Tarjeta de referencia rápida de C (ANSI)

Programa

Estructura/Funciones

type fnc(tipo1 ,...)	declaraciones de funciones
type nombre	declaraciones externas de
variables	
main() {	rutina principal
declaraciones	declaraciones de variables locales
}	
type fnc(arg 1,...) {	definición de función
declaraciones	declaraciones de variables
locales	
return *valor;*	
}	
/* */	comentarios
main(int argc, char *argv[])	main con argumentos
exit(*arg*)	terminación de ejecución

Preprocesador de C

inclución de archivo librería	**#include** <*nombre_archivo*>
inclución de achivo del usuario	**#include** "*normbre_archivo*"
reemplazando texto	**#define** *nombre texto*
macro de reemplazo	**#define** *nombre(var) texto*
Ejemplo: **#define max(A,B) ((A)>(B) ? (A) : (B))**	
indefinir	**#undef** *nombre*
en reemplazos, citar cadenas	**#**
concatenar argumentos y rebuscar	**##**
ejecución condicional	**#if, #else, #elif, #endif**
si nombre denegado ¿no definido?	**#ifdef, #ifndef**
¿nombre definido?	**defined**(*nombre*)
caracter de continuación de línea	****

Tipos de datos/Declaraciones

caracter (1 byte)	**char**
entero	**int**
flotante (presición simple)	**float**
flotante (doble presición)	**double**
corto (16 bit entero)	**short**
largo (32 bit entero)	**long**
positivo y negativo	**signed**
solamente positivo	**unsigned**
puntero a entero, flotante, ...	***int, *float,...**
constante de enumeración	**enum**
constante (descambiar) valor	**const**
declaración externa de variable	**extern**
variable de registro	**register**
local a archivo fuente	**static**
sin valor	**void**
estructura	**struct**
crear nombre por tipo de dato	**typedef** *nombre_tipo*
tamaño de objeto (tipo es size_t)	**sizeof** *objecto*
tamaño de tipo de dato (tipo es size_t)	**sizeof**(*type nombre*)

Inicialización

inicializa variable	*type nombre=valor*
inicialice matriz	*type nombre[]={valor 1 ,...}*
inicialice cadena de caracteres	**char** *nombre[]*="*cadena*"

Constantes

long (sufijo)	L ó l
float (sufijo)	F ó f
forma exponencial	e
octal (prefijo cero)	0
hexadecimal (prefijo cero-equis)	0x ó 0X
caracter constante (char, octal, hex)	'a', '\ooo', '\xhh'
nueva línea, cr, tab, retroceso	\n, \r, \t, \b
caracteres especiales	\\, \?, \', \"
cadena constante (fin con '\0')	"abc...de"

Punteros, matrices y estructuras

declarar puntero a tipo	type *name
declarar función retornado por puntero a tipo	type *f()
declarar puntero a función retornando tipo	type (*pf)()
tipo de puntero genérico	void *
puntero nulo	NULL
objeto apuntado a por puntero	*pointer
dirección de objeto nombre	&name
matriz	name[dim]
matriz multi dimensión	name[dim 1][dim 2]...

Estructuras

```
        struct etiqueta {
        estructrua de plantilla
          declaraciones                    declaración
de miembros
        };
```

crear estructura	**struct** *etiqueta*
nombre	
miembro de estructura desde plantilla	*nombre.miembro*
nombre de la estructura apuntada	*puntero -> miembro*

Ejemplo: (*p).x y p->x son lo mismo

valor simple, múltimple tipo de estructura	**union**
bit de campo con b bits	*miembro : b*

Operadores (agrupado por procedencia)

operador de miembro de estructura	*nombre.miembro*
puntero de estructura	*puntero -> miembro*
incremento, decremento	++, --
suma, resta, negación lógica, negación bitwise	+, -, !, ~
indirección vía puntero, dirección de objeto	**puntero , &nombre*
reparto de expresión a tipo	*(tipo) expresíon*
tamaño de objeto	sizeof
producto, división, módulo (resto)	*, /, %
adhición, sustracción	+, -
movimiento izquierdo, derecho [bit ops]	<<, >>
comparaciones	>, >=, <, <=
comparaciones	==, !=
Y bitwise	&
O exclusivo	^
O bitwise (incl)	|
Y lógico	&&
O lógico	||
expresión condicional	expr 1 ? expr 2 : expr 3
operadores de asignación	+=, -=, *=, ...
separador de expresiones de evaluación	,

Operadores unarios, expresiones condicionales y asignación de operadores de grupo de derecha a izquierda; para todos los demás grupos de izquierda a derecha.

Control de flujos

terminación de declaración	;
delimitadores de bloque	{}
salir del switch, while, do, for	break
próxima iteración de while, do, for	continue
ir a	goto *marca*
marca	*marca:*
valor retornado desde función	*return expr*

Construcción de flujos

if declaración	**if** *(expr) declaración* **else if** *(expr)* **else** *declaración*
declaración while declaración	**while** *(expr)* *declaración*
for declaración	**for** *(expr 1 ; expr 2 ; expr 3)* *declaración*
do declaración	**do** *declaración* **while** *(expr);*
switch declaración	**switch** *(expr){* **case** *const 1 : declaración1* **break**; **case** *const 2 : declaración2* **break**; **default**: *declaración* }

Librería 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>

Prueba de clase de caracter <ctype.h>

¿alfanumérico?	isalnum(c)
¿alfabético?	isalpha(c)
¿caracter de control?	iscntrl(c)
¿dígito decimal?	isdigit(c)
¿imprimiendo caracter (no incluye espacio)?	isgraph(c)
¿letra minúscula?	islower(c)
¿imprecsión de caracter (incluye espacio)?	isprint(c)
¿imprimiendo caracter excepto espacio, letra, dígito?	ispunct(c)
espacio, formfeed, nueva línea, cr, tab, vtab?	isspace(c)
¿letra mayúscula?	isupper(c)
¿dígito hexadecimal?	isxdigit(c)
¿convertir a minúscula?	tolower(c)
¿convertir a mayúscula?	toupper(c)

Operaciones con cadenas <string.h>

s,t son cadenas, cs,ct son constantes de cadena

lingitud de s	strlen(s)
copiar ct a s	strcpy(s,ct)
arriba a n caracteres	strncpy(s,ct,n)
concatenar ct antes de s	strcat(s,ct)
arriba a n caracteres	strncat(s,ct,n)
comparar cs a ct	strcmp(cs,ct)
solamente primer n caracteres	strncmp(cs,ct,n)
puntero a primer c en cs	strchr(cs,c)
puntero a último c en cs	strrchr(cs,c)
copiar n caracteres desde ct a s	memcpy(s,ct,n)
copiar n caracteres desde ct a s (may overlap)	memmove(s,ct,n)
compara n caracteres de cs con ct	memcmp(cs,ct,n)
puntero a primer c dentro primer n caracteres de cs	memchr(cs,c,n)
poner c dentro primer n caracteres de cs	memset(s,c,n)

Entrada/Salida <stdio.h>

Estándar Ent/Sal

flujo de entrada estándar	**stdin**
flujo de salida estándar	**stdout**
flujo de error estándar	**stderr**
fin de archivo	**EOF**
tomar caracter	**getchar** ()
imprimir caracter	**putchar** (*chr*)
imprimir datos formateados	**printf** ("*formato* ",*arg 1,...*)
imprimir a cadena s	**sprintf** (*s,*"*formato* ",*arg 1,...*)
leer datos formateados	**scanf** ("*formato* ",*&nombre 1,..*)
leer desde la cadena s	**sscanf** (*s,*"*format* ",*&nombre 1,...*)
leer línea a cadena s (< max caracteres)	**gets** (*s,max*)
imprimir cadena s	**puts** (*s*)

Archivo Entrada/Salida

puntero de declaración de archivo	**FILE** **fp*
puntero a nombre de archivo	**fopen** ("*nombre* ","*modo* ")
modos: r (lectura), w (escritura), a (agregar)	
tomar caracter	**getc** (*fp*)
escribir caracter	**putc** (*chr ,fp*)
escribir a archivo	**fprintf** (*fp,"formato ", arg 1, ...*)
leer de archivo	**fscanf** (*fp,"formato ",arg 1,...*)
cerrar archivo	**fclose** (*fp*)
no-cero si error	**ferror** (*fp*)
no-cero si EOF	**feof** (*fp*)
leer línea a cadena s (< max caracteres)	**fgets** (*s,max,fp*)
escribir cadena s	**fputs** (*s,fp*)

Códigos para formatear la entrada/salida: "%-+ 0w:pmc"

-	justificación izquierda
+	impresión con signo
espacio	impresión de espacio si no hay signo
0	bloc con precisión de ceros
w	ancho mínimo de campo
p	precisión
m	caracter de converción: **h** corto, **l** largo, **L** largo doble
c	caracter de converción:

d,i	entero	u	sin signo
c	caracter simple	s	cadena de caracter
f	doble	e,E	exponencial
o	octal	x,X	hexadecimal
p	puntero	n	número de caracteres escritos
g,G	algunos son f ó e,E dependiendo del exponente		

Listas de argumentos variables <stdarg.h>

declaración de puntero a argumento	**va_list** *nombre*;
inicialización de puntero argumento	**va_start** (*nombre ,último_arg*)
último_arg es el último parámetro llamado de la función	
acceso al próximo argumento, actualizar puntero	**va_arg** (*nombre, tipo*)
llamado anterior de salir de la función	**va_end** (*nombre*)

Funciones útiles estándar <stdlib.h>

valor absoluto de entero n	abs(n)
valor absoluto de largo n	labs(n)
división y resto de enteros n,d	div(n,d)
estructura de retornos con div_t.quot y div_t.rem	
división y resto de largos n,d	ldiv(n,d)
estructura de retornos con ldiv_t.quot y ldiv_t.rem	
entero seudo aleatorio [0,RAND_MAX]	rand()
fijar semilla aleatoria a n	srand(n)
terminación de la ejecución del programa	exit(status)
pasar cadena s al sistema para ejecución	system(s)

Converciones

convertir cadena s a doble	atof(s)
convertir cadena s a entero	atoi(s)
convertir cadena s a largo (long)	atol(s)
convertir prefijo de s a doble	strtod(s,endp)
convertir prefijo de s (base b) a largo (long)	strtol(s,endp,b)
algunos, pero sin signo largo (unsing long)	strtoul(s,endp,b)

Asignación de almacenamiento

asignación de almacenamiento	malloc(tamaño), calloc(nobj,tamaño)
cambiar tamaño de objeto	realloc(pts,tamaño)
desasignar espacio	free(ptr)

Funciones de matriz

clave para buscar en matriz	bsearch(clave,matriz,n,tamaño,cmp())
ordenar ascendentemente la matriz	qsort(matriz,n,tamaño,cmp())

Funciones de fecha y tiempo <time.h>

tiempo de proceso usado por el programa	**clock**()
Ejemplo: **clock**()/**CLOCKS_PER_SEC** tiempo en segundos	
tiempo de calendario corriente	**time**()
tiempo2 - tiempo1 en segundos (**doble**)	**difftime**(tiempo2 ,tiempo1)
tiempos de representación de tipos aritméticos	**clock_t,time_t**
tipo de estructura para comparación de tiempo en el calendario	**tm**

tm_sec	s	egundos antes del minuto
tm_min		minutos antes de la hora
tm_hour		horas desde medianoche
tm_mday		días del mes
tm_mon		meses desde Enero

tm_year	años desde 1900
tm_wday	días desde el Domingo
tm_yday	días desde Enero 1
tm_isdst	bandera de Salvado de Tiempo de Luz Solar

convertir tiempo local a tiempo calendario	**mktime**(tp)
convertio tiempo en **tp** a cadena	**asctime**(tp)
convertir tiempo calendario en tp a tiempo local	**ctime**(tp)
convertir tiempo calendario a GMT	**gmtime**(tp)
convertir tiempo calendario a tiempo local	**localtime**(tp)
formato de fecha e información de tiempo	**strftime**(s,smax,"formato ",tp)
tp es un puntero a estructura de tipo	**tm**

Funciones matemáticas <math.h>

Los argumentos y los valores retornados son de longitud doble (double)

funciones trigonométricas	sin(x), cos(x), tan(x)
funciones trigonométricas inversas	asin(x), acos(x), atan(x)
arcotangente(y/x)	atan2(y,x)
funciones trigonométricas hiperbólicas	sinh(x), cosh(x), tanh(x)
exponentes y logaritmos	exp(x), log(x), log10(x)
exponentes y logaritmos (2 potencias)	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 tipo entero <limits.h>

Los números dados entre paréntesis son valores típicos para constantes de sistemas Unix de 32 bits.

CHAR_BIT	bits en un caracter	(8)
CHAR_MAX	valor máximo de un caracter	(127 or 255)
CHAR_MIN	valor mínimo de un caracter	(..128 or 0)
INT_MAX	valor máximo de un entero	(+32,767)
INT_MIN	valor mínimo de un entero	(-32,768)
LONG_MAX	max valor de largo (long)	(+2,147,483,647)
LONG_MIN	min valor de largo (long)	(-2,147,483,648)
SCHAR_MAX	max valor de caracter con signo	(+127)
SCHAR_MIN	min valor de caracter con signo	(-128)
SHRT_MAX	max valor de corto (short)	(+32,767)
SHRT_MIN	min valor de corto (short)	(-32,768)
UCHAR_MAX	max valor de caracter sin signo	(255)
UINT_MAX	max valor de entero sin signo	(65,535)
ULONG_MAX	max valor de largo sin signo	(4,294,967,295)
USHRT_MAX	max valor de carto sin signo	(65,536)

Límites de tipo flotante <float.h>

FLT_RADIX	radix of exponent rep	(2)
FLT_ROUNDS	modo de redondeo de punto flotante	
FLT_DIG	dígitos decimales de precisión	(6)
FLT_EPSILON	mínimo x entonces 1.0 + x <> 1:0	(10^{-5})
FLT_MANT_DIG	número de dígitos en la mantisa	
FLT_MAX	número máximo de punto flotante	(10^{37})
FLT_MAX_EXP	exponente máximo	
FLT_MIN	número mínimo de punto flotante	(10^{-37})
FLT_MIN_EXP	exponente mínimo	
DBL_DIG	dígitos decimales de precisión	(10)
DBL_EPSILON	mínimo x entonces 1.0 + x <> 1.0	(10^{-9})
DBL_MANT_DIG	número de dígitos en la mantisa	
DBL_MAX	número máximo de punto flotante doble	(10^{37})
DBL_MAX_EXP	exponente máximo	
DBL_MIN	número mínimo de punto flotante doble	(10^{-37})
DBL_MIN_EXP	exponente mínimo	

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

Traducción
Febrero 2007 v1.0 por ariel50.-
Comentarios de la traducción : edgarross50@yahoo.com.ar