REFERENCIA RÁPIDA DE LENGUAJE C, SQL Y ALGEBRA DE BOOLE

Tipos básicos de datos

```
char            long
short           float
int             double
```

Operadores aritméticos

+ - * /(de menor a mayor prioridad)
% (operador módulo: resto de la división entera)
( ) (paréntesis, para agrupar operaciones)

Asignación de variables

```
variable_escalar = expresion;
variable_vector[indice] = expresion;
variable_estructura.campo = expresion;
```

Operadores relacionales

```
==  !=  <><=  >=
```

Operadores lógicos para condiciones

&&(AND lógico)|| (OR lógico)! (NOT lógico)

Constantes

```
#define NOMBRE_CTE valor
```
(no se termina en ;)

Condiciones

```
if (condicion) {
 bloque si condición cierta
}
else {
 bloque si condición falsa
}
```

OPCIONAL

Condicional múltiple IF-ELSE-IF

```
if (condicion1) {
bloque1
} else if (condicion2) {
 bloque2
} ...
else {
 bloque por defecto
}
```

OPCIONAL

Condicional múltiple SWITCH-CASE

```
switch (expresion) {
 case CONSTANTE1:
 ...
 break;
 case CONSTANTE2:
 ...
 break;
 default:
 ...
}
```

Bucle WHILE

```
while (condicion) {
 cuerpo del bucle
}
```

Bucle DO-WHILE

```
do {
 cuerpo del bucle
}while (condicion);
```

Bucle FOR

```
for (inicializ;condicion;actualiz) {
 cuerpo del bucle
}
```

Funciones

Prototipo (antes de main()):

```
tipo_devuelto nombre (tipo1 arg1, tipo2 arg2, .... , tipoN argN);
```

Implementación (después de main()):

```
tipo_devuelto nombre (tipo1 arg1, tipo2 arg2, .... , tipoN argN) {
   declaración de variables...
   bloque de código...
   return valor_a_devolver (si tipo_devuelto no es void)
}
```

Llamada a una función (ejemplos de uso):

```
 funcion(val1, val2,....);
```
(si funcion es void)

```
variable = funcion(val1, val2,...);
```
(si funcion no es void. En general, la llamada formará parte de una expresión)

Vectores (las cadenas son vectores de tipo char)

```
Tipo nombre[DIMENSION];
Índices válidos: 0 a DIMENSION-1
```

Matrices

```
Tipo nombre[FILAS][COLUMNAS];
Índices válidos: 0 a FILAS-1, 0 a
COLUMNAS-1
```

Estructuras

Definición:

```
struct nom_estructura {
 tipo1 campo1;
 tipo2 campo2;
 ...
};
```

Estructuras

Uso:

```
struct nom_estructura var;

var.campo1
var.campo2
...
```

Estructuras anidadas

Uso:

```
struct nom_estructura_anidada var;

var.campo1.campoX...
```

Biblioteca math.h (funciones matemáticas)

sqrt (v)	Raiz cuadrada de v
pow (x, y)	Eleva el valor x a la potencia y

Biblioteca stdio.h (funciones de manejo de pantalla, teclado y ficheros)

printf ("cadena de formato", exp1, exp2, ...);	Escribe el mensaje o valores de exp1, exp2 según las secuencias de sustitución de la cadena de formato
scanf ("cadena de formato", &variable);	Pide por teclado valores según se especifique en su cadena de formato y la asigna a una o más variables
"%d" "%c" "%x" "%f" "%lf" "%s"	Secuencias de sustitución para imprimir, o pedir por teclado: un entero decimal, un carácter, un valor hexadecimal, un valor float, un valor double, y una cadena (en scanf, los espacios terminan la cadena)
putchar(c);	Imprime el carácter cuyo código ASCII es c en pantalla

Biblioteca string.h (funciones de manejo de cadenas)

strcpy (a,b);	Copia la cadena b a la cadena a
strcat (a,b);	Añade por la derecha el contenido de la cadena b a la cadena a
strcmp (a,b);	Compara alfabéticamente la cadena a con b. Devuelve 0 si son iguales, 1 si a>b y -1 si a<b
strlen (a)	Devuelve el número de caracteres en la cadena a

Resumen de SQL

Sintáxis básica de una consulta:

SELECT [DISTINCT]
tabla1.campo1, tabla2.campo2,……
FROM tabla1, tabla2, tabla3……
WHERE condición……*(opcional)*
GROUP BY tabla.campo,……
(opcional)
ORDER BY tabla.campo,……
[DESC]*(opcional)*

tabla1,tabla2…. deben ser entidades, o relaciones de muchos a muchos.
GROUP BY da un resultado igual a DISTINCT sólo si los campos de GROUP BY son los mismos que hay en SELECT
Si en GROUP BY hay menos campos que en SELECT, los campos que haya en SELECT que no estén en GROUP BY sólo pueden ser campos agregados.
NOTA: Los [ ] alrededor de una palabra significan que esa palabra es opcional. Los corchetes **NO SE PONEN**.

Agregados:

```
COUNT() AVG()   MAX()   MIN()   SUM()
```

NOTACION DEL ALGEBRA DE BOOLE

$a + b$	suma lógica (OR)
ab **ó** $a \cdot b$	producto lógico (AND)
$\overline{a}$	negación lógica (NOT)
$\overline{a+b} = \overline{a}\,\overline{b}; \overline{ab} = \overline{a} + \overline{b}$	Ley de Morgan
$a + ab = a; a(a+b) = a$	Ley de absorción
$a \cdot a = a; a + a = a$	Ley de idempotencia
$a + \overline{a} = 1; a \cdot \overline{a} = 0$	Otras leyes