Funzioni predefinite: La *Standard Library* del C

Tutte le versioni ANSI C mettono a disposizione del programmatore un insieme di ***funzioni predefinite*** che costituiscono la ***Standard Library*** del C.

Tali funzioni sono racchiuse in vari file. I prototipi di tali funzioni sono riportate in ***file header*** (estensione ***.h***)

La compilazione di un programma che utilizzi tali funzioni richiede l'inclusione del file header della libreria che si vuole utilizzare:

Esempio:

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

Funzioni standard che riguardano:

- input/output *stdio.h*
- funzioni matematiche *math.h*
- gestione dinamica della memoria *stdlib.h*
- gestione di caratteri e stringhe *string.h*
- ricerca ed ordinamento *stdlib.h*
- interazione con il sistema operativo

Funzioni di libreria per la gestione di stringhe:

```
#include <string.h>
```

Funzione strlen:

restituisce la lunghezza di una stringa di caratteri

```
int strlen(char s[ ])
{
   int j;
   for (j = 0; s[j] != '\0'; j++) ;
                          /* scansione */
   return j;
}
```

Funzione strcmp:

confronta due stringhe di caratteri, s1 e s2, e restituisce un valore:

< 0 se s1 < s2

0 se s1 == s2

> 0 se s1 > s2

```
int strcmp(unsigned char s1[],unsigned char
s2[])
{
   int j = 0;
   while ( s1[j] && s2[j] && s1[j] == s2[j])
         j++;
   return (s1[j] - s2[j]);
}
```


Funzione strcpy:

copia una stringa (sorgente) in un'altra (destinazione) e restituisce l'***indirizzo*** della stringa destinazione

```
char *strcpy(char dest[ ],char src[ ])
{
   int j = 0;
   do
      dest[j] = src[j];
   while (src[j++])
   return dest;     /* return &dest[0] */
}
```


Argomenti delle linee di comando:

Anche la funzione ***main*** può avere parametri.

A ***main*** vengono sempre passati due parametri, corrispondenti a due ***parametri formali*** denominati ***argc*** ed ***argv***:

- ***argc***, di tipo intero, rappresenta il numero degli argomenti effettivamente passati al programma nella linee di comando con cui si invoca la sua esecuzione;
- ***argv***, vettore di stringhe, ciascuna delle quali contiene un argomento. Gli argomenti sono memorizzati nel vettore nell'ordine con cui sono dati dall'utente.

Per convenzione, ***argv[0]*** contiene il ***nome del programma stesso***.

Quindi argc vale sempre almeno 1.

I parametri effettivi iniziano quindi da ***argv[1]***.

prog arg1 arg2 ... argN

Quindi:

argv[0]="prog"

argv[1]="arg1"

argv[2]="arg2"

...

argv[N]="argN"

Per convenzione, ***argv[argc]*** vale ***NULL***.

Esempio:

Programma che stampa i suo argomenti, escluso il nome del programma stesso.

```
#include <stdio.h>

main(int argc, char *argv[])
{
   int i;

   for(i=1; i<argc; i++)
      printf("%s%s", argv[i],
          (i<argc-1) ? "\t" : "\n");

   return 0;
}
```

Invocazione:

```
esempio a b c zeta
```

Cosa stampa?

0 Esercizio 3.0:

Definire una nuova funzionalità, realizzata come programma C, che dato come argomento il nome di un file (di testo), ne visualizzi il contenuto a pagine (20 righe alla volta).

```
#include <stdio.h>
#define NL 20

void main (void)
{int n;
 char RIS='s';
 char S[30];
 FILE *f;
 if (argc==2)
    {f=fopen(argv[1],"rt");
     if (f==NULL)
        {printf("Errore di apertura");
         return;}
     while((!feof(f)&&(RIS='s'))
       {for(n=1;(n<=NL)&&(!feof(f));n++)
          {fscanf(f,"%s",S);
           printf("%s",S);}
        if (n>NL)
          {printf("Continua (s/n)?");
           scanf("%c",&RIS);}
       }
     }
 else printf("Num. argomenti non corretto");
}
```


Funzioni passate come parametri:

Procedure e funzioni possono apparire come ***parametri*** di altre procedure e funzioni e possono essere restituite (attraverso un puntatore) come ***risultato*** di una funzione (legame per **indirizzo**).

Per avere parametri procedura o funzione, occorre specificare un ***parametro formale*** di tipo funzione, come segue:

```
tipo1 F1 (tipo2  ( *F2)(lista-par-formali-F2),
                                altri-par-form-F1       )
```

intestazione della funzione F1 che ha la funzione F2 come parametro formale.

Nel corpo di F1 potrà esserci la chiamata:

```
k=F2(...) /* con parametri attuali opportuni */
```

Un ***parametro formale*** funzione è specificato indicando un ***nome*** (puntatore), la ***lista dei parametri formali*** ed il ***tipo di risultato***.

Il ***parametro attuale*** con cui si invoca F1 deve essere un identificatore di funzione con la stessa descrizione dei parametri e lo stesso tipo di risultato di F2.

Il C considera l'***identificatore*** di una funzione come un ***puntatore*** al suo codice.

Esempio

```
#include <math.h>

double fun (double x)/*funzione reciproco */
   { return 1.0 / x;}

/* double sin (double);   libreria math.h*/

double sommaquadratif (
   double (* f)( double par), int m, int n)
{ int k;
  double somma;
  somma = 0;
  for (k=m; k <= n; k++)
       somma = somma + (* f)(k) * (* f)(k);
  return somma;
}

 double sommacubif (
     double (* f) (double par),int m, int n)
{ int k;  double cubo;
  cubo= 0;
  for (k=m; k <= n; k++)
     cubo=cubo + (* f)(k) * (* f)(k) *
                 (* f)(k);
  return cubo;
}

main ()
{int a, b, c, i;
 printf (" Inversi %.7f\n",
          sommaquadratif (fun,1, 10000));
printf (" Seni %.7f\n",
          sommaquadratif (sin, 2, 13));
printf (" Inversi %.7f\n",
          sommacubif (fun, 1, 10000));
printf (" Seni %.7f\n",
          sommacubif (sin, 2, 13));}
```

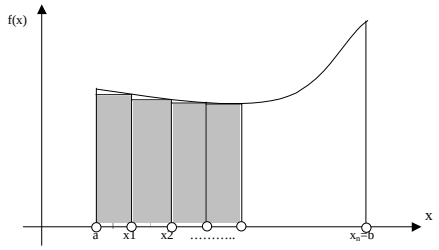
0 Esercizio 3.1:

Si realizzi un programma che calcola l'integrale di una funzione reale di una variabile reale in un intervallo dato [a,b].

Soluzione

Il problema del calcolo dell'integrale di una funzione può essere risolto in modo approssimato con il *metodo dei rettangoli*.

Data la funzione integranda f, si suddivide l'intervallo di integrazione [a,b] in N sotto-intervalli di uguale ampiezza h. Il valore di h è quindi dato da (b-a)/N.



Il contributo I_i di ciascun sotto-intervallo nel calcolo dell'integrale è quindi: $I_i = h.\ f(x_i)$.

L'integrale I si calcola quindi come sommatoria dei contributi di tutti gli sotto-intervalli:

$I=\Sigma_{i=1..N}\ I_i = \Sigma_{i=1..N}\ h.\ f(x_i) = h.\ \Sigma_{i=1..N}\ f(x_i)$


```
float rettangoli(float (*func)(float),
                 float a, float b, int n)
/*  func è la funzione integranda
    a,b sono gli estremi dell'intervallo,
     n è il numero di sotto-intervalli */
{
  float x,sum,h;
  int j;
  h=(b-a)/n;
  x=a;
  for (sum=0.0,j=0;j<n;j++,x+=h)
        sum += (*func)(x);
  sum=h*sum;
  return sum;
}
```

Vediamo infine, a titolo di esempio, un programma che usa la funzione **rettangoli** per calcolare l'integrale di due funzioni polinomiali:

```
#include <stdio.h>

float rettangoli(float (*func)(float),
                 float a, float b, int n);
float quadra(float x);
float cubo(float x);
```



```
main()
{ float A,B;
  printf("Estremi intervallo [a,b]?: ");
  scanf("%f%f", &A, &B);
  printf("Metodo dei rettangoli:\n\n");
  printf("Integrale della funzione x**2:");
  printf("%f\n",
       rettangoli(quadra, A, B, 1000));
  printf("Integrale della funzione x**3:");
  printf("%f\n",
       rettangoli(cubo, A, B, 1000));
}

float rettangoli(float (*func)(float),
                 float a, float b, int n)
{ . . . }

float quadra(float x)
{   return x*x;}

float cubo(float x)
{   return x*x*x;}
```

Il **main**, dopo avere acquisito gli estremi di integrazione A e B, calcola (chiamando la **rettangoli**) e stampa il valore di $\int x^2 dx$ e $\int x^3 dx$ nell'intervallo [**A, B**] dato.

0 Esercizio 3.2:

Scrivere un programma C definendo una funzione ***sigma*** che dati in ingresso un parametro funzione *F*, ad argomento intero e valore intero, e due estremi (interi) *Inferiore*, *Superiore*, calcoli la sommatoria:

$$\sum_{(i=Inferiore..\ Superiore)} f(i)$$

Scrivere un programma che utilizzando questa funzione calcoli:

$$\sum_{(i=0..100)} i$$

$$\sum_{(i=0..100)} i^2$$

$$\sum_{(i=0..100)} i^3$$