



# **Laboratorio di Programmazione 1**

**Docente: dr. Damiano Macedonio**  
**Lezione 12 - 16/01/2014**

**1**

# Le Funzioni

- Le *funzioni* permettono di isolare un certo numero di istruzioni e di riutilizzarle in varie parti del codice.

- Es. di funzioni: `main`, `printf`, `scanf`, ecc...

```
int main(void) {  
    printf("Hello world!\n");  
    return 0;  
}
```

- Il nome `main` è riservato in C e indica da dove iniziare l'esecuzione del programma.
- Ogni programma C deve avere una funzione `main`, che può contenere l'invocazione di altre funzioni.

# Le Funzioni: Dichiarazione

- La dichiarazione di una funzione descrive il prototipo della funzione:
  - Chi la può chiamare.
  - Il tipo del valore che restituisce.
  - Il nome della funzione.
  - Il tipo di ogni argomento che riceve.

Non è necessario specificare il nome degli argomenti (presente solo nella definizione)
- La dichiarazione di una funzione deve sempre precedere il primo utilizzo della funzione.

```
int sum(int, int);
```

# Le Funzioni: Definizione

- La definizione di una funzione è composta da 2 parti:
  - La *dichiarazione del prototipo della funzione*.
    - Chi la può chiamare.
    - Il tipo del valore che restituisce.
    - Il nome della funzione.
    - Il nome e il tipo di ogni argomento che riceve.
  - Il *corpo della funzione* racchiuso tra parentesi graffe.
- Il nome di ciascun argomento è detto *nome del parametro formale* e può essere utilizzato per far riferimento all'argomento all'interno del corpo della funzione.

```
int sum(int a, int b) {  
    return a + b;  
}
```

# Le Funzioni: parola chiave `void`

La parola chiave `void` può essere utilizzata per indicare che:

- la funzione non ha valore di ritorno (se posto all'inizio della dichiarazione),
- oppure la funzione non ha argomenti (se posto tra le parentesi tonde).

# Le Funzioni: Esempio

```
#include <stdio.h>

void printMessage(void) {
    printf("Hello world!\n");
}

int main(void) {
    int i;
    for (i = 0; i < 5; i++)
        printMessage();
    return 0;
}
```

# Funzioni:

## Invocazione di Altre Funzioni

- All'interno di una funzione  $C$  è possibile invocare altre funzioni.
- Per poter utilizzare correttamente una funzione  $g$  all'interno di un'altra funzione  $f$  è necessario che:
  - $g$  sia *definita* prima di  $f$ , oppure
  - $g$  sia *dichiarata* prima di  $f$ 
    - Solitamente tale dichiarazione è fatta all'inizio del programma.
    - In questo modo il compilatore può verificare che la funzione sia usata correttamente.

# Funzioni:

## Invocazione di Altre Funzioni (sol. 1)

```
int abs(int x) {  
    if (x < 0)  
        x = -x;  
    return x;  
}
```

```
int main (void) {  
    int n;  
    printf("Inserisci un numero intero\n");  
    scanf("%i", &n);  
    Printf("%i e' il valore assoluto di %i",  
           abs(n), n );  
}
```



# Funzioni:

## Invocazione di Altre Funzioni (sol. 2)

```
int abs(int);
```

Dichiarazione

```
int main(void) {  
    int n;  
    ...  
    Printf ("%i e' il valore assoluto di %i",  
            abs(n), n);  
}
```

```
int abs(int x) {  
    if (x < 0)  
        x = -x;  
    return x;  
}
```

Definizione

# Funzioni: istruzione `return`

- Una funzione può contenere più istruzioni `return`.

```
int abs(int x) {  
    if(x < 0)  
        return -x;  
    else  
        return x;  
}
```

- Ogni volta che viene eseguita l'istruzione `return`, il controllo è restituito alla funzione chiamante.
  - Tutte le istruzioni che si trovano dopo il `return` sono ignorate.
- L'istruzione `return` può essere usata anche in una funzione che non restituisce un valore.
  - Forma più semplice: `return;`

# Le Funzioni: Variabili Locali

- Le variabili definite all'interno di una funzione sono dette *variabili locali automatiche*.
  - Automatiche perché sono create automaticamente ad ogni invocazione della funzione.
  - Locali perché il loro valore è locale alla funzione, cioè è visibile ed utilizzabile solo all'interno della funzione nella quale la variabile è stata definita.

```
int sum(int a, int b) {  
    int s;  
    s = a + b;  
    return s;  
}
```

s è una variabile locale alla funzione sum

# Le Funzioni: Variabili Locali

```
#include <stdio.h>
int abs(int x) {
    If (x < 0)
        x = -x;
}
```

Le modifiche apportate ad x all'interno della funzione abs non hanno alcun effetto sui valori delle variabili a, b e c. Quando una di queste variabili è passata alla funzione abs, il suo valore è automaticamente copiato nella variabile x, che è locale alla funzione abs.

```
int main(void) {
    int a = 5, b = -3, c = 0;
    Printf("%i e' il valore assoluto di %i", abs(a), a);
    Printf("%i e' il valore assoluto di %i", abs(b), b);
    Printf("%i e' il valore assoluto di %i", abs(c), c);
    return 0;
}
```