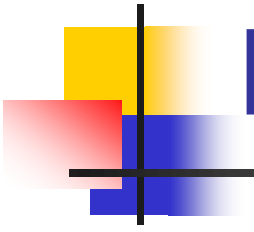


Funzioni in Linguaggio C

Concetti Chiave

- Dichiarazione
- Definizione
- Chiamata
- Passaggio dei parametri
- Valore di ritorno
- Tipo `void`
- Visibilità e mascheramento dei nomi

Le Funzioni

In C i sottoprogrammi sono detti ***funzioni***: a partire da uno o più valori in ingresso *ritornano* (o *restituiscono*) un valore al chiamante.

Figura 11.1 *La funzione come scatola nera.*

Un esempio di funzione è `abs`. Considerando la funzione come una **scatola nera**, tutto quello che dobbiamo sapere è che **immettendo** come argomento `i` ne **ritorna** il valore assoluto.

Figura 11.2 *La funzione* `abs` *come scatola nera che produce il valore assoluto di* `i`.

Le Funzioni

```
#include<stdio.h>

unsigned long esponente;
double base, p;

int main(void){
        printf("\nInserisci la base ");
        scanf("%lf",&base);
        printf("\nInserisci l'esponente ");
        scanf("%lu",&esponente);
        p=1.0;

        while (esponente-->0)
                p*=base;
        printf("\nPotenza = %e",p);
}
```

```
#include<stdio.h>

unsigned long esponente;
double base, p;
void potenza (void);

int main(void){
        printf("\nInserisci la base ");
        scanf("%lf",&base);
        printf("\nInserisci l'esponente ");
        scanf("%lu",&esponente);
        potenza();
}

void potenza(void){
   p=1.0;
   while (esponente-->0)
                p*=base;
   printf("\nPotenza = %e",p);
}
```

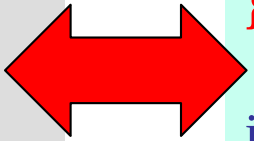
Due sintassi equivalenti

```
#include<stdio.h>

unsigned long esponente;
double base, p;
void potenza (void);

int main(void){
        printf("\nInserisci la base ");
        scanf("%lf",&base);
        printf("\nInserisci l'esponente ");
        scanf("%lu",&esponente);
        potenza();
}

void potenza(void){
   p=1.0;
   while (esponente-->0)
            p*=base;
   printf("\nPotenza = %e",p);
}
```



```
#include<stdio.h>

unsigned long esponente;
double base, p;

void potenza(void){
   p=1.0;
   while (esponente-->0)  p*=base;
   printf("\nPotenza = %e",p);
}

int main(void){
        printf("\nInserisci la base ");
        scanf("%lf",&base);
        printf("\nInserisci l'esponente ");
        scanf("%lu",&esponente);
        potenza();
}
```

Valore Ritorno e Variabili Locali

```
#include<stdio.h>

unsigned long esponente;
double base, p;

void potenza(void){
    p=1.0;
    while (esponente-->0)  p*=base;
    printf("\nPotenza = %e", p);
}

int main(void){
        printf("\nInserisci la base ");
        scanf("%lf",&base);
        printf("\nInserisci l'esponente ");
        scanf("%lu",&esponente);
        potenza();
}
```

```
#include<stdio.h>

unsigned long esponente;
double base;

double potenza(void){
    double p=1.0;
    while (esponente-->0) p*=base;
    return (p);
}

int main(void){
        printf("\nInserisci la base ");
        scanf("%lf",&base);
        printf("\nInserisci l'esponente ");
        scanf("%lu",&esponente);
        printf("\nPotenza = %e",potenza());
}
```

Due sintassi equivalenti

```
#include<stdio.h>

unsigned long esponente;
double base;

double potenza(void){
   double p=1.0;
   while (esponente-->0) p*=base;
   return (p);
}

int main(void){
      printf("\nInserisci la base ");
      scanf("%lf",&base);
      printf("\nInserisci l'esponente ");
      scanf("%lu",&esponente);
      printf("\nPotenza = %e",potenza());
}
```

```
#include<stdio.h>

unsigned long esponente;
double base;
double potenza (void);

int main(void){
      printf("\nInserisci la base ");
      scanf("%lf",&base);
      printf("\nInserisci l'esponente ");
      scanf("%lu",&esponente);
      printf("\nPotenza = %e",potenza());
}

double potenza(void){
   double p=1.0;
   while (esponente-->0)
            p*=base;
   return (p);
}
```

Parametri Formali

```
#include<stdio.h>

unsigned long esponente;
double base;
double potenza (void);

int main(void){
      printf("\nInserisci la base ");
      scanf("%lf",&base);
      printf("\nInserisci l'esponente ");
      scanf("%lu",&esponente);
      printf("\nPotenza = %e",potenza());
}

double potenza(void){
   double p=1.0;
   while (esponente-->0)
            p*=base;
   return (p);
}
```

```
#include<stdio.h>

unsigned long esponente1, esponente2;
double base1, base2;
double potenza (double, unsigned long);

int main(void){
      base1=3.0;
      esponente1=3;
      base2=2.0;
      esponente2=4;

      printf("\nPotenza = %f",potenza(base1,esponente1));
      printf("\nPotenza = %f",potenza(base2,esponente2));
      printf("\nPotenza = %f",potenza(4.0,4));
}

double potenza(double b, unsigned long e){
   double p=1.0;
   while (e-->0)
            p*=b;
   return (p);
}
```

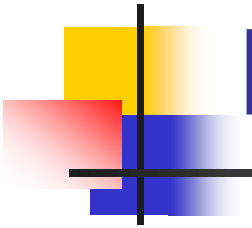
L'altra Sintassi

```
#include<stdio.h>


double potenza(double b, unsigned long e){
    double p=1.0;
    while (e-->0)
                p*=b;
    return (p);
}

int main(void){
     unsigned long esponente1=3, esponente2=4;
     double base1=3.0, base2=2.0;

      printf("\nPotenza = %f",potenza(base1,esponente1));
      printf("\nPotenza = %f",potenza(base2,esponente2));
      printf("\nPotenza = %f",potenza(4.0,4));
}
```



Le Funzioni

- Le funzioni possono essere definite in due modi:
  - Modo1
    - Dichiarazione del prototipo prima del int main(void) e dopo le direttive di precompilazione
    - Definizione della funzione dopo il int main(void)
  - Modo 2
    - Definizione completa prima del int main(void) o in un file da includere

Le Funzioni

- Modo 1
- Sintassi di dichiarazione, detta *prototyping*:

  ***tipoRitorno nomeFunz*(*tipoPar1*,...,*tipoParN*);**

- Sintassi di definizione:

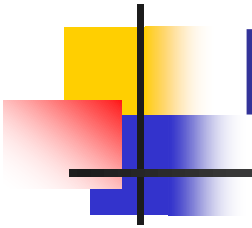
***tipoRitorno nomeFunz*(*tipoPar1 Par1*,...,*tipoParN ParN*){**

Definizione Variabili Locali;

Istruzioni;

return(valore); oppure return valore;

}



Le Funzioni

- Modo 2
- Sintassi di definizione:

***tipoRitorno nomeFunz*(*tipoPar1 Par1*,...,*tipoParN ParN*)**{

Definizione Variabili Locali;

Istruzioni;

return(valore); oppure return valore;

}

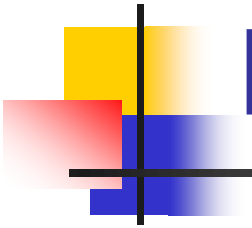
Le Funzioni

- In entrambi i casi, se il valore di ritorno non esiste o se non esistono i paramentri formali, si usa la parola chiave void

void nomeFunz **(*tipoPar1 Par1,...,tipoParN ParN*)**

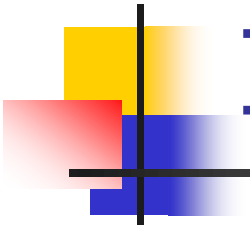
tipoRitorno nomeFunz **(void)**

- Se il valore di ritorno non esiste, allora return non deve essere usato



Le Funzioni

- Come si richiamano le funzioni ? Si utilizza il nome della funzione e si specifica l'**elenco dei parametri effettivi**
- Se non esistono parametri formali, la funzione viene richiamata con l'elenco dei parametri formali vuoto ()
  - Esempio: getchar()



Il main

- Il main è la prima funzione che viene eseguita
- Se definita: int main(void), ritorna un numero intero
- C'e' la convenzione che se il main ritorna 0, significa "nessun problema", altrimenti si torna un numero diverso da 0
- Se non viene indicato il return(valore intero), ritorna sempre 0
- Dunque se si deve tornare 0, SI PUO' NON specificare return(0) alla fine del main

Le Funzioni- Parametri Attuali/Formali

- La corrispondenza tra parametri formali e attuali è posizionale
- Questo può innescare errori di conversione
- I parametri formali assumono come valore iniziale quello dei parametri passati dal chiamante (esempio main)
- **Le modifiche ai parametri formali non alterano i parametri corrispondenti del chiamante**
  - Nell'esempio, la variabile esponente non viene modificata, ma solo il parametro formale e

```
#include<stdio.h>

unsigned long esponente=4;
double base=3.0;
double potenza(double , unsigned long );

int main(void){
     printf("\nPotenza = %f",potenza(base,esponente));
}

double potenza(double b, unsigned long e){
   double p=1.0;
   while (e-->0)
            p*=b;
   return (p);
}
```

3.0 4

Le Variabili Locali alle Funzioni

- Una variabile dichiarata in una funzione (variabile *locale*) ha visibilità dal punto di dichiarazione alla fine della funzione in cui è contenuta.
- Anche il main potrebbe avere variabili locali
- **Le variabili locali assumono valore iniziale casuale**
- **La dichiarazione di una variabile in una funzione può *nascondere* la dichiarazione di una variabile globale con lo stesso nome.**

Le Variabili Locali alle Funzioni

- La dichiarazione di una variabile in una funzione può *nascondere* la dichiarazione di una variabile globale con lo stesso nome.
- Nell'esempio si noti la variabile i

```
#include<stdio.h>

unsigned int i,n;

double prova(unsigned int a){
   double i=10.0;

   i*=a;
   return (i);
}

int main(void){

      printf("\nInserisci n  ");
      scanf("%u",&n);
      for (i=0;i<n;i++)
          printf("\nValore = %f",prova(i));
}
```

Passaggio di Parametri Vettore

- i parametri formali passati per valore relativamente ad un vettore, si esprimono nelle tre forme equivalenti:

tipobase * nome_parametro;

→ **tipobase nome_parametro[];**

tipobase nome_parametro[dimensione];

Passaggio di Parametri Vettore

```
#include<stdio.h>
#define N 5
#define M 100
float vett[N],  vett2[M];


void riempi(float v[], int dim){
   unsigned long i;
   for (i=0; i<dim; i++) {
        printf("Inserisci l'elemento di indice %u ",i);
        scanf("%f",&v[i]);
   }
}

void stampa(float v[], int dim){
   unsigned long i;
   for (i=0; i<dim; i++)
        printf("\nElemento di indice %u = %f ",i,v[i]);
}
```

```
float max(float v[], int dim){
   unsigned long i;
   float max=v[0];

   for (i=1; i<dim; i++)
        if (v[i]>max) max=v[i];

   return max;
}

int main(void)
{
  riempi(vett,N);
  riempi(vett2,M);
  stampa(vett,N);
  stampa(vett2,M);
  printf("\nMassimo elemento = %f ",max(vett,N));
  printf("\nMassimo elemento = %f ",max(vett2,M));
}
```

Passaggio di Parametri Vettore

```
#define FFLUSH while (getchar()!='\n')

void LeggiStringa(char s[], unsigned long dim){
  unsigned long i;

  for (i=0; i<dim-1;i++)
       if ((s[i]=getchar())=='\n') break;
  if (i==dim-1) FFLUSH;
  s[i]='\0';
}
```