

---

# Esercitazioni con cicli **for**

Filippo Cugini

---

# Esempi di uso dell'istruzione `for`

---

1. Far variare la variabile di controllo da 1 a 100 per incrementi di 1

# Esempi di uso dell'istruzione **for**

---

1. Far variare la variabile di controllo da 1 a 100 per incrementi di 1

```
unsigned int c;  
  
for (c = 1; c <= 100; ++c)
```

# Esempi di uso dell'istruzione `for`

---

2. Far variare la variabile di controllo da 100 a 1 per decrementi di 1

# Esempi di uso dell'istruzione **for**

---

2. Far variare la variabile di controllo da 100 a 1 per decrementi di 1

```
unsigned int c;  
  
for (c = 100; c >= 1; --c)
```

# Esempi di uso dell'istruzione `for`

---

3. Far variare la variabile di controllo da 7 a 77 per incrementi di 7

## Esempi di uso dell'istruzione **for**

---

3. Far variare la variabile di controllo da 7 a 77 per incrementi di 7

```
unsigned int c;  
  
for (c = 7; c <= 77; c += 7)
```

# Esempi di uso dell'istruzione `for`

---

4. Far variare la variabile di controllo da 20 a 2 per incrementi di  $-2$



# Esempi di uso dell'istruzione **for**

---

4. Far variare la variabile di controllo da 20 a 2 per incrementi di  $-2$

```
unsigned int c;  
  
for (c = 20; c >= 2; c -= 2)
```

# Esempi di uso dell'istruzione `for`

---

5. Far variare la variabile di controllo secondo la seguente sequenza di valori: 2, 5, 8, 11, 14, 17

## Esempi di uso dell'istruzione **for**

---

5. Far variare la variabile di controllo secondo la seguente sequenza di valori: 2, 5, 8, 11, 14, 17

```
unsigned int c;  
  
for (c = 2; c <= 17; c += 3)
```

# Esempi di uso dell'istruzione `for`

---

6. Far variare la variabile di controllo secondo la seguente sequenza di valori: 44, 33, 22, 11, 0

## Esempi di uso dell'istruzione **for**

---

6. Far variare la variabile di controllo secondo la seguente sequenza di valori: 44, 33, 22, 11, 0

```
unsigned int c;  
  
for (c = 44; c >= 0; c -= 11)
```

# Esempi di uso dell'istruzione `for`

---

7. Scrivere un programma che utilizza l'istruzione `for` per sommare tutti i numeri interi pari da 2 a 100

# Esempi di uso dell'istruzione `for`

7. Scrivere un programma che utilizza l'istruzione `for` per sommare tutti i numeri interi pari da 2 a 100

```
#include <stdio.h>

int main( void )
{
    unsigned int c, sum = 0;

    for (c = 2; c <= 100; c += 2) {
        sum += c; //aggiunge c a sum
    }

    printf("%u\n", sum ); //stampa sum
}
```

# Esempi di uso dell'istruzione `for`

---



Buona pratica di programmazione

Se possibile, limitate le intestazioni delle istruzioni di controllo a una singola riga

```
for (c = 1; c <= 100; c++) {  
    ..  
}
```



# Esempio per applicazioni di calcolo

---

## Esempio

Una persona investe € 1000,00 in un conto corrente che frutta il 5% di interesse annuo

Supponendo che l'intero interesse resti depositato nel conto, calcolate e stampate la quantità di denaro nel conto alla fine di ogni anno per 10 anni

## Esempio per applicazioni di calcolo

---

Potete usare la seguente formula:

$$\textit{amount} = \textit{capital} * (1 + \textit{rate})^n$$

dove:

*capital* è la quantità iniziale di denaro investita

*rate* è il tasso annuale di interesse (0.05 per il 5%)

*n* è il numero degli anni

*amount* è la quantità di denaro in deposito alla fine dell'anno *n*

# Esempio per applicazioni di calcolo

---

Il C non include un operatore esponenziale

Tuttavia, possiamo usare a questo scopo la funzione `pow` della Libreria Standard

La funzione `pow(x, y)` calcola il valore di `x` elevato alla potenza `y`

Essa prende due argomenti di tipo `double` e restituisce un valore `double`

L'intestazione `<math.h>` va inclusa ogni volta che si usa una funzione della libreria `math` come `pow`

Nota: Con diversi compilatori Linux/UNIX C dovete includere l'opzione `-lm` (es. `gcc -o progr progr.c -lm`)

# Esempio per applicazioni di calcolo

---

La funzione `pow` richiede due argomenti `double`

Il file di intestazione `math.h` contiene informazioni che dicono al compilatore di convertire il valore di eventuali argomenti interi in una rappresentazione temporanea `double` prima di chiamare la funzione

Queste informazioni sono contenute nel prototipo della funzione `pow`

# Esempio per applicazioni di calcolo

---

Il tipo `double` è un numero in virgola mobile come `float`

Normalmente una variabile di tipo `double` memorizza un valore più elevato e con una precisione maggiore di `float`

Le variabili di tipo `double` occupano più memoria di quelle di tipo `float`

Per tutte le applicazioni eccetto quelle con uso intensivo di memoria, i programmatori professionisti generalmente preferiscono `double` a `float`

# Esempio per applicazioni di calcolo

---

```
#include <stdio.h>
#include <math.h>

int main( void ) {

    double amount, capital = 1000;
    double rate = 0.05;
    unsigned int n;

    /* stampa intestazioni delle colonne */
    printf( "%4s", "Year" );
    printf( "%21s\n", "Amount on deposit" );
```

# Esempio per applicazioni di calcolo

---

```
for (n = 1; n <= 10; n++) {  
    amount = capital * pow( 1.0+rate, n );  
    printf( "%4u%21.2f\n", n, amount );  
}  
}
```

# Esempio per applicazioni di calcolo

---

Esecuzione del programma:

Year	Amount on deposit
1	1050.00
2	1102.50
3	1157.63
4	1215.51
5	1276.28
6	1340.10
7	1407.10
8	1477.46
9	1551.33
10	1628.89



# Esempio per applicazioni di calcolo

---

Formattare output numerici

Nel programma è usato lo specificatore di conversione

`%21.2f` per stampare il valore della variabile

`amount`

Il 21 nello specificatore di conversione indica la larghezza del campo con cui il valore sarà stampato

Una larghezza di campo di 21 specifica che il valore stampato apparirà in 21 posizioni di stampa

## Esempio per applicazioni di calcolo

---

Il `.2` in `%21.2f` specifica la precisione (cioè il numero di posizioni decimali)

Se il numero di caratteri stampati è minore della larghezza del campo, il valore sarà automaticamente allineato a destra con spazi iniziali nel campo

Questo è particolarmente utile per allineare con la stessa precisione i valori in virgola mobile (in modo che i loro punti decimali si allineino verticalmente)

## Esempio per applicazioni di calcolo

---

Per allineare a sinistra un valore in un campo, mettete un `-` (segno meno) tra il `%` e la larghezza del campo

Il segno meno si può anche usare per allineare a sinistra numeri interi (come in `%-6d`) e stringhe di caratteri (come in `%-8s`)

# Esempio per applicazioni di calcolo

---

Attenzione alla formattazione quando usate `float` o `double`

Il seguente programma:

```
double x = 14.234;  
double y = 18.673;  
  
printf( "%.2f + %.2f = %.2f\n", x, y, x + y );
```

Produce: `14.23 + 18.67 = 32.91`

Ma  $14.23 + 18.67 = 32.90$  (!)

# Errori comuni con istruzione `for`

---

Determinate l'output del seguente programma:

```
01 #include <stdio.h>
02
03 int main( void )
04 {
05
06     for (int c = 1; c <= 3; c++) {
07         printf("%d\n", c );
08     }
09     printf("%d\n", c );
10 }
```

# Errori comuni con istruzione for


```
printf1.c: In function 'main':  
printf1.c:9:22: error: 'i' undeclared (first use in this function)  
   9 | printf( "i = %u\n", i );  
     |                     ^  
printf1.c:9:22: note: each undeclared identifier is reported only  
once for each function it appears in
```

```
01 #include <stdio.h>  
02  
03 int main( void )  
04 {  
05  
06     for (int c = 1; c <= 3; c++) {  
07         printf("%d\n", c );  
08     }  
09     printf("%d\n", c );  
10 }
```



# Errori comuni con istruzione `for`

Le variabili di controllo definite in un'intestazione del `for` esistono solo fino al termine del ciclo



```
01 #include <stdio.h>
02
03 int main( void )
04 {
05     int c;
06     for (c = 1; c <= 3; c++) {
07         printf("%d\n", c );
08     }
09     printf("%d\n", c );
10 }
```

1  
2  
3  
4

# Errori comuni con istruzione `for`

---

E' corretto il seguente programma?

```
#include <stdio.h>
int main( void ) {
    unsigned int i, j, m;
    for (i = 1, i <= 3, ++i) {
        for (j = 1; j <= 3; ++j) {
            for (k = 1; k <= 4; ++j) {
                Printf("%s", "*");
            }
            puts("");
        }
        puts("'");
    }
}
```



# Errori comuni con istruzione for

E' corretto il seguente programma?

```
#include <stdio.h>
int main( void ) {
    unsigned int i, j, m;
    for (i = 1, i <= 3, ++i) {
        for (j = 1; j <= 3; +j) {
            for (k = 1; k <= 4; ++j) {
                Printf("%s", "*");
            }
            puts("");
        }
        puts("");
    }
}
```

# Errori comuni con istruzione `for`

---

Versione corretta. Cosa produce?

```
#include <stdio.h>
int main( void ) {
    unsigned int i, j, k;
    for (i = 1; i <= 3; ++i) {
        for (j = 1; j <= 3; ++j) {
            for (k = 1; k <= 4; ++k) {
                printf("%s", "*");
            }
            puts("");
        }
        puts("");
    }
}
```

# Errori comuni con istruzione `for`

---

Output:

```
* * * *
```

```
* * * *
```

```
* * * *
```

```
* * * *
```

```
* * * *
```

```
* * * *
```

```
* * * *
```

```
* * * *
```

```
* * * *
```