

# Utilizzo delle librerie

---

Prof. Salvatore Venticinque

Prof. Emanuele Bellini

# Utilizzare le librerie

---

- Librerie standard del linguaggio:
  - Studiare i prototipi delle funzioni
  - Includere il file header
  - Passare correttamente i parametri
- Librerie di terze parti
  - Installare la libreria
  - Studiare li prototipi delle funzioni
  - Includere il file header
  - Passare correttamente i parametri

*In ogni caso: imparare a utilizzare la documentazione.*

# Librerie standard

---

- Nel sistema sono già presenti
  - I file oggetto della libreria
    - \*.o, \*.a (archivi di \*.o)
    - \*.so (linux), \*.dll (windows)
  - I file header contenenti i prototipi

# Librerie standard

---

File header	Area di riferimento
<code>&lt;assert.h&gt;</code>	Diagnostica.
<code>&lt;ctype.h&gt;</code>	Controllo e conversione caratteri. Per esempio: <code>isdigit(c)</code> ritorna un valore diverso da zero se <code>c</code> è una cifra decimale; analogamente operano <code>isalpha(c)</code> , <code>isspace(c)</code> , <code>isupper(c)</code> ecc.
<code>&lt;Errno.h&gt;</code>	Segnalazioni di errore.
<code>&lt;float.h&gt;</code>	Floating point.
<code>&lt;limits.h&gt;</code>	Definisce alcune costanti come <code>INT_MAX</code> e <code>INT_MIN</code> che contengono rispettivamente il massimo e il minimo valore rappresentabile con un <code>int</code> .
<code>&lt;locale.h&gt;</code>	Inizializzazione dei parametri locali.

# Librerie standard

<code>&lt;math.h&gt;</code>	Funzioni matematiche in doppia precisione.
<code>&lt;setjmp.h&gt;</code>	Salti non locali. Contiene la dichiarazione di funzioni che permettono di alterare l'esecuzione della normale sequenza di chiamata e uscita di una funzione, per esempio per obbligare a un ritorno immediato da una chiamata di funzione profondamente annidata.
<code>&lt;signal.h&gt;</code>	Gestione segnali. Contiene la dichiarazione di funzioni per la gestione di condizioni di eccezione che si verificano durante l'esecuzione, come l'arrivo di un segnale di interrupt da una sorgente esterna, oppure un errore nell'esecuzione.
<code>&lt;stdarg.h&gt;</code>	Gestione lista di argomenti variabili in numero e tipo. Contiene funzioni e/o macro che permettono di scandire tali liste, quindi può essere utile a sua volta per la realizzazione di funzioni che accettano un numero variabile di parametri.
<code>&lt;stddef.h&gt;</code>	Definizioni standard. Per esempio contiene la definizione di <code>ptrdiff_t</code> in grado di contenere la differenza, con segno, tra due puntatori e <code>size_t</code> il tipo (intero privo di segno) prodotto dalla funzione <code>sizeof</code> .
<code>&lt;stdio.h&gt;</code>	Input e Output. Funzioni quali <code>printf</code> e <code>scanf</code> .
<code>&lt;stdlib.h&gt;</code>	Utilità generale. Per esempio le funzioni per la conversione dei numeri, come <code>atof</code> , che trasforma una stringa in un <code>double</code> , o <code>rand</code> che ritorna un numero pseudo casuale.
<code>&lt;string.h&gt;</code>	Gestione di stringhe. Funzioni quali <code>strcpy</code> , che consente di copiare una stringa su un'altra e <code>strcat</code> che concatena due stringhe.
<code>&lt;time.h&gt;</code>	Gestione della data e dell'ora. Per esempio la funzione <code>time</code> che ritorna l'ora corrente.

# Librerie standard

- In linux: man stdlib.h

```
Activities Terminal
gio 19:58
salvatore@r7: /var/www/emoncms
File Edit View Search Terminal Help
ldiv_t ldiv (long __num, long __denom) __asm__('__divmodsi4')
void qsort (void *__base, size_t __nmemb, size_t __size,
__compar_fn_t __compar)
long strtol (const char *__nptr, char **__endptr, int __base)
unsigned long strtoul (const char *__nptr, char **__endptr, int
__base)
long atol (const char *__s) __ATTR_PURE__
int atoi (const char *__s) __ATTR_PURE__
void exit (int __status) __ATTR_NORETURN__
void * malloc (size_t __size) __ATTR_MALLOC__
void free (void *__ptr)
void * calloc (size_t __nele, size_t __size) __ATTR_MALLOC__
void * realloc (void *__ptr, size_t __size) __ATTR_MALLOC__
double strtod (const char *__nptr, char **__endptr)
double atof (const char *__nptr)
int rand (void)
void srand (unsigned int __seed)
int rand_r (unsigned long *__ctx)

Manual page stdlib.h(3avr) line 29 (press h for help or q to quit)
```

# stdlib.h

- Da Internet:  
<http://www.cplusplus.com/reference/cstdlib/>

The screenshot shows a web browser window with the address bar displaying `www.cplusplus.com/reference/cstdlib/`. The page title is `<cstdlib> (stdlib.h)`. The left sidebar contains a 'Reference' section with a list of headers: `<cassert>`, `<cctype>`, `<cerrno>`, `<cfenv>`, `<cmath>`, `<csetjmp>`, `<csignal>`, `<cstdlib>` (highlighted), `<cstring>`, `<ctgmath>`, `<ctime>`, `<uchar>`, `<wchar>`, and `<wctype>`. Below this are sections for 'Containers', 'Input/Output', 'Multi-threading', and 'Other'. The main content area is titled `<cstdlib> (stdlib.h)` and includes a description of the 'C Standard General Utilities Library'. It lists several function categories: 'String conversion' (including `atof`, `atoi`, `atol`, `atoll`, `strtod`, `strtof`, `strtol`, `strtold`, `strtoll`, `strtoul`, and `strtoull`), 'Pseudo-random sequence generation' (including `rand` and `srand`), and 'Dynamic memory management' (including `calloc`, `free`, `malloc`, and `realloc`).

header  
**<cstdlib> (stdlib.h)**

**C Standard General Utilities Library**

This header defines several general purpose functions, including dynamic memory management, random number generation, communication with the environment, integer arithmetics, searching, sorting and converting.

*fx* **Functions**

**String conversion**

<code>atof</code>	Convert string to double (function )
<code>atoi</code>	Convert string to integer (function )
<code>atol</code>	Convert string to long integer (function )
<code>atoll</code> <small>C++11</small>	Convert string to long long integer (function )
<code>strtod</code>	Convert string to double (function )
<code>strtof</code> <small>C++11</small>	Convert string to float (function )
<code>strtol</code>	Convert string to long integer (function )
<code>strtold</code> <small>C++11</small>	Convert string to long double (function )
<code>strtoll</code> <small>C++11</small>	Convert string to long long integer (function )
<code>strtoul</code>	Convert string to unsigned long integer (function )
<code>strtoull</code> <small>C++11</small>	Convert string to unsigned long long integer (function )

**Pseudo-random sequence generation**

<code>rand</code>	Generate random number (function )
<code>srand</code>	Initialize random number generator (function )

**Dynamic memory management**

<code>calloc</code>	Allocate and zero-initialize array (function )
<code>free</code>	Deallocate memory block (function )
<code>malloc</code>	Allocate memory block (function )
<code>realloc</code>	Reallocate memory block (function )

# rand

```
int rand (void);
```

## Generate random number

Returns a pseudo-random integral number in the range between 0 and `RAND_MAX`.

This number is generated by an algorithm that returns a sequence of apparently non-related numbers each time it is called. This algorithm uses a seed to generate the series, which should be initialized to some distinctive value using function `srand`.

`RAND_MAX` is a constant defined in `<cstdlib>`.

A typical way to generate trivial pseudo-random numbers in a determined range using `rand` is to use the modulo of the returned value by the range span and add the initial value of the range:

```
1 v1 = rand() % 100;           // v1 in the range 0 to 99
2 v2 = rand() % 100 + 1;       // v2 in the range 1 to 100
3 v3 = rand() % 30 + 1985;      // v3 in the range 1985-2014
```

Notice though that this modulo operation does not generate uniformly distributed random numbers in the span (since in most cases this operation makes lower numbers slightly more likely).

C++ supports a wide range of powerful tools to generate random and pseudo-random numbers (see `<random>` for more info).



## Parameters

(none)



## Return Value

An integer value between 0 and `RAND_MAX`.



# Utilizzo di int rand()

## Example

```
1 /* rand example: guess the number */
2 #include <stdio.h>      /* printf, scanf, puts, NULL */
3 #include <stdlib.h>     /* srand, rand */
4 #include <time.h>       /* time */
5
6 int main ()
7 {
8     int iSecret, iGuess;
9
10    /* initialize random seed: */
11    srand (time(NULL));
12
13    /* generate secret number between 1 and 10: */
14    iSecret = rand() % 10 + 1;
15
16    do {
17        printf ("Guess the number (1 to 10): ");
18        scanf ("%d",&iGuess);
19        if (iSecret<iGuess) puts ("The secret number is lower");
20        else if (iSecret>iGuess) puts ("The secret number is higher");
21    } while (iSecret!=iGuess);
22
23    puts ("Congratulations!");
24    return 0;
25 }
```

In this example, the random seed is initialized to a value representing the current time (calling `time`) to generate a different value every time the program is run.

Possible output:

```
Guess the number (1 to 10): 5
The secret number is higher
Guess the number (1 to 10): 8
The secret number is lower
Guess the number (1 to 10): 7
Congratulations!
```

# Utilizzo di int rand()

---

`int rand()`

- È una funzione che appartiene alla libreria `stdlib.h`
- Ritorna un intero da una sequenza random compreso tra 0 e `RAND_MAX`
- Cosa succede se viene eseguito più volte il seguente programma?

```
int a =random();  
printf(“%d\n”,a);  
printf(“%d\n”,random());
```

# Utilizzo di int rand()

---

// v1 in the range 0 to 99

```
v1 = rand() % 100;
```

// v2 in the range 1 to 100

```
v2 = rand() % 100 + 1;
```

// v3 in the range 1985-2014

```
v3 = rand() % 30 + 1985;
```

# Utilizzo di srand(time\_t)

---

- void srand (unsigned int seed);
- 
- la sequenza di numeri random.

Es:

```
srand(10);
```

```
int a =random();
```

```
printf(“%d\n”,a);
```

```
printf(“%d\n”,random());
```

*Stamperà una nuova coppia di numeri random!!!*

# Utilizzo di srand()

---

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

```
#include <time.h>
```

```
int main()
```

```
{
```

```
    srand(time(NULL));
```

```
    int a = random();
```

```
    printf("%d\n",a);
```

```
    printf(random());
```

```
}
```

# math.h

---

- `double cos (double x);`
- `double sin (double x);`
- `double tan (double x);`
- `[...]`
- `double cosh(double x);`
- `[...]`
- `double log2 (double x);`
- `float log2f (float x);`
- `double exp (double x);`
-

# log

---

double log(double d)

- Ritorna il logaritmo naturale di d
- Se d è negativo viene settata una variabile globale **errno**
- in caso di errore **errno** è uguale ad un valore intero dichiarato come costante **EDOM** in **errno.h**

# Esempio

---

```
#include <stdio.h>
#include <math.h>
#include <errno.h>
int main()
{
    double d=log(123.4);
    if(errno==EDOM)
        printf("errore\n");
    else
        printf("%f\n",d);
    d=log(-123.4);
    if(errno==EDOM)
        printf("errore\n");
    else
        printf("%f\n",d);
}
```



# Alcuni esercizi

---

- Realizzare un programma che consente all'utente di giocare contro il computer a carta, forbice, sasso.
- Realizzare un programma che stampa la figura del coseno in verticale a console.
- Realizzare il calcolo del pigreco come rapporto tra i punti random che cadono all'interno di un cerchio di raggio  $r$  e all'interno del quadrato di lato  $2r$  in cui il cerchio è iscritto.