Fondamenti di Programmazione (A)

Lecture 5 - Introduzione a C++

Vincenzo Arceri - Università di Parma - vincenzo.arceri@unipr.it

Linguaggi di programmazione

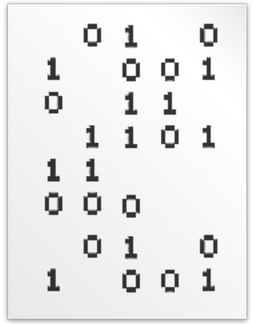
- E' un linguaggio **formale** che comprende un insieme di istruzioni/comandi che possono essere utilizzati per comunicare con una macchina (cioè è **eseguibile**)
- Perché formale?
  - **Sintassi**: specifica quali istruzioni sono *conformi* al linguaggio di programmazione
  - **Semantica**: specifica il comportamento e il significato delle istruzioni
- Un **programma** di un linguaggio L è una sequenza *finita* di istruzioni di L

Linguaggi eseguibili

- Perché **eseguibile?**
  - Perché la macchina su cui esegue il programma scritto in L è in grado di capire le istruzione del linguaggio L
  - La macchina ne *conosce* la sintassi e la semantica

Come eseguire un programma di L?

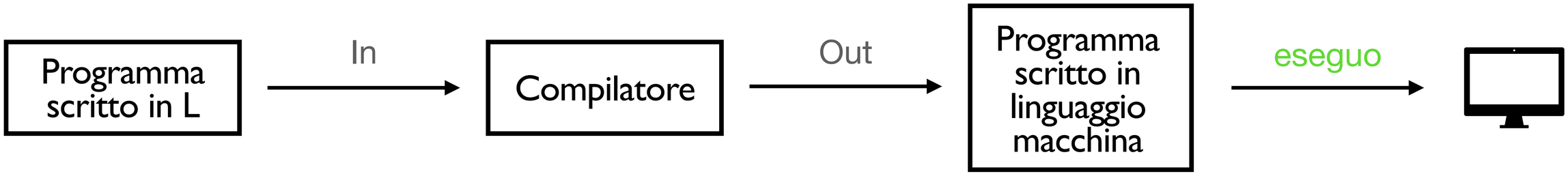
- Problema: la macchina conosce solamente il linguaggio macchina, non L



- Soluzioni
  - Compilazione
  - Interpretazione

Come eseguire un programma di L?

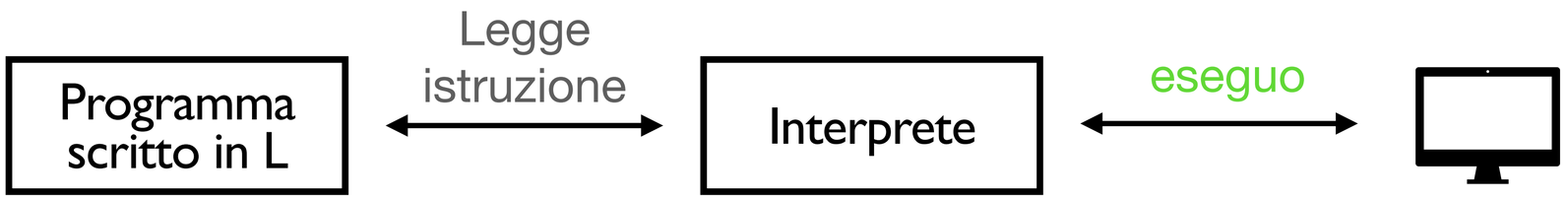
Compilazione



Traduce un qualsiasi programma scritto in L in un programma scritto in linguaggio macchina

Come eseguire un programma di L?

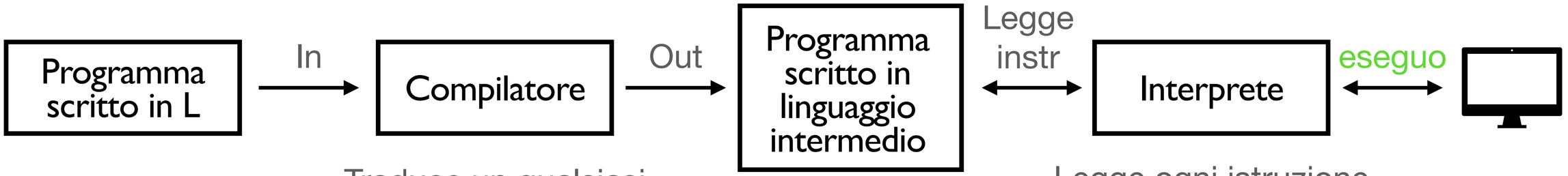
Interpretazione



Legge ogni istruzione del programma scritto in L **una alla volta** e la esegue sulla macchina, poi passa alla successiva

Come eseguire un programma di L?

Approcci misti



Traduce un qualsiasi programma scritto in L in un programma scritto in linguaggio intermedio

Legge ogni istruzione del programma scritto nel linguaggio intermedio **una alla volta** e la esegue sulla macchina, poi passa alla successiva

Esempi

- C++ (compilato)

```
g++ example.cpp // compilazione
./a.out // eseguo
```

- Python (interpretato)

```
python example.py // interpreto
```

- Java (approccio misto)

```
javac Example.java // compilo in linguaggio intermedio
java Example // interpreto linguaggio intermedio
```

C++

- Sviluppato all'inizio degli anni '80
- *Estensione* del linguaggio C
  - Tutto quello che è possibile scrivere in C è possibile scriverlo anche in C++
  - Posso usare un compilatore C++ per compilare ed eseguire programmi C
- Principale differenza fra C e C++
  - programmazione orientata agli oggetti
  - Fondamenti di programmazione A (questo corso): paradigma imperativo
  - Fondamenti di programmazione B (prossimo semestre): paradigma orientato agli oggetti

Primo programma in C++

Problema

- Problema: dati in input tre numeri interi, calcolare e stampare a video la loro media

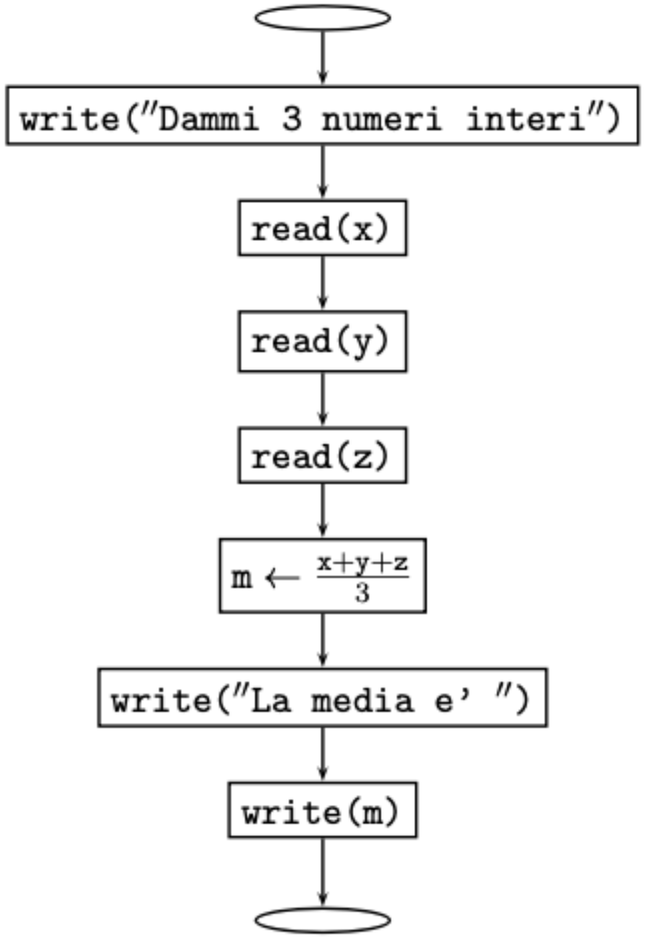
Input: tre numeri interi

Output: un numero reale

Input	Output
1, 2, 3	2
4, 7, 8	6.33
-1, 5, 3	2,33

Primo programma in C++

Flow-chart



Primo programma in C++

Codice C++

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

int main() {
    int x, y, z;
    cout << "Inserisci 3 numeri interi" << endl;
    cin >> x >> y >> z;
    float m;
    m = (x + y + z) / 3.0;
    cout << "La media e " << m << endl;
    return 0;
}
```

Primo programma in C++

Codice C++

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

int main() {
    int x, y, z;
    cout << "Inserisci 3 numeri interi" << endl;
    cin >> x >> y >> z;
    float m;
    m = (x + y + z) / 3.0;
    cout << "La media e " << m << endl;
    return 0;
}
```

#include è una direttiva per il pre-processore (che è integrato nel compilatore) per includere le funzioni contenute nel file specificato (iostream)

iostream: Standard Input / Output Streams Library

Contiene una serie di funzioni input/output predefinite

Primo programma in C++

Codice C++

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

int main() {
    int x, y, z;
    cout << "Inserisci 3 numeri interi" << endl;
    cin >> x >> y >> z;
    float m;
    m = (x + y + z) / 3.0;
    cout << "La media e " << m << endl;
    return 0;
}
```

int main(): indica l'inizio del *programma principale*, il cui contenuto è compreso fra le due parentesi graffe

Primo programma in C++

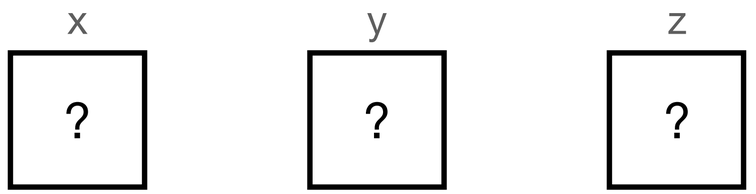
Codice C++

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

int main() {
    int x, y, z;
    cout << "Inserisci 3 numeri interi" << endl;
    cin >> x >> y >> z;
    float m;
    m = (x + y + z) / 3.0;
    cout << "La media e " << m << endl;
    return 0;
}
```

int x, y, z;
Dichiarazione di variable: definizione di tre nuove variabili, chiamate **x, y** e **z**
di tipo intero, cioè possono contenere **solo** valori interi (-2, 5, 1001, ma non 5.3)

NB: ogni variabile utilizzata dal programma deve essere prima dichiarata



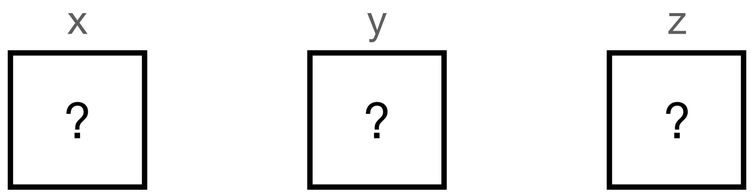
Primo programma in C++

Codice C++

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

int main() {
    int x, y, z;
    cout << "Inserisci 3 numeri interi" << endl;
    cin >> x >> y >> z;
    float m;
    m = (x + y + z) / 3.0;
    cout << "La media e " << m << endl;
    return 0;
}
```

cout: stream di ***output***
Invio (<<) di una stringa/testo verso lo standard output (monitor)
endl denota il carattere speciale "a capo"



Primo programma in C++

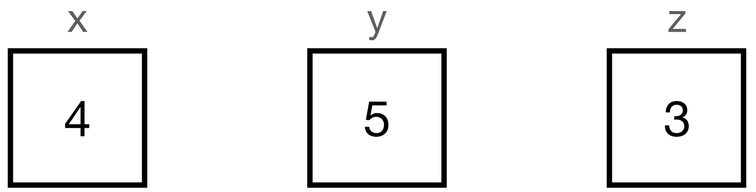
Codice C++

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

int main() {
    int x, y, z;
    cout << "Inserisci 3 numeri interi" << endl;
    cin >> x >> y >> z;
    float m;
    m = (x + y + z) / 3.0;
    cout << "La media e " << m << endl;
    return 0;
}
```

cin: stream di ***input***
Il programma si aspetta l'inserimento di tre valori dallo standard input (tastiera) che verranno memorizzati nelle variabili **x, y** e **z**

Il programma attenderà finché i tre valori non verranno inseriti



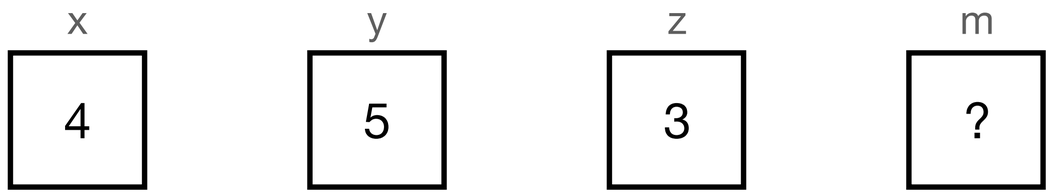
Primo programma in C++

Codice C++

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

int main() {
    int x, y, z;
    cout << "Inserisci 3 numeri interi" << endl;
    cin >> x >> y >> z;
    float m;
    m = (x + y + z) / 3.0;
    cout << "La media e " << m << endl;
    return 0;
}
```

float m;
Dichiarazione di variable: definizione di una nuova variabile, chiamata **m**
di tipo float, cioè possono contenere **solo** valori reali (5.4, -6.1, 3, …)



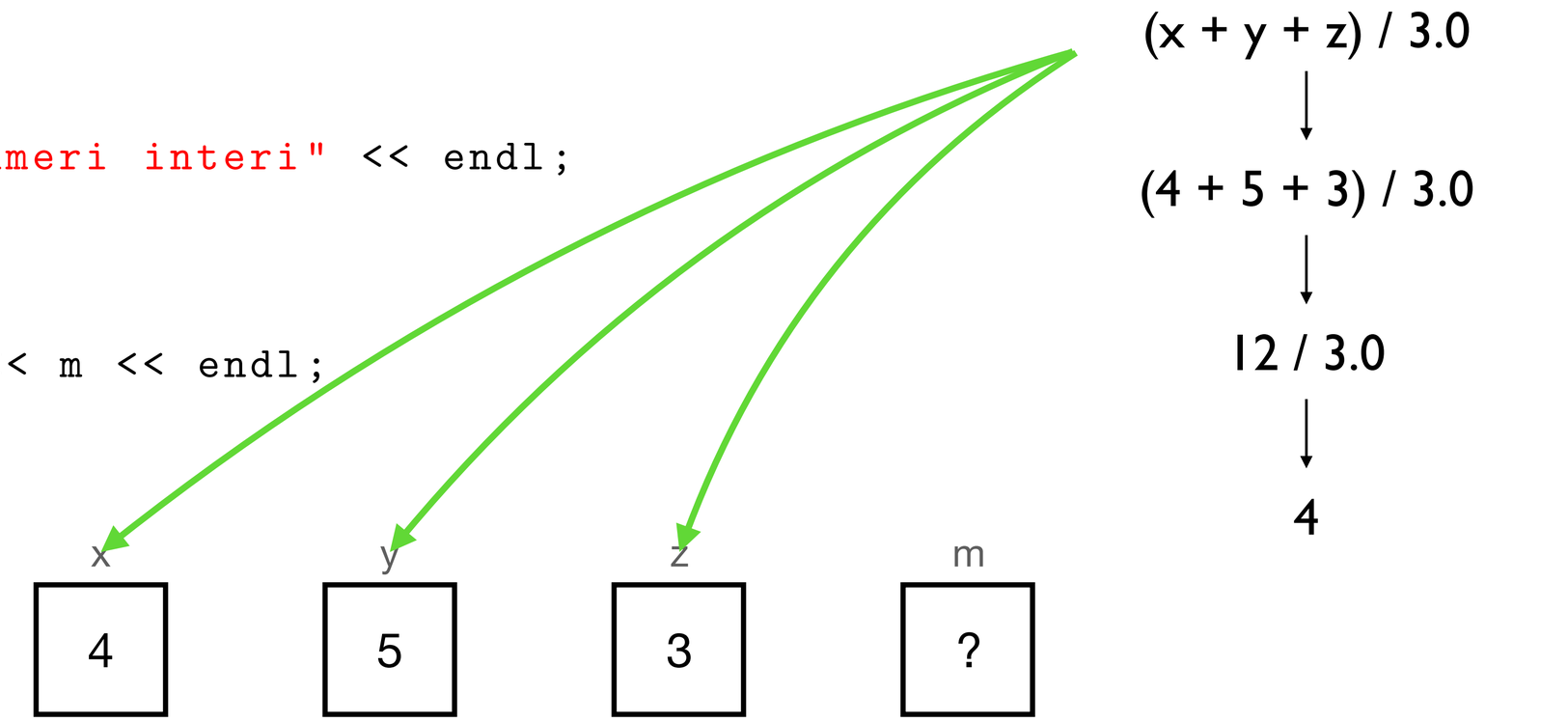
Primo programma in C++

Codice C++

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

int main() {
    int x, y, z;
    cout << "Inserisci 3 numeri interi" << endl;
    cin >> x >> y >> z;
    float m;
    m = (x + y + z) / 3.0;
    cout << "La media e " << m << endl;
    return 0;
}
```

Assegnamento:
Valuta l'espressione a destra del simbolo **=** e il suo risultato viene memorizzato all'interno della variable **m**



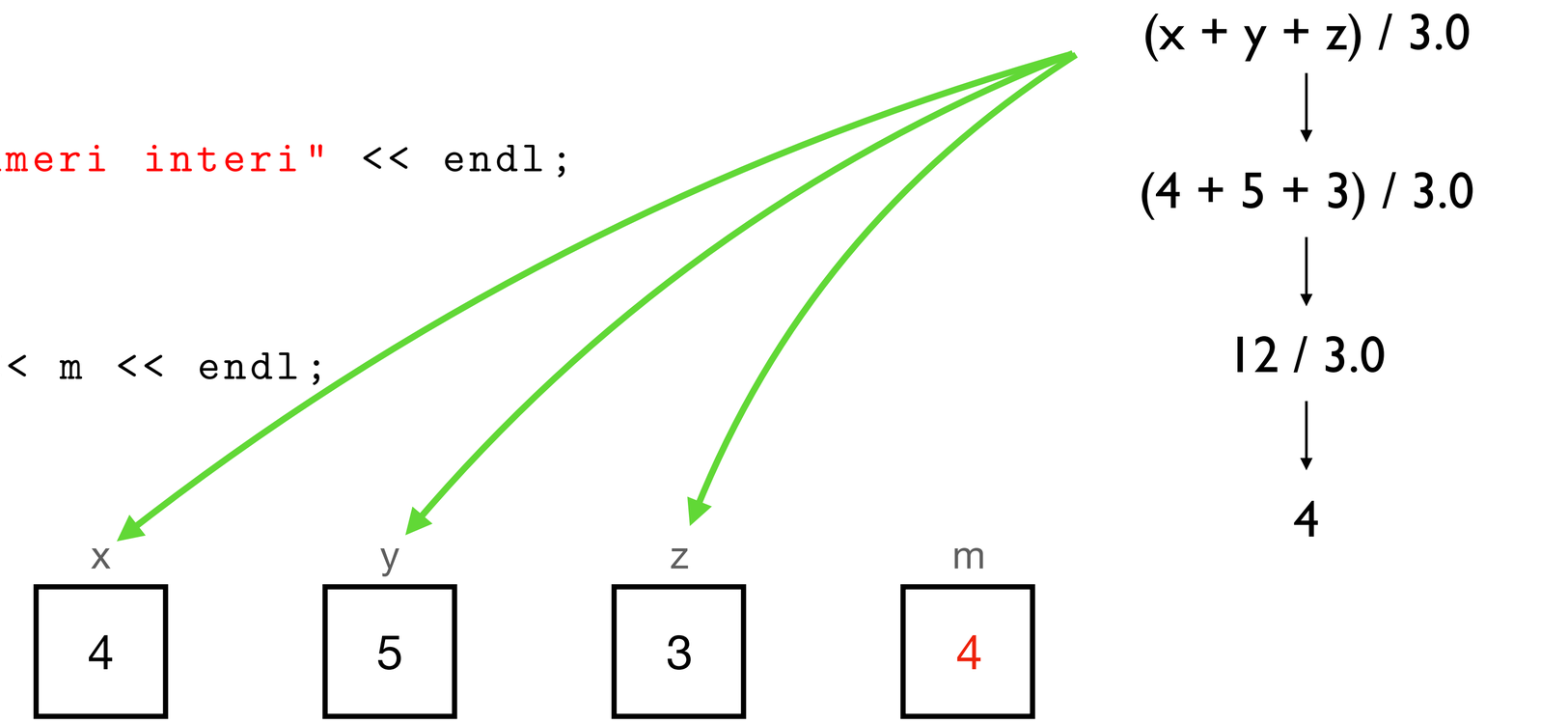
Primo programma in C++

Codice C++

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

int main() {
    int x, y, z;
    cout << "Inserisci 3 numeri interi" << endl;
    cin >> x >> y >> z;
    float m;
    m = (x + y + z) / 3.0;
    cout << "La media e " << m << endl;
    return 0;
}
```

Assegnamento:
Valuta l'espressione a destra del simbolo **=** e il suo risultato viene memorizzato all'interno della variable **m**



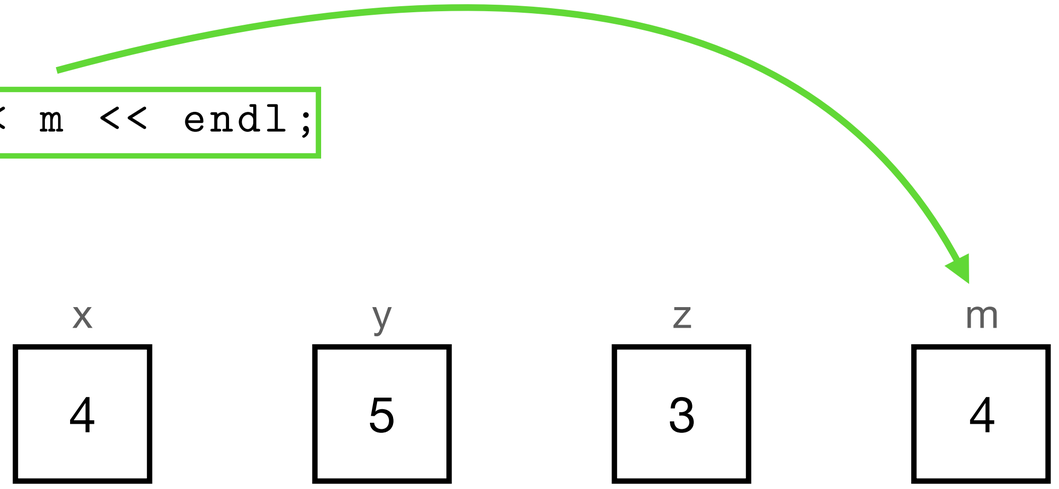
Primo programma in C++

Codice C++

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

int main() {
    int x, y, z;
    cout << "Inserisci 3 numeri interi" << endl;
    cin >> x >> y >> z;
    float m;
    m = (x + y + z) / 3.0;
    cout << "La media e " << m << endl;
    return 0;
}
```

cout: stream di ***output***
Invio di una stringa/testo verso lo standard output (monitor)



Primo programma in C++

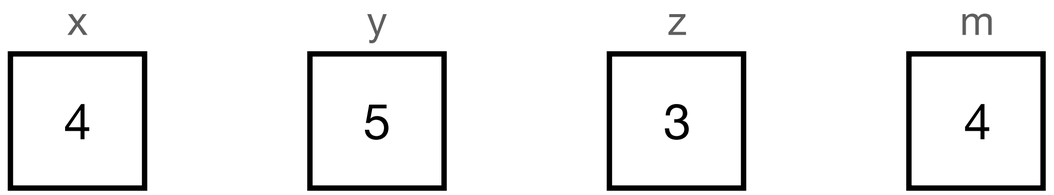
Codice C++

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

int main() {
    int x, y, z;
    cout << "Inserisci 3 numeri interi" << endl;
    cin >> x >> y >> z;
    float m;
    m = (x + y + z) / 3.0;
    cout << "La media e " << m << endl;
    return 0;
}
```

return 0:
Il programma ha terminato e ritorna al chiamante del programma il valore 0

Tipicamente 0 significa che il programma è terminato correttamente



Primo programma in C++

Codice C++

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

int main() {
    int x, y, z;
    cout << "Inserisci 3 numeri interi" << endl;
    cin >> x >> y >> z;
    float m;
    m = (x + y + z) / 3.0;
    cout << "La media e " << m << endl;
    return 0;
}
```

using namespace std
Gli identificatori che non sono dichiarati dal programma vanno cercati nello *spazio dei nomi* std (dentro iostream)

Primo programma in C++

Codice C++

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

int main() {
    int x, y, z;
    cout << "Inserisci 3 numeri interi" << endl;
    cin >> x >> y >> z;
    float m;
    m = (x + y + z) / 3.0;
    cout << "La media e " << m << endl;
    return 0;
}
```

using namespace std
Gli identificatori che non sono dichiarati dal programma vanno cercati nello *spazio dei nomi* std (dentro iostream)

cin, cout, endl non sono identificatori dichiarati dal programma e quindi vengono cercati dentro std

Primo programma in C++

Codice C++

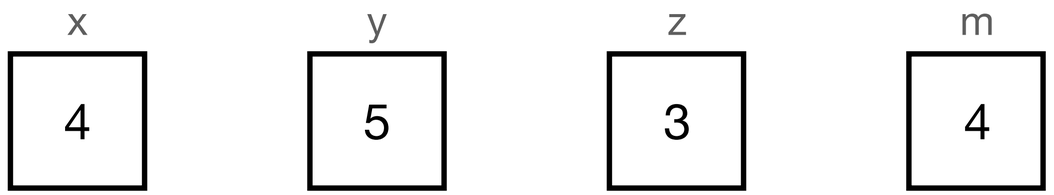
```
#include <iostream>
using namespace std;

int main() {
    int x, y, z;
    cout << "Inserisci 3 numeri interi" << endl;
    cin >> x >> y >> z;
    float m;
    m = (x + y + z) / 3.0;
    cout << "La media e " << m << endl;
    return 0;
}
```

Ogni istruzione/comando/statement **deve** finire con un punto e virgola ; (terminatore di comando)

int, float, using, return sono **parole chiave** di C++: per esempio, non posso dichiarare una variable e chiamarla **return**

Commenti: testo libero (non deve seguire la sintassi di C++) per annotare il codice e aggiungere informazioni aggiuntive su parti di esso per aumentarne la leggibilità



Primo programma in C++

Indentare il codice

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

int main() {
    int x, y, z;
    cout << "Inserisci 3 numeri interi" << endl;
    cin >> x >> y >> z;
    float m;
    m = (x + y + z) / 3.0;
    cout << "La media e " << m << endl;
    return 0;
}
```

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

int main() {
    int x, y, z; cout << "Inserisci 3 numeri interi" << endl; cin >> x >> y
    float m; m = (x + y + z) / 3.0;
             cout << "La media e " << m << endl;
                                                    return 0;
}
```

Indentare il codice: dividere ciascun statement su più righe e allinearli rispetto al margine sinistro per aumentare la leggibilità del codice

Convenzione: allineare sulla stessa colonna tutti gli statement compresi fra due parentesi graffe

(prenderemo dimestichezza con questo durante il corso)