

Variabili e Strutture di Controllo in C++

Informatica - CdL Triennale in Fisica

Carlo Mereghetti

Sito del corso: `mereghetti.di.unimi.it/inf`

Pagina personale: `mereghetti.di.unimi.it`

Dichiarazione di variabili

```
int x, y, z;  
double d;  
string s;
```

Definizione di variabili

```
int x = 1, y = 2, z;  
double d = 3.5;  
string s = "ciao carlo";
```

Le variabili si riempiono con:

- operazioni di lettura da input (vedi dopo)
- assegnamenti: $z = x + y$;

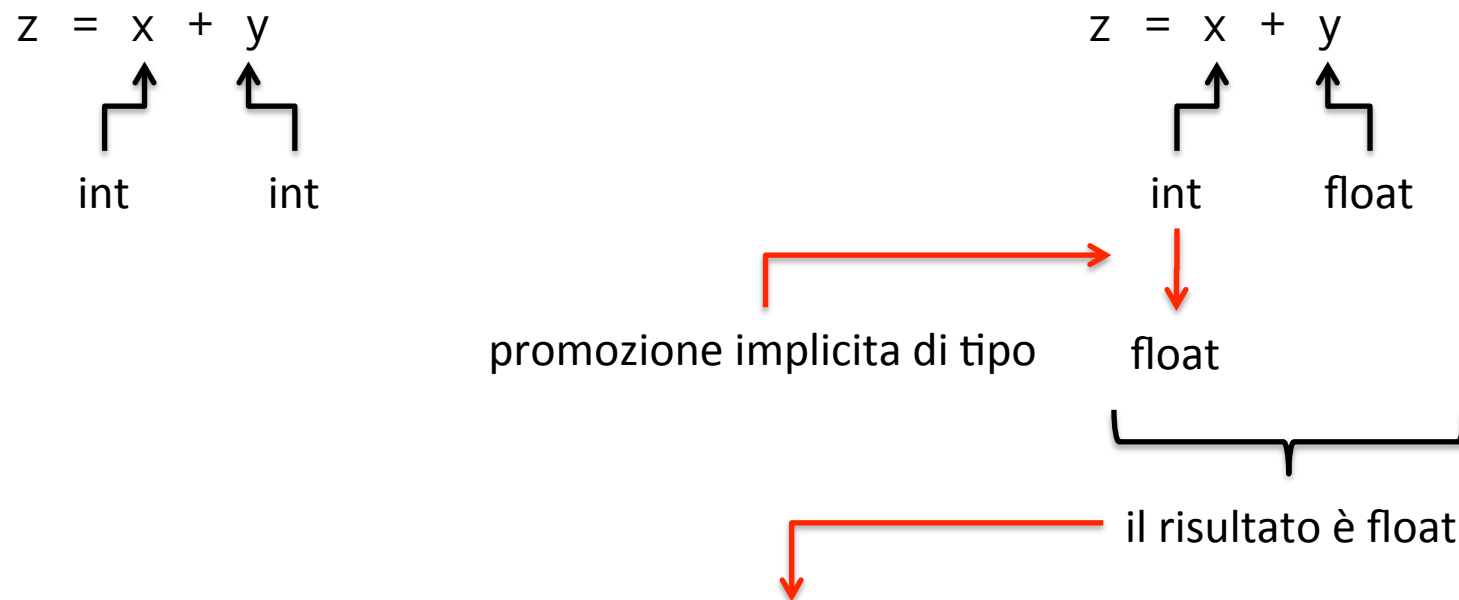
Definizione di costanti

```
const double PI = 3.14;  
...  
double area = PI * r * r
```

Operazioni su variabili numeriche:

- $+$, $-$, $*$, $/$, $\%$ = resto della divisione intera (modulo): $x \% y$
- **ATTENZIONE!!** Se $/$ è tra due variabili intere allora restituisce la divisione intera
Se è tra due double o float restituisce la divisione reale
- $++x$ oppure $x++ \rightarrow x = x + 1$ $--x$ oppure $x-- \rightarrow x = x - 1$ Attenzione all'uso in espressioni:
 $\text{int } x = 5; \text{int } y = ++x; \rightarrow x = 6 \text{ e } y = 6$ invece $\text{int } x = 5; \text{int } y = x++; \rightarrow x = 6 \text{ e } y = 5$

Espressioni aritmetiche su tipi eterogenei



ATTENZIONE!! Se z fosse `int`, l'assegnamento è permesso e il risultato viene troncato!

Forzare il tipo di variabili ed espressioni: casting

`int s, k;` $s / k \rightarrow \text{int}$

$(\text{float}) s / k \rightarrow \text{float}$

casting ATTENZIONE!! $(\text{float}) (s / k) \rightarrow \text{float}$

Leggere e scrivere variabili

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

cout ← stream di output

cin ← stream di input

```
int x;
double d;
string s;
```

output cout << "x = " << x << " s = " << s << endl;

input cin >> x >> d >> s;

La struttura di controllo **selezione**

```
SE condizione ALLORA
    istruzione
    ...
    istruzione
ALTRIMENTI
    istruzione
    ...
    istruzione
FINESE
```

```
if ( condizione ) {
    istruzione;
    ...
    istruzione;
}
else {
    istruzione;
    ...
    istruzione;
}
```

```
SE condizione ALLORA
    istruzione
    ...
    istruzione
FINESE
```



```
if ( condizione ) {
    istruzione;
    ...
    istruzione;
}
```

La struttura di controllo **selezione**

Problema: leggere due numeri e stampare il massimo

```
int x, y;

cout << "Primo numero: ";
cin >> x;
cout << "Secondo numero: ";
cin >> y;

if ( x > y )
    cout << x << endl;
else
    cout << y << endl;
```

- **condizione:** espressione booleana può restituire **true** o **false**
- operatori in espressioni booleane: < , > , == , != , >= , <= , ...

E se volessimo rilevare l'uguaglianza tra x e y ?

```
int x, y;

cout << "Primo numero: ";
cin >> x;
cout << "Secondo numero: ";
cin >> y;

if ( x > y )
    cout << x << endl;
else
    if ( x == y )
        cout << "UGUALI" << endl;
    else
        cout << y << endl;
```

La struttura di controllo **selezione**

Problema: leggere tre numeri e stampare il massimo

```
int x, y, z;

cout << "Inserire i numeri: ";
cin >> x >> y >> z;

if ( x > y )
    if ( x > z )
        cout << x << endl;
    else
        cout << z << endl;
else
    if ( z > y )
        cout << z << endl;
    else
        cout << y << endl;
```

```
int x, y, z;

cout << "Inserire i numeri: ";
cin >> x >> y >> z;

if ( x < y )
    x = y;
if ( x < z )
    x = z;

cout << x << endl;
```

L'operatore ternario condizionale

condizione ? espressione1 : espressione2

Problema: leggere due numeri e mettere il massimo in una variabile z

```
int x, y, z;

cout << "Primo numero: ";
cin >> x;
cout << "Secondo numero: ";
cin >> y;

if ( x > y )
    z = x;
else
    z = y;
```

```
int x, y, z;

cout << "Primo numero: ";
cin >> x;
cout << "Secondo numero: ";
cin >> y;

z = ( x > y ? x : y );
```

Problema: leggere due numeri e stampare il massimo

```
...
cout << ( x > y ? x : y ) << endl;
```


L'operatore ternario condizionale

condizione ? espressione1 : espressione2

Problema: leggere due numeri e mettere in una variabile z la loro somma **se** il primo è maggiore del secondo, **altrimenti** il loro prodotto

```
int x, y, z;

cout << "Primo numero: ";
cin >> x;
cout << "Secondo numero: ";
cin >> y;

if ( x > y )
    z = x + y;
else
    z = x * y;
```

```
int x, y, z;

cout << "Primo numero: ";
cin >> x;
cout << "Secondo numero: ";
cin >> y;

z = ( x > y ? x + y : x * y );
```

Il tipo **bool**

bool → **true** **false** occupazione: **1byt**

```
bool b = ( x >= y );  
bool c = true;
```

Di fatto:

- **true** → qualsiasi valore intero diverso da 0
- **false** → 0

Operatori booleani per espressioni booleane complesse

espr1 && espr2 **AND** (congiunzione): vera quando **entrambe** le espressioni sono vere

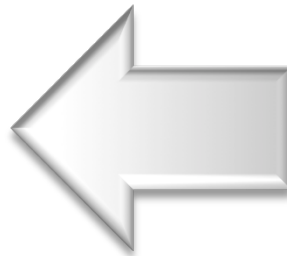
espr1 || espr2 **OR** (disgiunzione): vera quando **almeno un'**espressione è vera

!espr **NOT** (negazione): vera quando l'espressione è falsa

```
x >= 5 && x <= 10
```

```
x > 5 || x < 10
```

```
!(x < 5 || x > 10)
```



Per quali valori di x sono vere
le seguenti espressioni booleane?

Short circuit evaluation di espressioni booleane

`espr1 && espr2`

1. Viene valutata `espr1`
2. Se `espr1` è **falsa** allora `espr2` **NON** viene valutata (sarebbe inutile) e globalmente viene restituito **falso**
1. Altrimenti `espr2` viene valutata

`espr1 || espr2`

1. Viene valutata `espr1`
2. Se `espr1` è **vera** allora `espr2` **NON** viene valutata (sarebbe inutile) e globalmente viene restituito **vero**
1. Altrimenti `espr2` viene valutata

E' solo questione di "efficienza"?

`x > 10 && ++y < 20`

ATTENZIONE all'utilizzo dell'operatore di uguaglianza!!

```
int x;  
cin >> x;  
if ( x = 5 )  
    cout << "Hai inserito 5" << endl;  
else  
    cout << "Hai inserito un numero diverso da 5" << endl;
```

La struttura di controllo **iterazione** (cicli)

```
QUANDO ( condizione ) ESEGUI  
    istruzione  
    ...  
    istruzione  
RIPETI
```

Min: 0



Max: infinito

```
while ( condizione ) {  
    istruzione;  
    ...  
    istruzione;  
}
```

```
RIPETI  
    istruzione  
    ...  
    istruzione  
QUANDO ( condizione )
```

Min: 1



Max: infinito

```
do {  
    istruzione;  
    ...  
    istruzione;  
} while ( condizione )
```

La struttura di controllo **iterazione** (cicli)

Problema: media di una sequenza di interi terminata da 0

`while`

```
int n, s = 0, k = 0;

cin >> n;
while ( n != 0 ) {
    s = s + n;
    k++;
    cin >> n;
}

if ( k != 0 )
    cout << "Media: " << (double) s / k << endl;
else
    cout << "Non ci sono numeri" << endl;
```

La struttura di controllo **iterazione** (cicli)

Problema: media di una sequenza di interi terminata da 0

do ... while

```
int n, s = 0, k = 0;

do {
    cin >> n;
    if ( n != 0 ) {
        s = s + n;
        k++;
    }
} while ( n != 0 )

if ( k != 0 )
    cout << "Media: " << (double) s / k << endl;
else
    cout << "Non ci sono numeri" << endl;
```

Il ciclo **for**

Problema: stampa dei naturali non superiori a 100

```
int i;  
  
i = 0;  
  
while ( i <= 100 ) {  
    cout << i << endl;  
    i++;  
}
```

← inizializzazione

← condizione

← istruzioni

← aggiornamento

```
for ( inizializzazione; condizione; aggiornamento ) {  
    istruzioni;  
}
```


Il ciclo **for**

Problema: stampa dei naturali non superiori a 100

```
int i;  
  
i = 0;  
  
while ( i <= 100 ) {  
    cout << i << endl;  
    i++;  
}
```

← inizializzazione

← condizione

← istruzioni

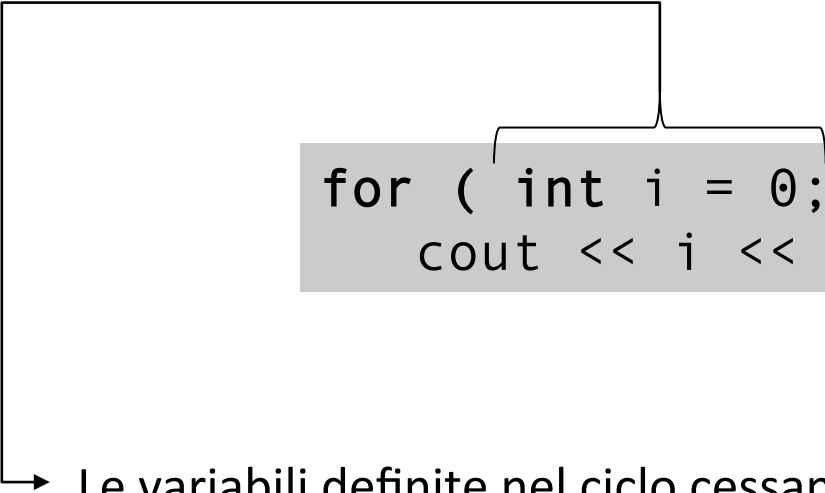
← aggiornamento

```
int i;  
  
for ( i = 0; i <= 100; i++ )  
    cout << i << endl;
```

Il ciclo **for**

Problema: stampa dei naturali non superiori a 100

```
int i;  
  
for ( i = 0; i <= 100; i++ )  
    cout << i << endl;
```



```
for ( int i = 0; i <= 100; i++ )  
    cout << i << endl;
```

→ Le variabili definite nel ciclo cessano di esistere al termine dello stesso

Il ciclo **for**

Problema: somma dei naturali non superiori a 100

```
int s = 0;

for( int i = 0; i <= 100; i++ )
    s = s + i;

cout << s << endl;
```

```
int s = 0;

for( int i = 0; i <= 100; s = s + i, i++ );

cout << s << endl;
```

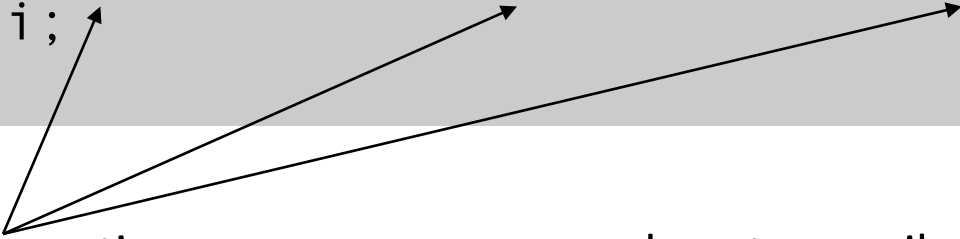
```
for( int i = 0, s = 0; i <= 100; s = s + i, i++ );

cout << s << endl;
```

**Perché
non
funziona?**

Il ciclo **for**

```
for ( inizializzazione; condizione; aggiornamento ) {  
    istruzioni;  
}
```



Alcune delle componenti possono mancare, ma devo tenere il `;` come separatore

```
int i = 0, s = 0;  
  
for( ; i <= 100; ) {  
    s = s + i;  
    i++;  
}  
  
cout << s << endl;
```

A quale struttura corrisponde?

Cicli innestati

Problema: per ogni naturale non superiore a 100 elencarne i divisori

```
for( int i = 0; i <= 100; i++ ) {  
    cout << "Divisori di " << i << ": ";  
    for ( int j = 1; j <= i; j ++ )  
        if ( i % j == 0 )  
            cout << j << " ";  
    cout << endl;  
}
```

Interrompere un ciclo: **break**

Problema: media di una sequenza di interi terminata da 0

while

```
int n, s = 0, k = 0;

while ( true ) {
    cin >> n;
    if ( n == 0 )
        break;
    s = s + n;
    k++;
}

if ( k != 0 )
    cout << "Media: " << (double) s / k << endl;
else
    cout << "Non ci sono numeri" << endl;
```

Interrompere un ciclo: **break**

Nel caso di cicli innestati, `break` termina il ciclo più interno

