GUÍA DE APRENDIZAJE PARA CONCURSANTES ICPC Y IOI: INSTRUCCIÓN BUCLE DO WHILE

1. Introducción

En el diseño e implementación de un algoritmo el mismo no siempre va ser de forma lineal (que se ejecuten cada instrucción una detrás de la otra) sino que llegado determinado paso del algoritmo el mismo debe ser capaz repetir una o un cojunto de instrucciones mientras se cumpla una determinada condición y cuando esta deje de cumplirse seguir con el resto de las instrucciones del programa. Para poder realizar es necesario el uso de estructura de control las cuales permiten modificar el flujo de ejecución de las instrucciones de un algoritmo.

Vamos analizar dentros de la estructura de control las sentencias de bucle que realizan un grupo de instrucciones chequea una pregunta la cual retorna verdadero o falso (evalúa una condición) si es verdadero repite el proceso anterior sino continua con el resto de las instrucciones. Especificamente veremos la instrucción **do { }while;**

2. Conocimientos previos

La instrucción **break** interrumpe la ejecución del bucle donde se ha incluido, haciendo al programa salir de él aunque la expresion de control correspondiente a ese bucle sea verdadera. La sentencia **continue** hace que el programa comience el siguiente ciclo del bucle donde se halla, aunque no haya llegado al final de las sentencia compuesta o bloque.

Un **bucle** se utiliza para realizar un proceso repetidas veces. Se denomina también **lazo** o **loop**. El código incluido entre las llaves {} (opcionales si el proceso repetitivo consta de una sola línea), se ejecutará mientras se cumpla unas determinadas condiciones. Hay que prestar especial atención a los bucles infinitos, hecho que ocurre cuando la condición de finalizar el bucle (booleanExpression) no se llega a cumplir nunca. Se trata de un fallo muy típico, habitual sobre todo entre programadores poco experimentados.

3. Desarrollo

Esta sentencia funciona de modo análogo a **while**, con la diferencia de que la evaluación de **expresion_de_control** se realiza al final del bucle, después de haber ejecutado al menos una vez las sentencias entre llaves; éstas se vuelven a ejecutar mientras **expresion_de_control** sea **true**. La forma general de esta sentencia es:

```
do{
   sentencia;
} while(expresion_de_control);
```

donde **sentencia** puede ser una única sentencia o un bloque, y en la que debe observarse que *hay que poner (;) a continuación del paréntesis* que encierra a **expresion_de_control**, entre otros motivos para que esa línea se distinga de una sentencia **while** ordinaria.

La sentencia **do** es usada generalmente en cooperación con **while** para garantizar que una o más instrucciones se ejecuten al menos una vez.

4. Implementación

La implementación de esta estructura tanto en Java y C++ no varía en nada son identicamente iguales.

4.1. Java

```
int contador = 0;
do{
   contador++;
   System.out.print(" Hola Mundo ") ;
}while( contador < 10) ;
```

4.2. C++

```
int contador = 0;
do{
   contador++;
   cout<<" Hola Mundo " ;
}while( contador < 10) ;
```

5. Complejidad

El tiempo de ejecución de un bucle de sentencias WHILE C DO S END; es T = T(C) + (no iteraciones)*(T(S) + T(C)). Obsérvese que tanto T(C) como T(S) pueden variar en cada iteración, y por tanto habrá que tenerlo en cuenta para su cálculo. Donde:

1. **T(C) :** Es la complejidad de la expresión de control que se define dentro de los parentisís del while.
2. **T(S) :** Es la complejidad del bloque de instrucciones que conforman las instrcciones que serán ejecutadas si la expresión de control del while es verdadera.

6. Aplicaciones

Este tipo de bucles se utiliza con frecuencia para controlar la satisfacción de una determinada condición de error o de convergencia.

Otra de la utilización de este tipo de bucle aunque no es muy común para los ejercicios de concursos (pero por esos no vamos obviar su existencia) es cuando tenemos un grupo de instrucciones la cual tenemos que repetir una cantidad de veces donde sabemos que la minima cantidad de veces siempre es es uno en esos casos es útil esta estructura.