Fondamenti di Informatica

Ingegneria Clinica

Esercitazione 15/11/2010

Prof. Raffaele Nicolussi

FUB - Fondazione Ugo Bordoni

Via del Policlinico, 147 - 00161 Roma

Docente	**Raffaele Nicolussi**	***rnicolussi@fub.it*** ***0654803323***
Lezioni **Aula 54 (ex aula 4)** **Via del Castro Laurenziano, 7**	**Lunedì, Giovedì, Venerdì**	**12:00 – 13:30**
Esercitazioni **Aula 15** **Via Tiburtina, 205**	**Lunedì**	**14:00 – 17:30**
Ricevimento:	**Per appuntamento**	**in FUB, per email, per telefono**
Sito web:	**http://w3.uniroma1.it/IngClinFondinf**	

Le Funzioni

il C considera "main()" come una funzione.

La forma generale di una funzione e':

```
returntype function_name (param1, param2, ...)
{
      local variables
      function code (C statements)
}
```

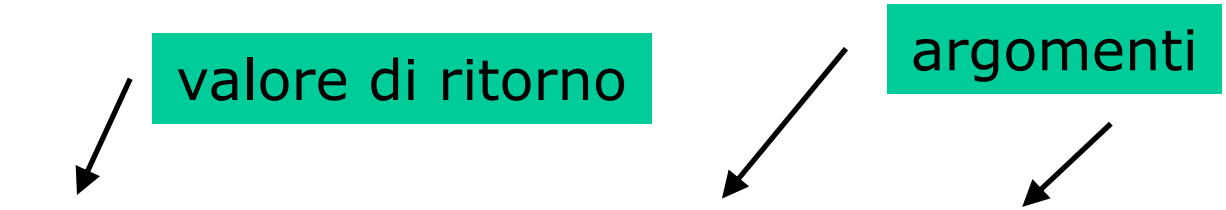
Tipo di ritorno

Se manca la definizione del tipo della funzione ("`returntype`", tipo della variabile di ritorno della funzione), il C assume che il ritorno della funzione sia di tipo integer (`int`);

ATTENZIONE: questo può essere una delle cause di problemi nei programmi.

Esempio

Esempio di una funzione che calcola la media tra due valori:




```
float calcolamedia(float a, float b)
{
      float media;
      media = (a + b) / 2;
      return(media);
}
```

Eseguire una funzione

Per richiamare tale funzione si procede nel seguente modo:

```
main()
{
      float a = 10, b = 25, risultato;
      risultato = calcolamedia(a, b);
      printf("Valore medio = %f\n",
      risultato);
}
```

Nota: l'istruzione "`return`" ritorna il risultato della funzione al programma principale.

Programma calcola media


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

float calcolamedia(float a, float b)
{
        float media;
        media = (a + b) / 2;
        return(media);
}

main()
{
        float a = 10, b = 25, risultato;
        risultato = calcolamedia(a, b);
        printf("Valore medio = %f\n",
        risultato);
}
```

Domanda

Cosa succede se inverto l'ordine delle funzioni?

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

main(){
        float a = 10, b = 25, risultato;
        risultato = calcolamedia(a, b);
        printf("Valore medio = %f\n",
        risultato);
}
float calcolamedia(float a, float b){
        float media;
        media = (a + b) / 2;
        return(media);
}
```

Risposta

Il compilatore non può sapere cos'è "calcolamedia"

la funzione prima va dichiarata

poi va usata

```
#include <stdio.h>
#include <stdlib.h>
float calcolamedia(float a, float b);
main(){
        float a = 10, b = 25, risultato;
        risultato = calcolamedia(a, b);
        printf("Valore medio = %f\n", risultato);
}
float calcolamedia(float a, float b){
        float media;
        media = (a + b) / 2;
        return(media);
}
```

Prototipi di funzioni

Prima di usare una funzione, il C deve riconoscere il tipo di ritorno e il tipo dei parametri che la funzione si aspetta.

Lo standard ANSI del C ha introdotto un nuovo e migliore metodo per fare questa dichiarazione rispetto alle vecchie versioni di C (ricordiamo che le moderne versioni del C aderiscono allo standard ANSI).

Va fatta una dichiarazione

Dichiarazione di funzione

L'importanza della dichiarazione e' doppia:

- viene fatta per avere un codice sorgente più strutturato e perciò facile da leggere ed interpretare;

- permette al compilatore C di controllare la sintassi delle chiamate di funzioni.

Dichiarazione di funzione (2)

Fondamentalmente, se una funzione e' stata definita prima di essere usata (call) allora è possibile semplicemente usare la funzione.

Nel caso contrario, è obbligatorio dichiarare la funzione; la dichiarazione stabilisce in modo semplice il ritorno della funzione ed il tipo dei parametri utilizzati da questa.

E' buona norma (e solitamente viene fatto) dichiarare tutte le funzioni all'inizio del programma, sebbene non sia strettamente necessario.

Dichiarazione di funzione (3)

Per dichiarare un prototipo di funzione bisogna semplicemente stabilire il ritorno della funzione, il nome della funzione e tra le parentesi elencare il tipo dei parametri nell'ordine in cui compaiono nella definizione di funzione.

Ad esempio:

```
int strlen(char[]);
```

Questo dichiara che una funzione di nome "strlen" ritorna un valore integer ed accetta una singola stringa come parametro.

Una scorciatoia

le funzioni e le variabili possono essere dichiarate sulla stessa linea di codice sorgente.

Questa procedura era molto più diffusa nei giorni del pre-ANSI C; da allora le funzioni solitamente vengono dichiarate separatamente all'inizio del programma.

La prima procedura risulta ancora perfettamente valida, purchè venga rispettato l'ordine in cui gli oggetti compaiono nella definizione della funzione.

```
int length, strlen(char[]);
```

dove "length" e' una variabile, e "strlen" e' la funzione (come nell'esempio precedente).

Funzioni "void"

Se non si vuole ritornare alcun valore da una funzione è sufficiente dichiararla di tipo `void` ed omettere il `return`.

```
void cento(){
     int loop;
     for (loop = 0; loop < 10; loop++);
     printf("%d\n", loop * loop);
}

main(){
     cento();
}
```

Nota: è obbligatorio mettere le parentesi () dopo il nome della funzione anche se non ci sono parametri.

Esercizio 1

Scrivere un programma che, acquisiti da stdin 3 valori reali (float) a, b, c, usi due funzioni per determinare, rispettivamente, minimo e massimo tra i valori acquisiti, e visualizzi su standard output i risultati restituiti dalle funzioni

Soluzione esercizio 1

```
/* definizione funzione min3 */
float min3( float x , float y , float z )
{
  float min ;

  if ( x < y )
    min = x ;
  else
    min = y ;

  if ( z < min )
    return z ;
  else
    return min ;
}
```

Soluzione esercizio 1

```
/* definizione funzione max3 */
float max3(float x , float y , float z )
{
 float max ;

  if ( x > y )
     max = x ;
  else
     max = y ;

  if ( z > max )
     return z ;
  else
     return max ;
}
```

Soluzione esercizio 1


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

/* prototipo funzione min3 */
float min3(float , float , float ) ;

/* prototipo funzione maxi3 */
float max3(float , float , float ) ;

int main()
{
 float a , b , c ;

   printf( "\nInserire a (float): " ) ;
   scanf( "%f" , &a ) ;
   printf( "\nInserire b (float): " ) ;
   scanf( "%f" , &b ) ;
   printf( "\nInserire c (float): " ) ;
   scanf( "%f" , &c ) ;
   printf( "\nminimo( %f , %f , %f ) = %f\n" , a , b , c , min3( a , b , c ) ) ;
   printf( "\nmassimo( %f , %f , %f ) = %f\n" , a , b , c , max3( a , b , c ) ) ;
   system( "pause" ) ;
   return 0 ;
}
```

Esercizio 2

Progettare una funzione che, ricevuti un carattere (char) c e due valori reali (float) x e y, restituisca il reale (float) corrispondente all'operazione specificata da c ('+' sta per somma, '-' sta per differenza, '*' sta per prodotto, '/' sta per divisione) eseguita tra x e y. In caso di operatore non valido (carattere c diverso dai quattro valori specificati sopra), la funzione restituirà il valore 0.0 e non effettuerà alcuna operazione tra x e y.

Inserire poi la funzione in un programma di prova, che provveda all'acquisizione da standard input degli argomenti della funzione, alla chiamata della funzione e alla visualizzazione su standard output del valore da questa restituito

Soluzione esercizio 2


```
/* Definizione funzione operation */
float operation( char op , float opnd1 , float opnd2 )
{
  if ( op == '+' )
    return ( opnd1 + opnd2 ) ;

  else if ( op == '-' )
    return ( opnd1 - opnd2 ) ;

  else if ( op == '*' )
    return ( opnd1 * opnd2 ) ;

  else if ( op == '/' )
    return ( opnd1 / opnd2 ) ;

  else
    return 0.0 ;
}
```

Soluzione esercizio 2


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

/* Prototipo funzione operation */
float operation( char , float , float ) ;

int main()
{
  float x , y ;
  char op ;

  printf( "Inserire carattere operatore ('+' , '-' , '*' o '/'): " ) ;
  scanf( "%c" , &op ) ;

  printf( "\nInserire 1o operando (float): " ) ;
  scanf( "%f" , &x ) ;

  printf( "\nInserire 2o operando (float): " ) ;
  scanf( "%f" , &y ) ;

  printf( "\n%f %c %f = %f\n\n" , x , op , y , operation( op , x , y ) ) ;

  system( "pause" ) ;
  return 0 ;
}
```

Upgrade esercizio 2

- Modificare il precedente esercizio sostituendo le IF con una SWITCH
- Aggiungere le operazioni:
  - ^ elevamento a potenza
  - % modulo
  - …

Esercizio 3: somma di due frazioni

Scrivere un programma per **calcolare la somma di due frazioni**.

Il programma deve:

1. ricevere in input quattro interi (num1, den1, num2, den2) corrispondenti ai numeratori e ai denominatori delle frazioni da sommare
2. fornire in output numeratore e denominatore della frazione (eventualmente semplificata) risultante dalla somma dei due numeri.

Esercizio 3: funzioni del programma

1. funzione massimoComunDivisore che calcola il massimo comun divisore di due interi
2. funzione minimoComuneMultiplo che calcola il minimo comune multiplo di due interi
3. funzione main

Esercizio 3: massimoComunDivisore

Calcolare il massimo comune divisore tra due numeri **a** e **b** in ingresso.

- **Algoritmo di Euclide**: permette di determinare il M.C.D. di due numeri naturali in modo molto più efficiente che non elencando tutti i divisori dei due numeri considerati.

```
do {
    r = a % b;
    a = b;
    b = r;
} while (r != 0)
```

Il massimo comun divisore è **a**

Esempio a = 12 b=15

1)	r=12	a= 15 b=12
2)	r=3	a=12 b=3
3)	r=0	a=3 b=0

MCD = 3

Esercizio 3: minimoComuneMultiplo

- La funzione **minimoComuneMultiplo** utilizza la funzione **massimoComunDivisore**

- Il minimo comune multiplo è uguale al prodotto dei due numeri diviso il loro massimo comun divisore, **mcm(a,b) = a*b / MCD(a,b)**

Esercizio 3: main

La funzione **main**:

- legge l'input
- calcola il risultato usando gli altri moduli
- verifica se il risultato deve essere semplificato
- stampa il risultato

Per effettuare la somma di due frazioni si dovrà

1. calcolare il mcm dei due denominatori che costituisce il denominatore della frazione_somma
2. calcolare i numeratori delle frazioni equivalenti a quelle date con il nuovo denominatore
3. sommare i nuovi numeratori per avere il numeratore della frazione_somma

Per semplificare una frazione si dovrà calcolare il MCD tra numeratore e denominatore

Soluzione esercizio 3


```
int minimoComuneMultiplo (int n1, int n2)
{
  return n1 * n2 / massimoComunDivisore(n1, n2);
}

int massimoComunDivisore (int n1, int n2)
{
  int resto;

  do {
    resto = n1 % n2;
    n1 = n2;
    n2 = resto;
  } while (resto != 0);

  return n1;
}
```

Soluzione esercizio 3


```
void stampaRisultato (int n1, int d1, int n2, int d2, int ns, int ds)
{
  printf("%d/%d + %d/%d = %d/%d\n", n1, d1, n2, d2, ns, ds);
}
```

Soluzione esercizio 3


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

int massimoComunDivisore (int n1, int n2);

int minimoComuneMultiplo (int n1, int n2);

void stampaRisultato (int n1, int d1, int n2, int d2, int ns, int ds);
/* scrive il risultato della somma */
```

Soluzione esercizio 3

$$\frac{3}{2} + \frac{5}{4} = \frac{3 * (4 / 2) + \ldots}{4}$$

```
int main ( )
{
 int num1, den1, num2, den2 ;
 int num_sum, den_sum, semplifica;

 printf("Inserire quattro interi [num1 den1 num2 den2]\n");
 scanf("%d%d%d%d", &num1, &den1, &num2, &den2);

 den_sum = minimoComuneMultiplo(den1, den2);      /* denominatore comune */
 num_sum = num1*(den_sum/den1) + num2*(den_sum/den2);

 semplifica = massimoComunDivisore(num_sum, den_sum);
 if (semplifica != 1) {
  num_sum /= semplifica;
  den_sum /= semplifica;
 }
 stampaRisultato(num1, den1, num2, den2, num_sum, den_sum);

 system("pause");

 return 0;
}
```