele cheie fac parte din **vocabularul limbajului de programare/ sintaxa limbajului.**

- Programatorul nu poate folosi aceste cuvinte decat cu scopul pentru care au fost ele create

- C++ ~ 80 de cuvinte cheie
- Java ~ 68 de cuvinte cheie
- C ~ 54 de cuvinte cheie
- Pyton ~ 35 de cuvinte cheie

Cuvintele cheie ale C++

- https://en.cppreference.com/w/cpp/keyword

C++ keywords

This is a list of reserved keywords in C++. Since they are used by the language, these keywords are not available for re-definition or overloading.

A - C	D - P	R - Z
alignas (since C++11)	decltype (since C++11)	reflexpr (reflection TS)
alignof (since C++11)	default (1)	register (2)
and	delete (1)	reinterpret_cast
and_eq	do	requires (since C++20)
asm	double	return
atomic_cancel (TM TS)	dynamic_cast	short
atomic_commit (TM TS)	else	signed
atomic_noexcept (TM TS)	enum	sizeof (1)
auto (1)	explicit	static
bitand	export (1) (3)	static_assert (since C++11)
bitor	extern (1)	static_cast
bool	false	struct (1)
break	float	switch
case	for	synchronized (TM TS)
catch	friend	template
char	goto	this
char8_t (since C++20)	if	thread_local (since C++11)
char16_t (since C++11)	inline (1)	throw
char32_t (since C++11)	int	true
class (1)	long	try
compl	mutable (1)	typedef
concept (since C++20)	namespace	typeid
const	new	typename
consteval (since C++20)	noexcept (since C++11)	union
constexpr (since C++11)	not	unsigned
constinit (since C++20)	not_eq	using (1)
const_cast	nullptr (since C++11)	virtual
continue	operator	void
co_await (since C++20)	or	volatile
co_return (since C++20)	or_eq	wchar_t
co_yield (since C++20)	private	while
	protected	xor
	public	xor_eq

Identificatori ai C++

- **Identificatori definiti de utilizator**
- **Identificatori predefiniti** (`main`, `cout`, `std`, `endl`, s.a) -- au o semnificatie rezervata

- **Identificatorii definiti de utilizator** sunt nume alfanumerice folosite în C++ pentru a **identifica (denumi) elementele programului**, cum ar fi variabilele, constantele, funcțiile, clasele, obiectele, tipurile de date, spațiile de nume etc.
- Cu alte cuvinte, un **identificator** este **numele atribuit unei entități** din program, prin care aceasta poate fi recunoscută și utilizată ulterior în cod.

- **Reguli pentru identificatorii definiti de utilizator:**

❖ Pot conține litere (A–Z, a–z), cifre (0–9) și underscore (_).
❖ Primul caracter nu poate fi o cifră.
❖ Sunt sensibili la majuscule/minuscule (C++ este case-sensitive).
❖ Nu pot fi cuvinte cheie rezervate (ex. `if`, `class`, `for` etc.).
❖ Nu trebuie să coincidă cu identificatori predefiniți din bibliotecile standard, pentru a evita confuziile.

- **Identificatorii predefiniți** sunt *nume de entități (funcții, clase, obiecte, constante, spații de nume etc.) definite deja în biblioteca standard*.
- Ei nu fac parte din sintaxa limbajului propriu-zis (ca „`if`” sau „`while`”), dar sunt **gata definiți** pentru utilizatori, astfel încât sa ii putem folosi fără să îi declaram.

Identificatori ai C++

```cpp
#include <iostream>

int main() {

    int numarFavorit;
    std::cout << "Introduceti numarul vostru favorit cuprins intre 1 si 100: ";
    std::cin >> numarFavorit;
    std::cout << "Surprinzator, acesta este si numarul meu favorit!"<<std::endl;
    return 0;
}
```

Operatori ai C++

- Operatori artimetici: +, x, -, /, %
- Operatori logici ! (negatia), || (disjunctia), && (conjunctia)
- Operatori relationari: <, =, >, <=, >=, ==, !=
- Operatori pe biti &, |, ^,~
- Operatorul de conversie explicita
- Operatorul sizeof

- Alti operatori:
  - Operatori de incrementare ++
  - Operatori de decrementare --
  - Operatorul conditional ?
  - Operatorul ,

Operatori ai C++

```cpp
#include <iostream>

int main() {

    int numarFavorit;
    std::cout << "Introduceti numarul vostru favorit cuprins intre 1 si 100: ";
    std::cin >> numarFavorit;
    std::cout << "Surprinzator, acesta este si numarul meu favorit!"<<std::endl;
    return 0;
```

Operator de inserare

Operator de extragere

Operator de rezolutie

- **Operatorul de inserare in flux** << se foloseste pentru afisare –insereaza (trimite) date din fluxul de intrare
- **Operatorul de extragere din flux** >> se foloseste pentru citire –extrage (preia) date din fluxul de intrare
- **Operatorul de rezolutie : :** indica apartenenta unui nume la un anumit spatiu de nume sau clasa

Semne de punctuatie in C++

```cpp
#include <iostream>

int main() {

    int numarFavorit;
    std::cout << "Introduceti numarul vostru favorit cuprins intre 1 si 100:";
    std::cin >> numarFavorit;
    std::cout << "Surprinzator, acesta este si numarul meu favorit!"<<std::endl;
    return 0;
}
```

Semnele de punctuație în C++ sunt simbolurile care delimitează, separă sau structurează instrucțiunile dintr-un program.
Exemple: ; , : . ( ) { } # ‘ “ //

Sintaxa limbajului C++

In cazul limbajelor de programare, succesiunile de cuvinte construite după anumite reguli, formează **propoziții**, numite **instrucțiuni,** iar acestea se finalizeaza cu "**;**"

Sintaxa unui limbaj de programare reprezinta ansamblul de reguli prin care se stabileste forma corecta a instructiunilor din acel limbaj.

Sintaxa cuprinde **toate regulile privind structura programului**

- Cuvintele cheie
- Identificatori
- Semnele de punctuatie
- Operatori
- Instructiuni

Ce este un preprocesor C++?

- Este un **program** care proceseaza codul sursa **inainte** de a fi vazut de catre compilator
- Detecteaza toate **comentariile** si le inlocuieste cu un spatiu
- Identifica **directivele preprocesor** si le executa (liniile din codul sursa care incep cu ‘#’)
- **Prepocesorul C++ nu intelege codul C++ (doar cauta in cod directivele prepocesor si pregateste codul sursa pentru compilator) !**

Exemple directive preprocesor

```
# include <iostream>
# include ”myfile.h”

# if
# elif
# else
# endif
```

```
# ifdef
# ifndef
# define
# undef
# line
# error
# pragma
```

- Cand prepocesorul detecteaza aceasta directiva, **inlocuieste linia de cod** respectiva cu **fisierul** la care se refera (in acest caz, fisierul *iostream.h*) si apoi **reproceseaza** acel fisier.
- Astfel, atunci cand compilatorul va vedea codul sursa, toate directivele prepocesor vor fi inlocuite si procesate si toate comentariile vor fi eliminate
- Se pot crea programe si **fara directiva `#include<iostream>`**, de exemplu cand nu facem scriere sau citire din consola!

Ce sunt comentariile intr-un limbaj de programare?

- Sunt explicatii scrise de catre programator in codul sursa
- Nu sunt niciodata citite de catre compilator

Tipuri de comentarii

1. // comentariu scris pe o singura linie

2. /* comentariu scris
pe mai
multe linii */

Tipuri utile de comentarii

- Mentionam autorul

```
/*****************************

author Simona

*****************************/
```

- Mentionam de ce am dorit sa folosim o anumita abordare

```
/*****************************

Folosim Teorema lui Pitagora pentru a calcula lungimea ipotenuzei

*****************************/
```

Comentarii in C++

- Comentariile trebuie evitate pe cat posibil pentru ca:
  - Nu trebuie sa comentam ceva evident
  - Codul trebuie sa fie inteles si fara comentarii
  - Sunt greu de intretinut (deseori se intampla ca un cod sa fie schimbat dupa ce a fost adaugata o nota explicativa ca si comentariu, dupa care comentariul sa nu mai fie reactualizat)
  - **Un comentariu bun nu scuza un cod scris gresit!**

  - **Shortcuts**:
    - Selectati textul pe care doriti sa il puneti in comentariu si folositi combinatia de taste **Ctrl+K+C**
    - Selectati textul pe care doriti sa il scoateti din comentariu si folositi combinatia de taste **Ctrl+K+U**

Functia `main()`

- Fiecare program C++ **trebuie** sa contina exact **1** functie **`main()`**
- Cuvantul **`main`** trebuie scris cu litere **mici**
- Cand un program C++ este executat, sistemul de operare apeleaza functia **`main()`** si se executa codul din interiorul acoladelor **{...}.**
- Cand executia ajunge la linia "**`return 0;`**", programul returneaza catre sistemul de operare un numar intreg. Daca acest numar este "**`0`**" atunci programul s-a finalizat cu succes, iar daca numarul este diferit de "**`0`**" sistemul de operare poate verifica valoarea returnata si sa identifice ce nu a functionat bine

```
int main()
{
    // code

    return 0;
}


program.exe
```

```
int main(int argc, char *argv[])
{
    // code

    return 0;
}


program.exe argument1 argument2
```

Functia main()

- Ambele variante sunt valide
  - In primul exemplu nu este necesar de niciun input de la utilizator
  - In al doilea exemplu, trebuie sa adaugam unul sau mai multe argumente in momentul in care rulam programul

```
int main()
{
    // code

    return 0;
}


program.exe
```

```
int main(int argc, char *argv[])
{
    // code

    return 0;
}


program.exe argument1 argument2
```

Functia main()

- Numele parametrilor din exemplul 2 sunt aleatorii
  - `argc` -> argument count
  - `argv` -> argument vector (ignorati *, o sa aflam mai multe despre ea ulterior)

```
int main()
{
    // code

    return 0;
}


program.exe
```

```
int main(int argc, char *argv[])
{
    // code

    return 0;
}


program.exe argument1 argument2
```

Functia main()

- Trebuie intotdeauna sa returneze un numar intreg
- Este o functie speciala, ea fiind apelata automat la inceputul rularii programului.
- In cadrul acestui curs, vom scrie programe care vor contine si **alte functii** pe langa functia principala **main()**

Spatiile de nume

- De ce trebuie sa folosim **std::cout** si nu doar **`cout`** pentru a printa un text?
    - Libraria **`iostream`** foloseste elemente care sunt definite in spatiul de nume **std** (`cout, cin, endl`, s.a.)
- Daca libraria `iostream` si o alta librarie externa contin functii sau obiecte diferite cu acelasi nume, de exemplu **`cout`**, compilatorul nu va stii daca ne referim la `std::cout` sau la `cout` din librarie externa. Prin urmare, vom avea un **conflict de nume.** Astfel, putem crea o functie noua **`cout`** intr-un spatiu de nume denumit **`myNameSpace`**, dupa care putem folosi **`myNameSpace::cout`**, iar compilatorul va stii care dintre cele doua functii dorim sa fie apelata

Spatiile de nume

- Un **spatiu de nume (namespace)** este un mecanism C++ care organizeaza si grupeaza elemente precum functii, variabile, clase si obiecte pentru a preveni conflictele de nume.

- ‘**::**’ este **operatorul de rezolutie** cu ajutorul caruia definim ce spatiu de lucru dorim sa folosim

- Deseori, programatorii nu doresc ca de fiecare data cand folosesc o functie din spatiu de nume **standard**, sa fie nevoiti sa scrie **std::numeFunctie.**

Spatiile de nume

```
#include <iostream>

using namespace std;      // Use the entire std namespace

int main()
{
    int favorite_number;

    cout << "Enter your favorite number between 1 and 100: ";

    cin >> favorite_number;

    cout << "Amazing!! That's my favorite number too!" << endl;
    cout << "No really!!, " << favorite_number
         << " is my favorite number!" << endl;

    return 0;
}
```

Spatiile de nume

- Foloseste tot spatiul de nume standard (chiar daca nu folosim toate metodele din acesta)

```
#include <iostream>

using namespace std;      // Use the entire std namespace

int main()
{
    int favorite_number;

    cout << "Enter your favorite number between 1 and 100: ";

    cin >> favorite_number;

    cout << "Amazing!! That's my favorite number too!" << endl;
    cout << "No really!!, " << favorite_number
         << " is my favorite number!" << endl;

    return 0;
}
```

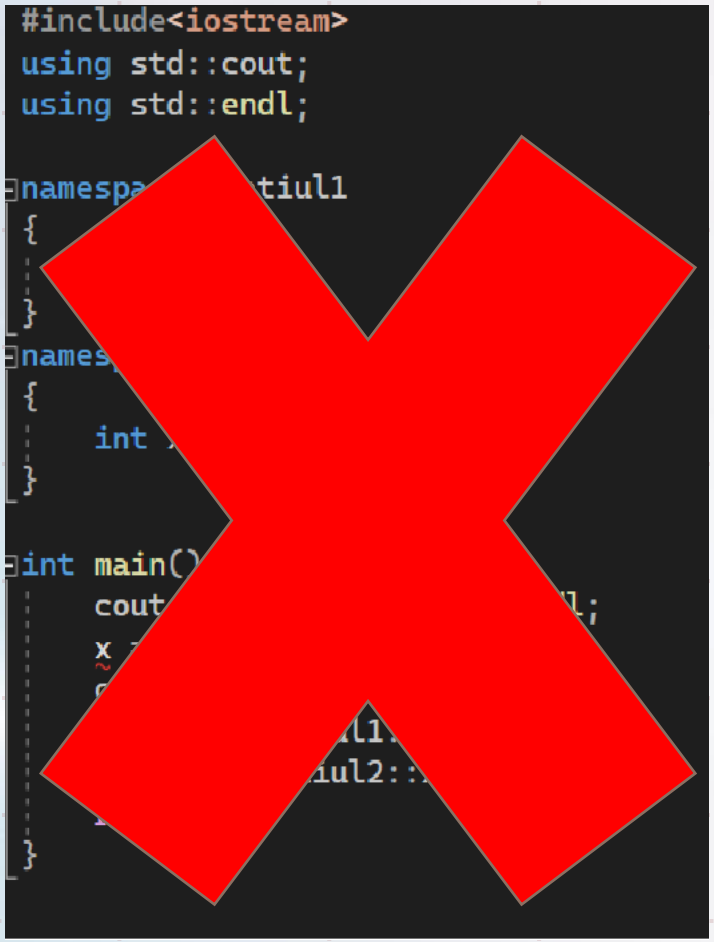
Spatiile de nume

- Este mult mai optim deoarece utilizam doar metodele de care avem nevoie reducand astfel riscul de conflicte de nume

```
#include <iostream>

using std::cout;    // use only what you need
using std::cin;
using std::endl;

int main()
{
    int favorite_number;

    cout << "Enter your favorite number between 1 and 100: ";

    cin >> favorite_number;

    cout << "Amazing!! That's my favorite number too!" << endl;
    cout << "No really!!, " << favorite_number
         << " is my favorite number!" << endl;

    return 0;
}
```

Spatiile de nume

```
#include<iostream>
using std::cout;
using std::endl;
int x;
namespace Spatiul1
{
    int x = 4;
}
namespace Spatiul2
{
    int x = 10;
}

int main() {
    cout << x<<endl;                    0
    cout << Spatiul1::x << endl;        4
    x = 5;
    cout << x << endl;                  5
    cout << Spatiul1::x << endl;        4
    Spatiul1::x = 7;
    cout << x << endl;                  5
    cout << Spatiul1::x << endl;        7
    cout << Spatiul2::x <<endl;         10
    return 0;
}
```



EROARE!

Spatiile de nume

```cpp
#include <iostream>

namespace Bibliotecă {
    void print() {
        std::cout << "Aceasta este funcția din biblioteca externă.\n";
    }
}

namespace Program {
    void print() {
        std::cout << "Aceasta este funcția din programul meu.\n";
    }
}

int main() {
    // Folosim funcția din spațiul de nume "Bibliotecă"
    Bibliotecă::print();

    // Folosim funcția din spațiul de nume "Program"
    Program::print();

    return 0;
}
```

Operatiile de citire/scriere in C++

- Sunt metodele prin care primim si dam date prin intermediul tastaturii si a consolei
- **cout** este identificatorul de **afisare/scriere** a datelor pe consola sau pe ecran
- **cerr** este identificatorul prin care afisam o eroare
- **cin** este identificatorul prin care **citim** date de la tastatura

- **<<** este operatorul de inserare/ scriere (folosit pentru **afisare/scriere**)
- **>>** este operatorul de extragere/ citire (folosit pentru **citire**)

Identificatorul *cout* si operatorul de inserare <<

```
cout << “valoare de iesire”;
```

- Operatorul de inserare “**<<**” insereaza datele din partea dreapta a operatorului in fluxul de iesire al datelor, fiind astfel afisat pe ecran

```
cout << “Valoarea de iesire este “ << “aabbcc”;
```

- Operatia poate fi inlantuita dupa cum se poate vedea mai sus, afisand in consola mesajul “Valoarea de intrare este aabbcc”

```
cout << “Valorea de iesire este “ << “aabbcc” << endl;
cout << “Un text” << “\n”;
```

- Pentru a afisa datele pe linii separate trebuie utilizat **endl** (care se regaseste in spatiul de nume standard) ori **“\n”** ori **‘\n’**

```
cout << “Valorea de iesire este “ << “aabbcc” << endl<< “Un text” << “\n”;
```

- Textul este afisat tot pe doua linii ca si mai sus!

Identificatorul *endl* si caracterul de linie noua *\n*

- "\n" sau '\n' este un caracter special care reprezinta o noua linie. Este interpretat ca un salt la urmatoarea linie si este pur si simplu inserat in fluxul de iesire.
    1. Nu face altceva decat sa insereze o linie noua.
    2. Nu forteaza golirea (flush) bufferului de iesire, deci poate fi mai rapid decat endl in situatii in care nu este necesara golirea bufferului.

- endl este un manipulator din C++ care face doua lucruri:
    1. Insereaza o linie noua (ca si "\n")
    2. Forteza golirea (flush) bufferului de iesire, adica se asigura ca toate datele din bufferul de iesire sunt trimise imediat catre destinatia finala (de exemplu, ecranul)

Identificatorul *endl* si caracterul de linie noua \n

Ce este un buffer?

Un **buffer** este o zonă temporară de memorie unde sunt stocate date înainte de a fi procesate sau trimise către destinația finală. În cazul ieșirii în C++, cum ar fi atunci când folosim `std::cout`, bufferul colectează datele care urmează să fie afișate pe ecran și le trimite când:

- bufferul este plin,
- programul termină execuția,
- sau când este făcut un **flush** explicit (`std::flush`) sau se foloseste `std::endl`.

Când să folosești fiecare?

- Foloseste `\n` cand nu ai nevoie de golirea imediată a bufferului și vrei să obții performanță maximă.

- Foloseste `endl` atunci când ai nevoie să te asiguri că datele sunt scrise imediat (de exemplu, în programe interactive sau la lucrul cu fișiere unde vrei să te asiguri că datele sunt salvate imediat).

Identificatorul *cin* si operatorul de extragere >>

```
cin >> Variable;
```

- Operatorul de extragere "**>>**" extrage datele din fluxul de intrare al datelor in Variable

```
cin >> variable1>> variable2;
```

- Operatia poate fi **inlantuita** dupa cum se poate vedea mai sus
- Se tasteaza valoarea primei variabile, dupa care se apasa **enter** urmand ca mai apoi sa setam a doua variabila
- Daca tipul datelor de intrare nu se potriveste cu tipul variabilei, **va rezulta o eroare semantica,** adica ceea ce interpreteaza compilatorul nu este correct
- Daca introducem de la tastatura sirul de caractere **" text"**, valoarea stocata in variabila nu mai contine spatiile sau tab-urile de la inceput si va fi **text**
- Similar **" text1 text2"** va fi citit doar **text1** oprindu-se la primul spatiu

Exemple

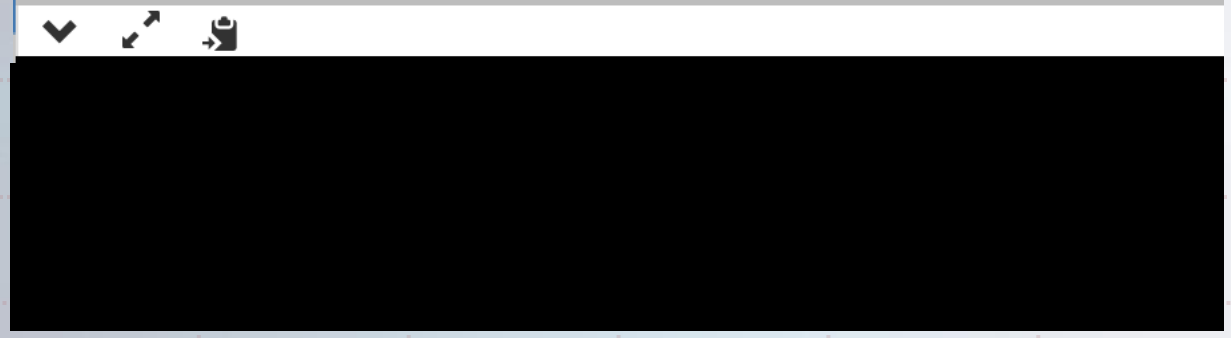
```cpp
#include <iostream>

using namespace std;

int main()
{
    cout << "Hello World" << endl;
    cout << "Hello";
}

```

Exemple

```cpp
#include <iostream>

using namespace std;

int main()
{
    cout << "Hello";
    cout << "World"<< endl;
}

```

Exemple

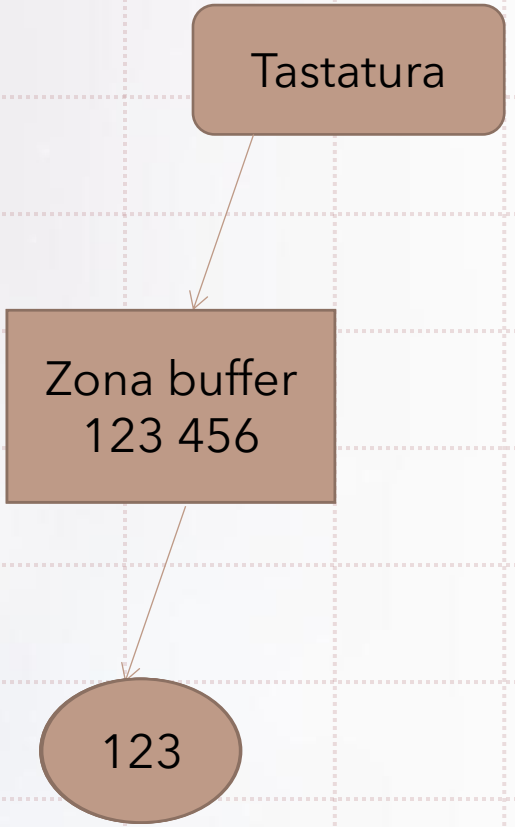
```cpp
#include <iostream>

using namespace std;

int main()
{
    cout << "Hello world!" << endl ;
    cout << "Hello" << " world!" << endl;
    cout << "Hello"<< " world!\n";
    cout << "Hello\nOut\nThere\n";
}
```



Exemple

```cpp
#include <iostream>

using namespace std;

int main()
{
    int num1;

    cout << "Introdu un integer:";
    cin >> num1;
    cout << "Ai introdus " << num1 << endl;
}
```

```
Introdu un integer:123
Ai introdus 123
```



Exemple

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      int num1;
8      int num2;
9
10     cout << "Introdu un integer:";
11     cin >> num1;
12      cout << "Introdu alt integer:";
13     cin >> num2;
14     cout << "Ai introdus " << num1 << " si " << num2<< endl;
15 }
```

input

```
Introdu un integer:12
Introdu alt integer:34
Ai introdus 12 si 34
```

Ce va afisa programul daca vom introduce de la tastatura "2 3"?

Exemple

- Daca introducem de la tastatura numerele “2 3” separate prin spatiu, urmate un singur Enter va atribui ambelor variabile cele 2 valori, neasteptand al doilea Enter, deoarece primul input deja contine 2 valori de tipul integer

```cpp
#include <iostream>

using namespace std;

int main()
{
    int num1;
    int num2;

    cout << "Introdu un integer:";
    cin >> num1;
     cout << "Introdu alt integer:";
    cin >> num2;
    cout << "Ai introdus " << num1 << " si " << num2<< endl;
}
```

input

```
Introdu un integer:2 3
Introdu alt integer:Ai introdus 2 si 3
```

Exemple

```cpp
#include <iostream>

using namespace std;

int main()
{
    int num1;
    int num2;

    cout << "Introdu doua variabile de tip integer:";
    cin >> num1 >> num2;
    cout << "Ai introdus " << num1 << " si " << num2<< endl;
}

```

Ai introdus 100 si 200

Exemple

```cpp
#include <iostream>

using namespace std;

int main()
{
    int num1;
    int num2;

    cout << "Introdu doua variabile de tip integer:";
    cin >> num1 >> num2;
    cout << "Ai introdus " << num1 << " si " << num2<< endl;
}

```

input

```
Introdu doua variabile de tip integer:100 200 300
```

Ai introdus 100 si 200

Exemple

```cpp
#include <iostream>

using namespace std;

int main()
{
    double num1;


    cout << "Introdu o variabila de tipul double:";
    cin >> num1;
    cout << "Ai introdus " << num1 << endl;
}

```

```
Introdu o variabila de tipul double:  3.5
Ai introdus 3.5
```

Exemple

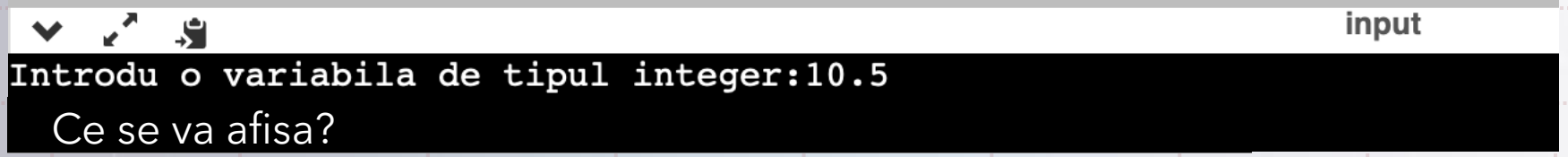
```cpp
#include <iostream>

using namespace std;

int main()
{
    int num1;
    double num2;

    cout << "Introdu o variabila de tipul integer:";
    cin >> num1;
    cout << "Introdu o variabila de tipul double:";
    cin >> num2;
    cout << "Ai introdus " << num1 << " si "<< num2 << endl;
}
```

input

```
Introdu o variabila de tipul integer:10
Introdu o variabila de tipul double:2.5
Ai introdus 10 si 2.5
```

Exemple

```cpp
#include <iostream>

using namespace std;

int main()
{
    int num1;
    double num2;

    cout << "Introdu o variabila de tipul integer:";
    cin >> num1;
    cout << "Introdu o variabila de tipul double:";
    cin >> num2;
    cout << "Ai introdus " << num1 << " si "<< num2 << endl;
}
```



Ce se va afisa?

Exemple

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

int main()
{
    int num1;
    cout << "Introdu o variabila de tip integer: ";
    cin >> num1;
    cout << "Ai introdus "<< num1 << endl;
    return 0;
}
```

Ce se va afisa daca introducem de la tastatura “Informatica”? Ai introdus 0

Ce se va afisa daca introducem de la tastatura “Informatica 10”? Ai introdus 0

Ce se va afisa daca introducem de la tastatura “10Informatica”? Ai introdus 10

https://test-master.space