Tema II.- Introducción a la programación en C.

Estructura general de un programa

La estructura genérica de un programa en C es la siguiente

- #directivas del preprocesador, estas declaraciones se usan para indicar al compilador la declaración de:
  - cte. simbólica
  - inclusión de librerías del compilador con funciones específicas, tratamiento de cadenas, entrada/salida, funciones gráficas y de diversos tipos. Se irán indicando los nombres de las librerías a medida que se necesite su uso.

- Prototipos de funciones(cuando estas existan en el programa, se verán en el tema 3)

```
main()
{
declaración de variables(tipos de datos manejados por el programa)
bloque del programa
-
-
-
-
}
funciones (se verán en el tema 3)
```

todas las instrucciones del bloque deben finalizar con ;

Inclusión de comentarios

- En un programa en C, se pueden (y deben) poner comentarios.
- La función de los comentarios es explicar el código fuente del programa en C para su estudio y comprensión.
- Los comentarios pueden ir en cualquier parte del programa aunque es aconsejable que vayan en la zona que pretenden explicar.
- Las formas de indicar expresiones de comentario son las siguientes:

/* texto del comentario */

Esta forma puede ser multilinea es decir se pueden hacer comentarios de mas de una línea

- La otra (estilo de C++) es la siguiente:

//texto comentario

- Esta forma solo permite escribir una línea de comentario
- Nuestro compilador puede usar las dos.

Declaración de constantes simbólicas

Se realizan antes del main() de la siguiente forma:
#define nombre valor (sin ;)

Ejemplos:

#define PI 3.14151592
#define MENSAJE "Ha ocurrido un error pulse return"

- Cuando el compilador compile el programa (traducción a código ejecutable) sustituirá PI por el valor indicado y mensaje por el literal indicado y trabajará con ellos de forma normal, obviamente el tipo de una cte. es el tipo de valor asignado y no pueden ser modificadas durante el transcurso del programa o se producirá un error.

- El uso de constantes es muy importante para parametrizar programas, si por ejemplo ejecuto un programa donde la longitud de los bucles es 100 y luego deseo que sea 200 y tengo 10 bucles es aconsejable definir una cte. K (p.ej) como 100 y sólo con un cambio en una línea tendré hecho el cambio en los 10 bucles del programa.

- El código de C (sentencias y variables) hay que escribirlo siempre en **minúsculas**, pues el tipo de letra es significativo, por ello las constantes se ponen siempre en mayúsculas.

Declaración de variables

Todas las variables de un programa deben declararse antes de su uso aunque por legibilidad del programa es mejor hacerlo en el comienzo del programa main() principal

Los tipos de variables predefinidos por el compilador de C así como su tamaño en bits y el rango permitido para almacenar viene dado por la siguiente tabla :

Data Types (32-bits)

Tipo de variable	long.	Rango
unsigned char	8 bits	0 to 255
char	8 bits	-128 to 127
short int	16 bits	-32,768 to 32,767
unsigned int	32 bits	0 to 4,294,967,295
int	32 bits	-2,147,483,648 to 2,147,483,647
unsigned long	32 bits	0 to 4,294,967,295
enum	16 bits	-2,147,483,648 to 2,147,483,647
long	32 bits	-2,147,483,648 to 2,147,483,647

float	32 bits	3.4 x 10-38	to	3.4 x 10+38
double	64 bits	1.7 x 10-308	to	1.7 x 10+308
long double	80 bits	3.4 x 10-4932	to	1.1 x 10+4932

- En muchas (pero no todas) las implementaciones de C y C++ un long es mayor que un int. Actualmente, la mayoría de las aplicaciones de oficina y personales, con entornos como Windows o Linux, corren sobre plataformas hardware de 32 bits, de forma que la mayoría de los compiladores para estas plataformas utilizan un int de 32 bits (del mismo tamaño que el long).
- Aunque esta tabla parece muy farragosa en cuanto a la cantidad de tipos , en realidad los tipos más utilizados en los programas de gestión , que son los que nos ocuparán son:
  - int (enteros)
  - char (caracteres)
  - float (reales en coma flotante)

- Por ejemplo para declarar a,b como enteros y r como real, se haría del siguiente modo:

```
main()
{
int a,b;
float r;
...
...
```

- Se permite si se desea en el momento de declarar variables inicializarlas

Por ejemplo :

```
int a=0;
float r=3.45;
```

Operadores en C

- Aritméticos

Las cinco reglas matemáticas son:

```
+
-
*
/
```

% (MOD resto de la división)

- Relacionales

Operador	Significado
>	Mayor que
>=	Mayor o igual que
<	Menor que

<=	Menor o igual que
==	Igual (dos símbolos =)
!=	Distinto que

- Lógicos

Operador	Significado
&&	Y
||(doble barra vertical)	O
!	No

Instrucciones de un programa

Asignación

El operador de asignación es el conocido = (pseudocódigo)
También existen en C otros operadores de asignación combinados con cálculo y son los siguientes:

Operador	Acción que se realiza
++	Incremento x++; equivale a x=x+1;
--	Decremento x-- equivale a x=x-1;
Estos operadores son prioritarios sobre *,/,+,-	

Funciones de entrada/salida

Las funciones de entrada salida están contenidas en la librería del sistema denominada <stdio.h>, es decir para utilizar estas dos funciones que a continuación se explican habrá que poner en la cabecera la declaración de la directiva:

#include <stdio.h>
Siempre que utilicemos funciones del sistema del compilador habrá que indicar la librería donde están incluidas

Función de entrada de datos

scanf("formatos de entrada ó conversión ",lista de direcciones de memoria de las variables a leer separadas por comas)

Donde los formatos de entrada pueden ser según el tipo de variable a leer los siguientes:

Tipos de datos	Especificaciones de conversión printf	Especificaciones de conversión scanf
long double	%Lf	%Lf
double	%f	%lf
float	%f	%f
unsigned long int	%lu	%lu
long int	%ld	%ld
unsigned int	%u	%u
int	%d	%d
short	%hd	%hd
char	%c	%c

Fig. 5.5 Jerarquía de promoción para tipos de datos.

Tabla de conversión de formatos para printf y scanf

En este caso los formatos permitidos están en la segunda columna, la primera son los formatos necesarios para imprimir, de la función printf que veremos a continuación.

Ejemplo: si deseo leer un int llamado dato, la llamada a la función para leerlo será:

scanf("%d",&dato)

En la lista de variables se pone el nombre de la leída precedido del operador & que es el operador que da la dirección de memoria RAM (de datos) donde reside dicha variable.

Puede leerse más de un dato, en un solo scanf pero en este caso, los datos deben introducirse con espacio en blanco como separador o con enter como separador y los formatos deben ser compatibles con cada tipo de dato.
Ejemplo(para leer tres enteros)

scanf("%d%d%d",&a,&b,&c);

La introducción por el teclado debería ser:

23 45 600
o bien
23 (enter)

45 (enter)
600 (enter)
con lo cual los valores pasarían a cada variable en el orden que se especifican en scanf.

Consejo: para evitar problemas con la separación de datos es mejor pedirlos de uno en uno en varias funciones de lectura lo que nos ahorrará problemas

Función de salida de datos

La función que permite escribir datos en la pantalla también incluida en la librería <stdio.h> es la siguiente

printf("Mensajes literales y formatos de salida o conversion",lista de variables para imprimir separadas por comas)

Los formatos de salida admitidos están en la primera columna de la tabla anterior

Ejemplo:
```
printf("Los datos son: %d  y su media %f",cuantos,media);
```
escribirá el literal puesto y en la parte del formato asignara una salida a cada variable de la lista ajustándola a su formato correspondiente, es decir escribirá cuantos como entero y media como real, obviamente los datos y los formatos deben ser compatibles en el orden indicado..

Modificadores de precisión

Si deseamos (opcionalmente) que nuestra salida se escriba bajo una máscara de precisión concreta (anchura de campos) hay que utilizar los denominados modificadores de formatos, éstos son los siguientes:(son usados para alinear datos en columnas)

Para datos reales formato %f

%n1.n2f esto indica que la salida debe hacerse con n1digitos(incluyendo la coma) como máximo y con n2 decimales
%0n1.n2f igual que el anterior pero rellena el espacio sobrante con ceros

Para datos enteros con formato %d

%n1d muestra n1 dígitos del entero
%0n1d muestra n1digitos y si no se rellena con ceros

- En ambos casos si el número tiene unos dígitos superiores a n1 el dato se muestra por completo con todas sus cifras.

- Todos los datos se justifican a la derecha si se quieren justificar a la izquierda debe ponerse después del % el símbolo -

Secuencias de escape

Existen determinados caracteres que usados en la función printf provocan efectos de salida en pantalla son 8 ó 9 pero los mas habituales son :

- \t tabulador horizontal
- \n salto de línea
- \a pitido del altavoz del ordenador

Normalmente se usa el \n

Ejemplo
printf("hola que tal estas\n");
Escribe el literal por la pantalla y despues salta a la línea siguiente.

Sentencias de toma de decisión

La sentencia de toma de decisión correspondiente al Si de pseudocódigo es

```
if  (expresion lógica v/f)
{
        s1;
        s2;
        -
        -
        sn;
}
else
{
        r1;
        r2;

        rn;
}
```

- Si se anidan varios if's deben especificarse muy bien el ámbito donde actúan para no tener problemas.

- Si después de un if solo se ejecuta una sentencia no es necesario poner corchetes.

Sentencia de selección múltiple

```
 switch(expresion)
{
case constante1:
sentencias separadas por;
break;
case constante2:
sentencias separadas por;
-
```

-
-
case constante-n :
sentencias separadas por;
break;
default:
sentencias separadas por ;
}

En este caso se evalúa la expresion entre paréntesis y se ejecuta el bloque correspondiente al valor de la constante que iguale a dicha expresion, el break hace que una vez ejecutado dicho bloque se salga de la estructura switch, si ninguna constante es igual a la expresion se ejecuta la opción default

- La cláusula break es opcional si no se pusiera seguiría comprobando el resto de cláusulas case
- La cláusula default es opcional y si ninguna constante tiene el valor de la expresion, no se ejecuta ningún bloque
- Esta sentencia es típica para construir y direccionar tareas de un menú de opciones siendo más clara que varios if seguidos
- La diferencia con un if es que se compara con valores constantes mientras que el if es más flexible porque permite rangos de valores con expresiones de relación de orden(>,<,<= ,>=,==,!=).

Repetición controlada por contador(para--finpara de pseudocódigo)

Su sintaxis es la siguiente:

for(inicializar variables contador;condicion de repeticion;incremento o decremento de variable contador)
{
s1;
s2;
-
-
sn;
}

for (t=0;t<=10;t++)
{
-
-
-
}
inicializa t a cero y mientras esta sea menor o igual que 10 ejecuta el bloque entre corchetes, la variable t se va incrementado con paso 1(t++)

Variantes del for

- La condición no tiene por que implicar sólo al contador, puede ser más general

Ejemplo:

for(x=0;x+y<10 && z!=0;x++)

- Puede haber en la tercera parte múltiples incrementos

Ejemplo:

for(x=0;x<=10;x++,y++)

- Pueden inicializarse varias variables aunque no sean el contador

Ejemplo:

for(x=0,y=0;x<=10;x++)

Repetición condicional mientras--finmientras

Su sintaxis es :

while(expresion lógica v/f)
{
bloque de sentencias separados por ;
}

Primero se evalúa la expresion si es cierta se ejecuta el bloque y se vuelve a evaluar, para seguir hasta que sea falsa

Repetición condicional repetir -mientras

do
{
bloque de sentencias separados por ;
}
while(expresion lógica);

Se ejecuta el bloque al menos una vez y se sigue ejecutando mientras la condición sea cierta.

Hay dos instrucciones adjuntas a las repeticiones y al if que son

- **break** : fuerza la salida de un **while,do ó for** en el momento que aparece
- **continue**: fuerza a un **while, do o for** a ejecutar el siguiente paso del bucle

Si hay varias repeticiones encajadas solo se fuerza la salida donde esta incluido el break o continue.