Tarjeta de referencia ANSI C (simplificada)

Estructura de programa/funciones

tipo func(*tipo*$_1$,. . .);	declaración de prototipo de funciones
tipo nombre;	declaración de variables globales
int main (**void**)	función principal sin argumentos
{	
declaraciones	definición de variables locales
instrucciones	
}	
tipo func(*arg*$_1$,. . .)	definición de función
{	
declaraciones	declaración de variables locales
instrucciones	
return *valor*$_1$;	
}	
/* */	comentarios

Preprocesador de C

incluir fichero de cabeceras	#include <*fichero*>
sustitución de texto	#define *nombre texto*

Tipos de datos. Declaraciones

carácter (1 byte)	**char**
entero	**int**
real (precisión simple)	**float**
real (precisión doble)	**double**
corto (entero de 16 bits)	**short**
largo (entero de 32 bits)	**long**
con signo (positivo y negativo)	**signed**
sólo positivo	**unsigned**
puntero a **int**, **float**,. . .	***int**, ***float**,. . .
variable estática	**static**
sin tipo	**void**

Inicialización

Inicializar variable	*tipo nombre=valor*
Inicializar vector	*tipo v*[]={*valor*$_1$, ...}
Inicializar cadena	***char** str*[]="*cadena*";

Constantes

hexadecimal (prefijo cero-equis)	0x ó 0X
carácter cte. (char, octal, hex.)	'a', '\ooo', '\xhh'
nueva línea, ret. carro, tab.	\n, \r, \t
caracteres especiales	\\, \?, \', \"
cadena constante (termina en '\0')	"abc. . . de"

Punteros y vectores

declarar un puntero a tipo	*tipo *nombre*
objeto apuntado por puntero	**puntero*
dirección del objeto nombre	*&nombre*
vector	*nombre*[*dim*]

Tablas de verdad

a	b	a & b a && b	a | b a || b	a ^ b	~a ! a
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

Operadores (según precedencia)

acceso a elemento de vector	*v*[*índ*]
incremento, decremento	++, --
más, menos, no lógico, negación bit a bit	+, -, !, ~
acceso por puntero, direcc. de objeto	**ptr*, *&var*
convertir tipo de expresión	*(tipo)expr*
tamaño de un objeto	*sizeof*
producto, división, módulo (resto)	*, /, %
suma, resta	+, -
desplazamiento a izda., dcha. (bit a bit)	<<, >>
comparaciones	>, >=, <, <=
comparaciones	==, !=
"Y" bit a bit	&
"O" exclusiva bit a bit	^
"O" bit a bit	|
"Y" lógico	&&
"O" lógico	||
expresión condicional	*expr1* ? *expr2* : *expr3*
operadores de asignación	=, +=, -=, *=, ...
separador de evaluación de expresión	,

Los operadores unarios, expresión condicional y operadores de asignación se agrupan de dcha. a izda.; todos los demás de izda. a dcha.

Control de flujo

finalizador de instrucción	;
delimitadores de bloque	{ }
salir de **switch**, **while**, **do**, **for**	**break**
siguiente iteración de **while**, **do**, **for**	**continue**
valor de retorno de función	**return** *expr*

Construcciones de flujo

instrucción if	**if** (*expr*) *instrucción* **else if** (*expr*) *instrucción* **else** *instrucción*
instrucción while	**while** (*expr*) *instrucción*
instrucción for	**for** (*expr1*; *expr2*; *expr3*) *instrucción*
instrucción do	**do** *instrucción* **while**(*expr*);

instrucción switch	**switch** (*expr*) {
	case *const1*: *instrucción1* **break**;
	case *const2*: *instrucción2* **break**;
	default: *instrucción*
	}

Consulta de tipos de carácter <ctype.h>

c es un carácter

¿alfanumérico?	isalnum(c)
¿alfabético?	isalpha(c)
¿carácter de control?	iscntrl(c)
¿dígito decimal?	isdigit(c)
¿carácter imprimible (excl. espacio)?	isgraph(c)
¿letra minúscula?	islower(c)
¿carácter imprimible (incl. espacio)?	isprint(c)
¿carácter puntuación?	ispunct(c)
¿separador?	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 cadenas constantes

longitud de s	strlen(s)
copiar ct en s	strcpy(s,ct)
...hasta n caracteres	strncpy(s,ct,n)
concatenar ct tras s	strcat(s,ct)
...hasta n caracteres	strncat(s,ct,n)
comparar cs con ct	strcmp(cs,ct)
...sólo los primeros n caracteres	strncmp(cs,ct,n)
puntero al primer c en cs	strchr(cs,c)
puntero al último c en cs	strrchr(cs,c)

Entrada/Salida <stdio.h>

obtener un carácter	getchar()
imprimir un carácter	putchar(car)
imprimir con formato	printf("formato",arg1,. . .)
imprimir en cadena s	sprintf(s,"formato",arg1,. . .)
leer con formato	scanf("formato",&nombre1,. . .)
leer de cadena s	sscanf(s,"formato",&nombre1,. . .)
leer línea en cadena s	gets(s)
imprimir cadena s	puts(s)

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

`-`	alineación a izquierda
`+`	imprimir con signo
espacio	imprimir espacio si no hay signo
`0`	rellenar por delante 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 sin signo
c	carácter	s cadena de caracteres
f	doble	e,E exponencial
o	octal	x,X hexadecimal
p	puntero	n número de caracteres escritos
g,G	como f ó e,E según exponente	

Funciones útiles <stdlib.h>

entero pseudo-aleatorio en [0,RAND_MAX]	rand()
fijar la semilla aleatoria a n	srand(n)

Funciones matemáticas <math.h>

los argumentos y valores devueltos son **double**

funciones trigonométricas	sin(x), cos(x), tan(x)
funciones trig. inversas	asin(x), acos(x), atan(x) arctg(y/x) atan2(y,x)
funciones trig. hiperbólicas	sinh(x), cosh(x), tanh(x)
exponenciales y logaritmos	exp(x), log(x), log10(x)
exps. y logs. (base 2)	ldexp(x,n), frexp(x,*e)
división y resto	modf(x,*ip), fmod(x,y)
potencia y raíz	pow(x,y), sqrt(x)
redondeo	ceil(x), floor(x), fabs(x)