C++
Information
Tutorials
Reference
Articles
Forum

Reference
C library:
<cassert> (assert.h)
<cctype> (ctype.h)
<cerrno> (errno.h)
<cfenv> (fenv.h)
<cfloat> (float.h)
<cinttypes> (inttypes.h)
<ciso646> (iso646.h)
<climits> (limits.h)
<clocale> (locale.h)
<cmath> (math.h)
<csetjmp> (setjmp.h)
<csignal> (signal.h)
<cstdarg> (stdarg.h)
<cstdbool> (stdbool.h)
<cstddef> (stddef.h)
<cstdint> (stdint.h)
<cstdio> (stdio.h)
<cstdlib> (stdlib.h)
<cstring> (string.h)
<ctgmath> (tgmath.h)
<ctime> (time.h)
<cuchar> (uchar.h)
<cwchar> (wchar.h)
<cwctype> (wctype.h)
Containers:
Input/Output:
Multi-threading:
Other:

<cstdlib> (stdlib.h)
functions:
abort
abs
atexit
atof
atoi
atol
atoll
at_quick_exit
bsearch
calloc
div
exit
free
getenv
labs
ldiv
llabs
lldiv
malloc
mblen
mbstowcs
mbtowc
qsort
quick_exit
rand
realloc
srand
strtod
strtof
strtol
strtold
strtoll
strtoul
strtoull
system
wcstombs
wctomb
_Exit
functions (non-standard):
itoa
types:
div_t
ldiv_t
lldiv_t
size_t

function

rand

<cstdlib>

```
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:

```
v1 = rand() % 100;          // v1 in the range 0 to 99
v2 = rand() % 100 + 1;      // v2 in the range 1 to 100
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.

Example

```
/* rand example: guess the number */
#include <stdio.h>      /* printf, scanf, puts, NULL */
#include <stdlib.h>     /* srand, rand */
#include <time.h>       /* time */

int main ()
{
  int iSecret, iGuess;

  /* initialize random seed: */
  srand (time(NULL));

  /* generate secret number between 1 and 10: */
  iSecret = rand() % 10 + 1;

  do {
    printf ("Guess the number (1 to 10): ");
    scanf ("%d",&iGuess);
    if (iSecret<iGuess) puts ("The secret number is lower");
    else if (iSecret>iGuess) puts ("The secret number is higher");
  } while (iSecret!=iGuess);

  puts ("Congratulations!");
  return 0;
}
```

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!
```

Compatibility

In C, the generation algorithm used by rand is guaranteed to only be advanced by calls to this function. In C++, this constraint is relaxed, and a library implementation is allowed to advance the generator on other circumstances (such as calls to elements of <random>).

Data races

The function accesses and modifies internal state objects, which may cause data races with concurrent calls to rand or srand.

Some libraries provide an alternative function that explicitly avoids this kind of data race: rand_r (non-portable).

macro constants:

EXIT_FAILURE
EXIT_SUCCESS
MB_CUR_MAX
NULL
RAND_MAX

C++ library implementations are allowed to guarantee no *data races* for calling this function.

Exceptions (C++)

No-throw guarantee: this function never throws exceptions.

See also

srand	Initialize random number generator (function)