# Tarjeta de referencia de C (ANSI)

## Liberías estándar ANSI

| | | | | |
|---|---|---|---|---|
| <assert.h> | <ctype.h> | <errno.h> | <float.h> | <limits.h> |
| <locale.h> | <math.h> | <setjmp.h> | <signal.h> | <stdarg.h> |
| <stddef.h> | <stdio.h> | <stdlib.h> | <string.h> | <time.h> |

## Pruebas de carácter <ctype.h>

| | |
|---|---|
| ¿Alfanumérico? | `isalnum(c)` |
| ¿Alfabético? | `isalpha(c)` |
| ¿Carácter de control? | `iscntrl(c)` |
| ¿Dígito decimal? | `isdigit(c)` |
| ¿Carácter imprimible (excepto espacio)? | `isgraph(c)` |
| ¿Minúscula? | `islower(c)` |
| ¿Carácter imprimible (incluye espacio)? | `isprint(c)` |
| ¿Carácter imprimible excepto espacio, letra, dígito? | `ispunct(c)` |
| space, formfeed, newline, cr, tab, vtab? | `isspace(c)` |
| ¿Mayúscula? | `isupper(c)` |
| ¿Dígito hexadecimal? | `isxdigit(c)` |
| Convertir a minúscula | `tolower(c)` |
| Convertir a mayúscula | `toupper(c)` |

## Operaciones de cadenas <string.h>

`s` es una cadena; `cs`, `ct` son cadenas constantes

| | |
|---|---|
| longitud de `s` | `strlen(s)` |
| copia `ct` a `s` | `strcpy(s,ct)` |
| concatena `ct` al final de `s` | `strcat(s,ct)` |
| compara `cs` con `ct` | `strcmp(cs,ct)` |
| solo los primeros `n` chars | `strncmp(cs,ct,n)` |
| puntero al primer `c` en `cs` | `strchr(cs,c)` |
| puntero al último `c` en `cs` | `strrchr(cs,c)` |
| copia `n` chars de `ct` a `s` | `memcpy(s,ct,n)` |
| copia `n` chars de `ct` a `s` (pueden solaparse) | `memmove(s,ct,n)` |
| compara `n` chars de `cs` con `ct` | `memcmp(cs,ct,n)` |
| puntero al primer `c` en los primeros `n` chars de `cs` | `memchr(cs,c,n)` |
| pone `c` en los primeros `n` chars de `s` | `memset(s,c,n)` |

## Entrada/Salida <stdio.h>

**E/S estándar**

| | |
|---|---|
| flujo de entrada estándar | `stdin` |
| flujo de salida estándar | `stdout` |
| flujo de error estándar | `stderr` |
| fin de archivo (el tipo es `int`) | `EOF` |
| leer un carácter | `getchar()` |
| imprimir un carácter | `putchar(`*chr*`)` |
| imprimir datos con formato | `printf("`*format*`",`$arg_1$`,...)` |
| imprimir a cadena `s` | `sprintf(s,"`*format*`",`$arg_1$`,...)` |
| leer datos con formato | `scanf("`*format*`",&`$name_1$`,...)` |
| leer de un cadena `s` | `sscanf(s,"`*format*`",&`$name_1$`,...)` |
| imprimir cadena `s` | `puts(s)` |

**E/S de archivos**

| | |
|---|---|
| declarar un puntero a archivo | `FILE *`*fp*`;` |
| abrir archivo | `fopen("`*name*`","`*mode*`")` |
| modos: `r` (leer), `w` (escribir), `a` (agregar), `b` (binario) | |
| leer un carácter | `getc(`*fp*`)` |
| escribir un carácter | `putc(`*chr*`,`*fp*`)` |
| escribir a archivo | `fprintf(`*fp*`,"`*format*`",`$arg_1$`,...)` |
| leer de archivo | `fscanf(`*fp*`,"`*format*`",`$arg_1$`,...)` |
| leer y almacenar `n` els a `*ptr` | `fread(*ptr,elssize,n,`*fp*`)` |
| escribir `n` els de `*ptr` a archivo | `fwrite(*ptr,elssize,n,`*fp*`)` |
| cerrar archivo | `fclose(`*fp*`)` |
| distinto de cero si error | `ferror(`*fp*`)` |
| distinto de cero si EOF | `feof(`*fp*`)` |
| leer línea a cadena `s` (< `max` chars) | `fgets(s,max,`*fp*`)` |
| escribir cadena `s` | `fputs(s,`*fp*`)` |

**Códigos para E/S con formato**: `"%-+ 0`*w*`.`*pmc*`"`

| | |
|---|---|
| `-` | alinear a izquierda |
| `+` | imprimir con signo |
| *space* | imprimir espacio si no hay signo |
| `0` | rellenar al principio con ceros |
| *w* | anchura mínima del campo |
| *p* | precisión |
| *m* | carácter de conversión: `h` short, `l` long, `L` long double |
| *c* | carácter de conversión: |

| | | | |
|---|---|---|---|
| `d,i` | entero | `u` | unsigned |
| `c` | carácter | `s` | cadena |
| `f` | double (`printf`) | `e,E` | exponencial |
| `f` | float (`scanf`) | `lf` | double (`scanf`) |
| `o` | octal | `x,X` | hexadecimal |
| `p` | puntero | `n` | número de carácteres escritos |
| `g,G` | lo mismo que `f` o `e,E` dependiendo del exponente | | |

## Listas de args variables <stdarg.h>

| | |
|---|---|
| declaración de puntero a argumentos | `va_list` *ap*`;` |
| inicialización de puntero a argumento | `va_start(`*ap*`,`*lastarg*`);` |
| *lastarg* es el último parámetro con nombre de la función | |
| accede al próximo arg sin nombre, actualiza el puntero | `va_arg(`*ap*`,`*type*`)` |
| invocar antes de salir de la función | `va_end(`*ap*`);` |

## Funciones estándar útiles <stdlib.h>

| | |
|---|---|
| valor absoluto de `int n` | `abs(n)` |
| valor absoluto de `long n` | `labs(n)` |
| cociente y resto de `int`s `n`,`d` | `div(n,d)` |
| restorna estructura con `div_t.quot` y `div_t.rem` | |
| cociente y resto de `long`s `n`,`d` | `ldiv(n,d)` |
| retorna una estructura con `ldiv_t.quot` y `ldiv_t.rem` | |
| entero pseudo-aleatorio `[0,RAND_MAX]` | `rand()` |
| establecer semilla aleatoria a `n` | `srand(n)` |
| terminar la ejecución del programa | `exit(status)` |
| pasar la cadena `s` al sistema para ejecutarla | `system(s)` |

**Conversiones**

| | |
|---|---|
| convertir cadena `s` a double | `atof(s)` |
| convertir cadena `s` a int | `atoi(s)` |
| convertir cadena`s` a long | `atol(s)` |
| convertir prefijo de `s` a `double` | `strtod(s,&endp)` |
| convertir prefijo de `s` (base `b`) a `long` | `strtol(s,&endp,b)` |
| lo mismo, pero `unsigned long` | `strtoul(s,&endp,b)` |

**Alocación de memoria**

| | |
|---|---|
| alocar memoria | `malloc(size), calloc(nobj,size)` |
| cambiar tamaño de lo alocado | `newptr = realloc(ptr,size);` |
| liberar memoria | `free(ptr);` |

**Funciones de array**

| | |
|---|---|
| buscar `key` en `array` | `bsearch(key,array,n,size,cmpf)` |
| ordenar `array` en orden ascendente | `qsort(array,n,size,cmpf)` |

## Funciones de fecha y hora <time.h>

| | |
|---|---|
| tiempo de procesador usado por el programa | `clock()` |
| *Example*. `clock()/CLOCKS_PER_SEC` es tiempo en segundos | |
| hora actual | `time()` |
| $time_2$`-`$time_1$ en segundos (`double`) | `difftime(`$time_2$`,`$time_1$`)` |
| tipos numéricos para representar tiempos | `clock_t,time_t` |
| estructura para representar fecha y hora | `struct tm` |

| | |
|---|---|
| `tm_sec` | segundos en el minuto |
| `tm_min` | minutos en la hora |
| `tm_hour` | horas desde la medianoche |
| `tm_mday` | día del mes |
| `tm_mon` | meses desde Enero |
| `tm_year` | años desde 1900 |
| `tm_wday` | días desde el domingo |
| `tm_yday` | días desde 1° de Enero |
| `tm_isdst` | indicador de horario de verano |

| | |
|---|---|
| convertir hora local a hora de calendario | `mktime(tp)` |
| convertir hora `tp` a cadena | `asctime(tp)` |
| convertir hora de calendario en `tp` a hora local | `ctime(tp)` |
| convertir hora de calendario a GMT | `gmtime(tp)` |
| convertir hora de calendario a hora local | `localtime(tp)` |
| formatear fecha y hora | `strftime(s,smax,"`*format*`",tp)` |
| `tp` es un puntero a una estructura de tipo `tm` | |

## Funciones matemáticas `<math.h>`

Los argumentos y valores de retorno son `double`

| | |
|---|---|
| trigonométricas | `sin(x), cos(x), tan(x)` |
| trig. inversas | `asin(x), acos(x), atan(x)` |
| arctan($y/x$) | `atan2(y,x)` |
| trig. hiperbólicas | `sinh(x), cosh(x), tanh(x)` |
| exponenciales y logs | `exp(x), log(x), log10(x)` |
| exponentiales y logs (2 power) | `ldexp(x,n), frexp(x,&e)` |
| división y resto | `modf(x,ip), fmod(x,y)` |
| potencias | `pow(x,y), sqrt(x)` |
| redondeo | `ceil(x), floor(x), fabs(x)` |

## Límites de tipos enteros `<limits.h>`

Los números dados entre paréntesis son los valores típicos para las constantes en un sistema Unix de 32 bits, seguidos por un valor mínimo requerido normalmente (si fueran significativamente diferentes).

Nota del traductor: Estos valores son ilustrativos.

| | | |
|---|---|---|
| `CHAR_BIT` | bits en `char` | (8) |
| `CHAR_MAX` | valor máximo de `char` | (`SCHAR_MAX` or `UCHAR_MAX`) |
| `CHAR_MIN` | valor mínimo de `char` | (`SCHAR_MIN` or 0) |
| `SCHAR_MAX` | max `signed char` | (+127) |
| `SCHAR_MIN` | min `signed char` | (−128) |
| `SHRT_MAX` | valor máximo de `short` | (+32,767) |
| `SHRT_MIN` | valor mínimo de `short` | (−32,768) |
| `INT_MAX` | valor máximo de `int` | (+2,147,483,647) (+32,767) |
| `INT_MIN` | valor mínimo de `int` | (−2,147,483,648) (−32,767) |
| `LONG_MAX` | valor máximo de `long` | (+2,147,483,647) |
| `LONG_MIN` | valor mínimo de `long` | (−2,147,483,648) |
| `UCHAR_MAX` | max `unsigned char` | (255) |
| `USHRT_MAX` | max `unsigned short` | (65,535) |
| `UINT_MAX` | max `unsigned int` | (4,294,967,295) (65,535) |
| `ULONG_MAX` | max `unsigned long` | (4,294,967,295) |

## Límites de tipos float `<float.h>`

Los números dados entre paréntesis son los valores típicos para las constantes en un sistema Unix de 32 bits, .

| | | |
|---|---|---|
| `FLT_RADIX` | dígitos del exponente | (2) |
| `FLT_ROUNDS` | modo de redondeo | |
| `FLT_DIG` | dígitos decimales de presición | (6) |
| `FLT_EPSILON` | mínimo $x$ tal que $1.0\text{f} + x \neq 1.0\text{f}$ | ($1.1\text{E} - 7$) |
| `FLT_MANT_DIG` | número de dígitos en la mantisa | |
| `FLT_MAX` | max número `float` | ($3.4\text{E}38$) |
| `FLT_MAX_EXP` | max. exponente | |
| `FLT_MIN` | min múmero `float` | ($1.2\text{E} - 38$) |
| `FLT_MIN_EXP` | min. exponente | |
| `DBL_DIG` | dígitos decimales de presición | (15) |
| `DBL_EPSILON` | mínimo $x$ tal que $1.0 + x \neq 1.0$ | ($2.2\text{E} - 16$) |
| `DBL_MANT_DIG` | número de dígitos en la mantisa | |
| `DBL_MAX` | max. número `double` | ($1.8\text{E}308$) |
| `DBL_MAX_EXP` | max. exponente | |
| `DBL_MIN` | min. número `double` | ($2.2\text{E} - 308$) |
| `DBL_MIN_EXP` | min. exponente | |



Send comments and corrections to J.H. Silverman, Math. Dept., Brown Univ., Providence, RI 02912 USA. ⟨jhs@math.brown.edu⟩