T5LP45ATP1

Nombre: *Sánchez Bonaldo, Camila*

FOR

Esta instrucción implementa un bucle, es decir repite un grupo de sentencias un número determinado de veces. Para utilizarlo se deben especificar tres partes diferentes separadas por ';' (punto y coma). Estas partes son:

Inicialización: En esta parte se realiza la asignación que fija el valor inicial de la variable que va a utilizarse como contador de iteraciones del bucle.

Condición: En esta parte se especifica una condición, justo antes de cada iteración se comprobará que sea cierta para pasar a ejecutar el grupo de sentencias. Si la condición se evalúa como falsa, se finalizará el bucle.

Incremento: Es la última de las tres partes, es donde se indica el incremento de la variable usada como contador por cada iteración del bucle.

```
for (inicialización; condición; incremento){
      sentencia;
    ...
}
```

EJEMPLO:

```
for (i=0; i<10; i=i+1){
      sentencia;
    ...
}
```

Este bucle se ejecutaría la primera vez con la variable **i** valiendo **0**, la segunda valiendo **1**, ..., y la última valiendo **9**; tras esta iteración se ejecutaría la parte del incremento, pasando **i** a valer **10** y, entonces, al comprobarse la condición de permanencia en el bucle (que **i** sea menor que **10**) y resultar ésta falsa, se daría por finalizado el bucle.

WHILE

Al igual que la sentencia anterior, esta implementa un bucle.

Para implementar este bucle, se debe identificar entre paréntesis la condición que se debe cumplir para que se ejecute el grupo de sentencias. Tras especificar esta condición en el bloque '{}', se pondrán las sentencias que se repitan.

Finalmente, cuando se ejecute una sentencia while se realizará la comprobación que se especifica, si esta resulta cierta, se ejecutaran las sentencias anteriores, en caso contrario, el programa continuará y saldrá del bucle.

```
while (condición){
      sentencia;
    ...
}
```

EJEMPLO:

```
i=0;
while (i<150){
	x=x+10;
	FRAME;
}
```

En este ejemplo se establece a la variable i a 0 y después, mientras i sea menor que 150, se le sumarán 10 a i y se dará un FRAME

SWITCH

Esta sentencia presenta una serie de sentencias case y, opcionalmente, una sección default.

Cuando se ejecuta una sentencia switch, primero se evalúa la expresión y después, si el resultado está dentro del rango de valores contemplados en la primera sección case, se ejecutarán las sentencias de la misma y se dará por finalizada la sentencia. En caso de no estar el resultado de la expresión en el primer case se pasará a comprobarlo con el segundo case, el tercero,
etc. Y, por último, si existe una sección default y el resultado de la expresión no ha coincidido con ninguna de las secciones case, entonces se ejecutarán las sentencias de la sección default. En una sección case solo se puede especificar un valor.

```
switch (expresión){
	case valor:
	sentencia;
	...
break;
...

default:

	sentencia;
	...
break;

}
```

EJEMPLO:

```
switch (i){
	case -2:
		x=-2;
	break;
	case 1:
		x=1;
	break;
	case 105:
		x=105;
	break;
	default 0:
		x=0;
	break;

}
```

La sentencia switch de este programa cambiará de signo la variable i si ésta vale -2, 1, o 105; en caso contrario la pondrá a 0.

IF

La sentencia if sirve para ejecutar un bloque de sentencias opcionalmente, cuando se cumpla una condición. En otra variante, se ejecutará además otro bloque de sentencias (dentro de la sección else) cuando la condición no se cumpla. Es posible anidar sentencias if sin ningún límite, es decir, se pueden poner más sentencias if dentro de la parte que se ejecuta cuando se cumple la condición (parte if) o dentro de la que se ejecuta cuando la condición no se cumple (parte else).

```
if (condición){
    sentencia;
    ...
}
```

(o bien)

```
if (condición){
    sentencia;
    ...
}else{
    sentencia;
    ...
}
```

EJEMPLO:

```
if ((x > 100) AND (x < 220)){
        y=y+4;
    }else{
        y=y-8;
    }
```

Si x es mayor que 100 y menor que 220, al valor de y se le sumara 4. Si x no cumple con la condición, se le restara 8 al valor de y.

Enteros (int): Son un tipo de datos se usa normalmente para representar números enteros. C no define el rango de valores que podemos representar con una variable de este tipo. Lo más frecuente es que ocupe 32bits. Como los enteros se codifican en complemento a 2, el rango de valores que podemos representar es [−2147483648, 2147483647], es decir, $[-2^{31}, (2^{31})-1]$.

Flotante (float): Estos tipos de datos representa números en coma flotante de 32bits. La codificación de coma flotante permite definir valores con decimales. El máximo valor se puede almacenar en este tipo de variable es 3.40282347 * 10^{38}, en un programa C se codifica así: 3.40282347 * e^{38}. El número mínimo que podemos representar es 1.17549435 * 10^{-38}, que se codifica así: 1.17549435 * e^{-38}.

Carácter (char): Representa en complemento de 2 con un solo byte (8bits), el rango de valores que puede tomar esta variable es solo de [-128, 127]. Es frecuente el uso de este tipo de variable para almacenar caracteres codificados en ASCII

Se los llama tipos de datos porque es la propiedad de un valor que determina su dominio (qué valores puede tomar), qué operaciones se le pueden aplicar y cómo es representado internamente.

PROGRAMA 1

El programa declara 2 variables de tipo entero ‘a’ y ‘m’. Luego imprime en pantalla la frase “teclea el numero de mes” donde el usuario ingresa el número correspondiente al mes, dicho número se guarda en la variable m. Si m es igual a 1,3,5,7,8,10 o 12; se imprime en pantalla “el mes ‘numero de mes’ tiene 31 días”. Si m es igual a 2, se imprime en pantalla “teclea el año”, luego el usuario digita un número y este se almacena en la variable a. Si a es divisible por 4 y por 100 o a es divisible por 400, se imprime en pantalla “el

mes 2 tiene 29 días". Si no se cumple la condición se imprime "el mes 2 tiene 28 días". Si m es igual a 4, 6, 9 u 11, se imprime en pantalla "el mes 'número del mes' tiene 30 dias".

PROGRAMA 2

Este programa declara una variable de tipo entero 'n'. Se imprime en pantalla "teclea un numero", luego el usuario ingresa un número y se almacena en n. Si el número dividido 7 da resto cero (o sea si es divisible por 7) se muestra en pantalla "el numero 'valor de n' es múltiplo de 7". En caso que no se cumpla la condición, se imprimirá en pantalla "el numero 'valor de n' no es múltiplo de 7".

PROGRAMA 3

El programa presenta 3 variables 1 tipo char 'op' y 2 de tipo entero 'x1' y 'x2'. En pantalla se imprime un título "----------DIVISION ENTERA----------". Se muestra en pantalla "Teclea un numero entero", el usuario digita un número que se guarda en la variable x1. Luego se repite el proceso, pero el nuevo dato se almacena en la variable x2. Después se despliegan una serie de opciones que se imprimen en pantalla

"+= Suma
-=Resta
*=Multiplicacion
/=Division"

Debajo del menú se imprime en pantalla "Que operacion deseas" el usuario digita el carácter correspondiente a la operación que desea realizar y dicho dato se almacena en la variable op. En el caso que op corresponda a + se imprime en pantalla " 'x1' + 'x2' = 'resultado de x1+x2'. Si op es igual a – se imprime en pantalla " 'x1' – 'x2' = 'resultado de x1-x2". En caso de que op sea equivalente a * se imprimirá en pantalla " 'x1'*'x2' = 'resultado de x1*x2'. Si op es / entonces se muestra en pantalla " 'x1' / 'x2' = 'resultado de x1/x2'. Si el carácter ingresado no corresponde a ninguna de las opciones, se imprimirá en pantalla " 'No existe esa opcion'.