

Le funzioni

La Forma Generale di Funzione

Prof. G. Ascia

La forma generale di funzione è la seguente:

```
SpecificatoreTipo NomeFunzione (ElencoParametri)
{
    CorpoDellaFunzione;
}
```

- Lo SpecificareTipo specifica il tipo dei dati restituito dalla funzione.
- L'ElencoParametri è un elenco di nomi di variabili separate da virgole e preceduti dai rispettivi tipi.

Es. `int somma(int a, int b)`

- Tali parametri noti come *parametri formali* ricevono il valore degli argomenti passati (noti come *parametri attuali*) nel momento dell'invocazione della funzione.
- I parametri formali sono viste come delle variabili locali della funzione.

Tipi di funzioni

Prof. G. Ascia

- In C si possono realizzare due tipi di funzioni:
 - Funzioni che restituiscono un valore (funzioni di tipo matematico)
 - Funzioni che non restituiscono alcun valore (funzioni di tipo procedurali).
- Il primo tipo viene usato quando la funzione deve produrre un risultato sulla base del valore dei parametri di ingresso
 - Es. `float media(float x, float y)`
- Il secondo tipo viene usato quando la funzione deve realizzare una qualche operazione che non richiede la restituzione di un valore
 - Es.

```
void stampa ( int Vettore[N])
{ int i;
  for(i=0;i<N;i++) printf("%d",Vettore[i]);
}
```
- Il tipo `void` indica il fatto che non viene restituito alcun valore.

Esempio: definizione della funzione somma

Prof. G. Ascia

```
#include <stdio.h>
int somma (int x, int y)
{ int s;
  s=x+y;
  return s;
}
main() {
  int a,b,som;
  scanf ("%d %d", &a, &b);
  som=somma (a,b);
  printf ("%d", som);
}
```

La funzione `somma` restituisce un valore intero e ha due parametri formali (`x` e `y`) entrambi di tipo `int`.

- Il valore viene restituito attraverso l'istruzione **return**.
- La funzione `somma` viene invocata passando i parametri attuali **a** e **b**.

Modello della memoria durante l'attivazione di una funzione

Prof. G. Ascia

```
int somma (int x, int y)
{ int s;
  s=x+y;
  return s;
}
```

In seguito all'attivazione della funzione somma:

1. Vengono riservate le locazioni di indirizzo &x, &y e &s.
2. I valori dei parametri attuali vengono copiati nei parametri formali.
3. Viene calcolato il valore della variabile s.
4. Viene eseguita l'istruzione return e pertanto viene copiato il valore di som nella locazione relativa al parametro attuale som.
5. Le locazioni relative ai parametri formali e alle variabili locali vengono rilasciate.

Locazione		Indirizzo
70	a	&a=150
35	B	&b=152
105	som	&som=154
70	x	&x=200
35	y	&y=202
105	s	&som=230

The diagram illustrates the memory stack during function activation. It shows a table with columns for 'Locazione' (Location) and 'Indirizzo' (Address). The rows represent memory locations for variables 'a', 'B', 'som', 'x', 'y', and 's'. Arrows indicate the flow of data: from the 'som' location (address &som=154) to the 's' location (address &som=230), and from the 's' location to the 'som' location (address &som=154), representing the return value being passed back to the caller.

Funzioni senza parametri

Prof. G. Ascia

- Una funzione può anche essere senza parametri, in tal caso l'elenco dei parametri sarà vuoto.
- Tuttavia, anche se non vi è alcun parametro, è richiesta la presenza delle parentesi.

Es.

```
void menu()  
{ printf("Scegli una delle seguenti opzioni\n");  
  printf("i) Inserimento\n");  
  printf("c) Cancellazione\n");  
  printf("f) Fine\n");  
}
```

Chiamata di una funzione all'interno di un'altra funzione

Prof. G. Ascia

- È possibile chiamare una funzione all'interno di un'altra funzione
- È necessario che la funzione chiamata sia dichiarata o definita prima della funzione chiamante

```
#include <stdio.h>
float f2(int x2, float y2);

float f1(int x1, float y1)
{ float z;
  z=f1(x1,y1);
  return z;
}
main()
{ int a;
  float b,c;
  scanf("%d %f",&a,&b);
  c=f1(a,b);
}

float f2(int x2, float y2) { return x2+3*Y2; }
```

Esempio (errato) di scambio dei valori di variabili mediante la funzione scambia

Prof. G. Ascia

```
#include <stdio.h>
void scambia (int x, int y);
main ()
{ int a, b;
  scanf("%d %d",&a, &b);
  printf("%d %d\n",a,b);
  scambia(a,b);
  printf("%d %d\n",a,b);
}
void scambia (int x, int y)
{ int temp;
  temp=x;
  x=y;
  y=temp;
}
```

Locazione		Indirizzo
70	a	&a=160
35	b	&b=162

Non appena il programma viene mandato in esecuzione:

- Vengono associate le locazioni di indirizzo &a, &b alle variabili a e b;
- In seguito alla scanf vengono copiati i valori letti dalla tastiera nelle locazioni &a e &b
- Viene chiamata la funzione scambia

Esempio (errato) di scambio dei valori di variabili mediante la funzione scambia

Prof. G. Ascia

La chiamata della funzione scambia fa sì che:

- vengano riservate le locazioni di indirizzo &x, &y e &temp per i parametri formali x, y e per la variabile locale temp;
- vengano copiati i valori dei parametri attuali **a** e **b**, di indirizzo &a!=&x e &b!=&y, nei parametri formali **x** e **y**;

Locazione		Indirizzo
70	a	&a=160
35	b	&b=162
70		&x=200
35		&y=202
		&temp=230

Esempio (errato) di scambio dei valori di variabili mediante la funzione scambia

Prof. G. Ascia

La chiamata della funzione scambia fa sì che:

- vengano riservate le locazioni di indirizzo $\&x$, $\&y$ e $\&temp$ per i parametri formali x , y e per la variabile locale $temp$;
- vengano copiati i valori dei parametri attuali a e b , di indirizzo $\&a!=\&x$ e $\&b!=\&y$, nei parametri formali x e y ;
- vengono scambiati i valori di x e y

i_1 : $temp=x$;

i_2 : $x=y$;

i_3 : $y=x$;

Locazione		Indirizzo
70	a	$\&a=160$
35	b	$\&b=162$
70 35		$\&x=200$
35 i_2	i_1 70	$\&y=202$
	70 i_3	$\&temp=230$

Esempio (errato) di scambio dei valori di variabili mediante la funzione scambia

Prof. G. Ascia

- Al termine dell'esecuzione della funzione, i parametri formali e le variabili locali vengono rilasciate ovvero l'associazione variabile-indirizzo viene eliminata e il contenuto delle variabili viene perso
- Il valore dei parametri attuali non è stato modificato.

Locazione		Indirizzo
70	a	&a=160
35	b	&b=162
35		&x=200
70		&y=202
70		&temp=230

Passaggio dei parametri di una funzione

Prof. G. Ascia

In C il passaggio dei parametri nel momento in cui una funzione viene chiamata può avvenire in due modi:

- **Passaggio per valore**, in cui il valore del parametro attuale viene copiato nel parametro formale
 - In tutti gli esempi visti fino a questo momento è stato usato sempre il passaggio per valore;
- **Passaggio per indirizzo**, in cui viene copiato l'indirizzo del parametro attuale nel parametro formale
 - Il parametro formale deve essere dichiarato come puntatore allo stesso tipo del parametro attuale

Passaggio per valore

Prof. G. Ascia

- Questa modalità può essere utilizzata quando non è necessario modificare il valore del parametro attuale, ad esempio se la funzione deve calcolare un risultato o deve visualizzare uno o più valori.

Es.

```
float media(int x, int y)
{ return x+y; }
```

```
void stampa_intero (int x)
{ printf("%d", x); }
```

Passaggio dei parametri per indirizzo

Prof. G. Ascia

- Per permettere la modifica del valore del parametro attuale ad opera di una funzione,
 - non possiamo passare direttamente il valore del parametro (passaggio per valore),
 - è necessario passare il valore dell'indirizzo del parametro attuale (passaggio per indirizzo).
- Il passaggio per indirizzo viene realizzato in C passando il puntatore al parametro attuale.

Passaggio dei parametri per indirizzo

Prof. G. Ascia

```
#include <stdio.h>
void scambia (int *px, int *py);

main ()
{ int a, b;
  scanf("%d %d",&a, &b);
  printf("%d %d\n",a,b);
  scambia(&a,&b);
  printf("%d %d\n",a,b);
}

void scambia (int *px, int *py)
{ int temp;

/* In temp viene copiato il contenuto della locazione di indirizzo px */
   temp=*px;
/* Nella locazione di indirizzo px viene copiato il contenuto della locazione py */
   *px=*py;
/* Nella locazione di indirizzo py viene copiato il valore di temp */
   *py=temp;
}
```

Chiamata per indirizzo della funzione scambia

Prof. G. Ascia

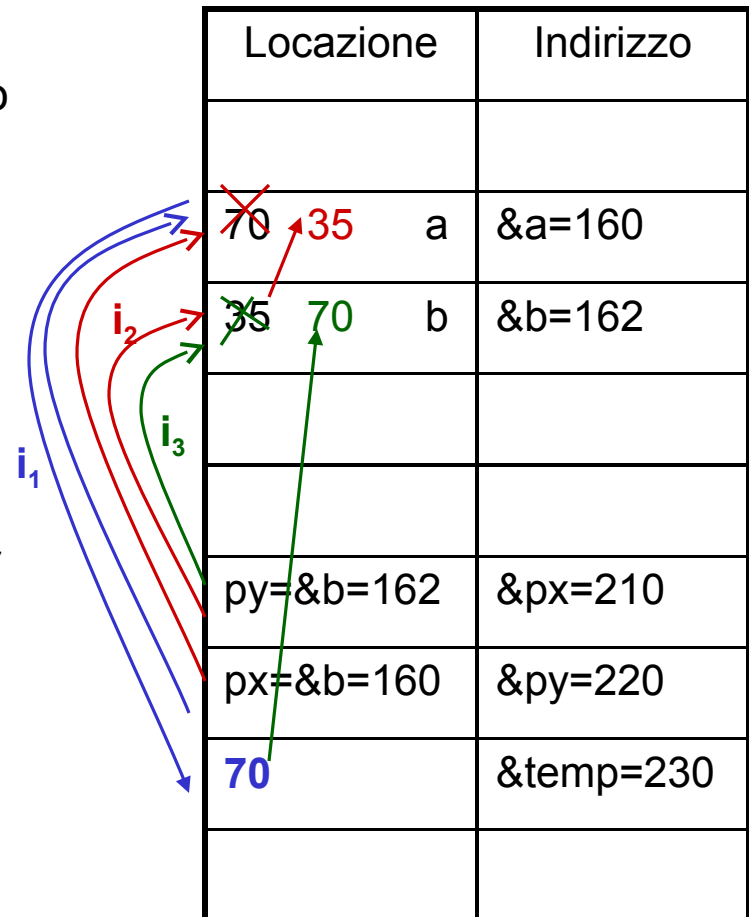
- La chiamata della funzione scambia fa sì che:
 - vengano riservate le locazioni di indirizzo &px, &py e &temp per i parametri formali px, py e per la variabile locale temp;
 - vengano copiati i valori degli indirizzi dei parametri attuali &a e &b, nei parametri formali **px** e **py**;

Locazione	Indirizzo
70 a	&a=160
35 b	&b=162
px=&a=160	&px=210
py=&b=162	&py=220
	&temp=230

Chiamata per indirizzo della funzione scambia

Prof. G. Ascia

- La chiamata della funzione scambia fa sì che:
 - vengano riservate le locazioni di indirizzo &px, &py e &temp per i parametri formali px, py e per la variabile locale temp;
 - vengano copiati i valori degli indirizzi dei parametri attuali &a e &b, nei parametri formali **px** e **py**;
 - Venga realizzato lo scambio tra a e b:
 1. Copia in temp del contenuto della locazione di indirizzo px=&a
i₁: temp=*px; /* =*(&a)=a */
 2. Copia nella locazione di indirizzo px del contenuto della locazione di indirizzo py
i₂: *px=*py;
 3. Copia nella locazione di indirizzo py del contenuto della variabile temp
i₃: *py=temp;



Chiamata per indirizzo della funzione scambia

Prof. G. Ascia

- Al termine dell'esecuzione della funzione, i parametri formali e le variabili locali vengono rilasciate ovvero l'associazione variabile-indirizzo viene eliminata e il contenuto delle variabili viene perso
- Il valore dei parametri attuali è stato modificato.

Locazione	Indirizzo
35 a	&a=160
70 b	&b=162
py=&b=162	&px=210
px=&b=160	&py=220
70	&temp=230

Esempio: lettura e visualizzazione degli elementi di un vettore (1)

Prof. G. Ascia

```
#include <stdio.h>

void leggi(int *px);
void stampa( int x);

main()
{ int V[5],i;
  /* passaggio per indirizzo per modificare il valore di V[i] */
  for(i=0;i<5;i++)
    leggi(&V[i]);
  /* passaggio per valore poiche' non si altera il valore di V[i] */
  for(i=0;i<5;i++)
    stampa(V[i]);
}

void leggi(int *px)
{ /* in px viene copiato l'indirizzo della variabile da leggere*/
  printf("Inserisci un numero: ");
  scanf("%d",px);
  /* nella scanf uso direttamente px e non &px
  poichè px e' gia' l'indirizzo della variabile da leggere */
}

void stampa( int x)
{ printf("%d\n",x);}
```

Esempio: lettura e visualizzazione degli elementi di un vettore (2)

Prof. G. Ascia

```
#include <stdio.h>

void leggi(int V[10]);
void stampa(int V[10]);

main()
{ int A[5],i;
  /* Nel caso di un vettore, usare come parametro attuale il
     nome del vettore equivale a passare l'indirizzo del 1°
     elemento */
    leggi(A);
    stampa(A);
}

void leggi(int V[10])
{ int i;
  for(i=0;i<10;i++)
  { printf("Inserisci un numero: ");
    scanf("%d",&V[i]); }
}

void stampa( int V[10])
{int i;
  for(i=0;i<10;i++) printf("%d\n",V[i]);
}
```

Vettore come parametro di una funzione

Prof. G. Ascia

- Nel caso in cui bisogna passare come parametro di una funzione un vettore, si possono usare indifferentemente le seguenti notazioni:
 1. `tipo_elemento V[dimensione_vettore]`
 2. `tipo_elemento V[]`
 3. `tipo_elemento *V`
- La ragione di questa equivalenza e' data dal fatto che, come detto in precedenza, il nome del vettore rappresenta il puntatore al primo elemento del vettore e che, al momento della chiamata della funzione, nel parametro formale viene copiato solo il valore dell'indirizzo del primo elemento e non viene fatto alcun controllo sulla dimensione del vettore.

Letture e visualizzazione degli elementi di un vettore (3)

Prof. G. Ascia

```
#include <stdio.h>

void leggi(int *V);
void stampa(int V[]);

main()
{ int A[5],i;
  /* Nel caso di un vettore, usare come parametro attuale il
   nome del vettore equivale a passare l'indirizzo del 1°
   elemento */
    leggi(A);
    stampa(A);
}

void leggi(int V[10])
{ int i;
  for(i=0;i<10;i++)
  { printf("Inserisci un numero: ");
    scanf("%d",&V[i]); }
}

void stampa( int *V)
{int i;
  for(i=0;i<10;i++) printf("%d\n",V[i]);
}
```