**SENTENCIAS DE CONTROL REPETITIVAS.**

**1. Introducción.**

Veremos todos los conceptos de teoría: las Sentencias de Control Repetitivas, que permiten repetir un trozo de código un número de veces determinadas.

Estudiaremos los tres tipos de estructuras de control repetitivas o iterativas que ofrece el Lenguaje C: while, do-while y for.

**2. Conceptos nuevos**

Los conceptos nuevos introducidos son:

1) Estructura y funcionamiento de las Sentencias de Control Repetitivas.

2) Uso de una variable como contador.

**3. Lenguaje C**

Las nuevas características de C que ha aprendido y que necesita saber para la realización de esta práctica son:

1) Funcionamiento y Diagrama de Flujo de la Sentencia de Control while

2) Funcionamiento y Diagrama de Flujo de la Sentencia de Control do-while

3) Funcionamiento y Diagrama de Flujo de la Sentencia de Control for

**4. Entorno DevC++.**

Para esta práctica 5 es totalmente imprescindible que sepa usted hacer el seguimiento y depuración de un programa ejecutándolo paso a paso, utilizando para ello las herramientas que ofrece el EID DevC++ y que han sido comentadas en prácticas anteriores. Además, sería conveniente que investigara y averiguara para qué sirven las siguientes opciones del DevC++:

• Depurar Æ Añadir/Quitar Punto de Ruptura (Ctrl F5)

• Depurar Æ Saltar Paso (Ctrl F7)

• Depurar Æ Ir a cursor (Shift F4)

Estas opciones se vuelven realmente importantes cuando queremos ejecutar un programa paso a paso que contiene bucles, ya que nos facilitan enormemente nuestro trabajo. Descubra cómo funcionan probándolas directamente y pensando un poco, y acudiendo a los manuales, libros o tutoriales cuando sea necesario (recuerde que deben ser manuales donde se explique el entorno DevC++).

**5. ¿Qué sentencia de control repetitiva utilizar?**

A continuación mostramos una breve comparativa de uso de los tres tipos de Sentencia de Control Repetitivas vistas (while, do-while y for), a fin de que entienda fácilmente en qué casos utilizar una u otra:

for 🡨 El uso más frecuente de este tipo de bucles se produce cuando el número de repeticiones se conoce por anticipado y la condición del bucle puede ser controlada por un contador.

While🡨 El uso más frecuente se produce cuando la repetición del bucle no está controlada por un contador sino por una cierta condición (simple o compleja). El bucle puede que no se ejecute ninguna vez

do-while 🡨 Se utiliza en las mismas condiciones que el while y cuando además se debe asegurar que el bucle se ejecute al menos una vez. Ejemplo: menú de opciones con filtro.

**6. Técnicas de prueba de un bucle.**

En general las técnicas de pruebas son métodos o técnicas cuyo objetivo es determinar si un programa (o un trozo de código) funciona correctamente. El objetivo final de una estrategia de prueba consiste en detectar errores.

Existen distintos tipos o técnicas de prueba. Nosotros nos centraremos en aquella que nos permite probar si un bucle está bien construido o no, esto es, averiguar mediante un número de pruebas si el bucle funciona correctamente.

Sea N el número máximo de pasos permitidos o iteraciones que da un bucle. A la hora de comprobar formalmente si un bucle funciona correctamente, deberemos probarlo con los siguientes casos:

a) Probar el bucle pasándolo por alto completamente.

b) Pasar sólo una vez por el bucle.

c) Pasar dos veces por el bucle.

d) Hacer que el bucle de m vueltas, siendo m<N

e) Hacer que el bucle de N-1 vueltas

f) Hacer que el bucle de N vueltas

g) Intentar que el bucle de N+1 vueltas.

Como podrá imaginar a veces no es necesario realizar todos los casos. Esto dependerá del tipo de bucle y de la condición que tenga ese bucle. Lo que sí es muy interesante que entienda es que los fallos en un bucle se suelen dar en la condición de salida del mismo, y que esa condición, que puede provocar que el bucle se ejecute una vez más o una vez menos de lo que queremos que se ejecute, sí es importante que sea muy bien analizada (aplicando los pasos e), f) y g) de los anteriormente propuestos).