GUÍA DE APRENDIZAJE PARA CONCURSANTES ICPC Y IOI: INSTRUCCIÓN BUCLE 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 una pregunta la cual retorna verdadero o falso (evalúa una condición) y realizan un grupo de instrucciones repetitivamente mientras la respuesta a la pregunta sea verdadera. Especificamente veremos la instrucción **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 es una de las instrucciones de bucles más simples de comprender. Como su significado en inglés lo indica (mientras), la repetición del bloque de instrucciones asociado se realizará "mientras"se cumple una condición determinada. Obviamente, es muy importante que, dentro del bloque de instrucciones, exista la posibilidad de que esa condición varíe y deje de cumplirse; de lo contrario, entraríamos en un ciclo de repetición infinito. La forma general es como sigue:

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

Se evalúa **expresion_de_control** y si el resultado es **false** se salta sentencia y se prosigue la ejecución. Si el resultado es **true** se ejecuta sentencia y se vuelve a evaluar **expresion_de_control** (evidentemente alguna variable de las que intervienen en **expresion_de_control** habrá tenido que ser modificada, pues si no el bucle continuaría indefinidamente, se produciría lo que se conoce

como un ciclo infinito). La ejecución de sentencia prosigue hasta que **expresion_de_control** se hace **false**, en cuyo caso la ejecución continúa en la línea siguiente a sentencia. En otras palabras, sentencia se ejecuta repetidamente mientras **expresion_de_control** sea **true**, y se deja de ejecutar cuando **expresion_de_control** se hace false. Obsérvese que en este caso el control para decidir si se sale o no del bucle está antes de sentencia, por lo que es posible que sentencia no se llegue a ejecutar ni una sola 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;
while ( contador <= 10)
{
   contador = contador +1;
   System.out.println("Hola Mundo") ;
}
```

4.2. C++

```
int contador = 0;
while ( contador <= 10)
{
   contador = contador +1;
   cout<<"Hola Mundo" ;
}
```

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

El uso de la estructura de control de tipo bucle especificamente la estructura while dentro de los ejercicios de concursos es muy utilizada no solo en algoritmos clásicos sino tambíen como parte de la estructura de solución de muchos ejercicios donde el juego de datos de entrada comienza especificando la cantidad de casos de pruebas a probar por tu algoritmo. Por ejemplo veamos la siguiente especificación de entrada de datos de un ejercicio:

La primera línea contendrá un número entero N (1$\leq$ N $\leq$ 100 000), el número de problemas de suma que Tudor necesita resolver. Las siguientes líneas N contendrán cada una dos números enteros separados por espacios cuyo valor absoluto es menor que 1 000 000 000, los dos números enteros que Tudor necesita sumar.

Veamos la estructura solución de este problema utilizando el **while** con C++

```
/*La variable casos es de tipo entera y previamente se leyo de consola su
   valor y almacena el valor de la cantidad de casos a probar*/
while (casos > 0){
   /*Decremento la cantidad de casos de pruebas que me queda por resolver*/
   casos= casos -1;

   /*Declaro las variables donde voy alamcenar los valores a sumar, podria
      poner esta linea arriba del while sin porblema y me ahorarria memoria*/
   long log a,b;

   /*Leo los valores para el caso en cuestion*/
   cin>>a>>b;

   /*Sumo e imprimo los valores*/
   cout<<(a+b)<<endl;
}
```

En el código anterior lo importante es ver como el algoritmo principal del problema (sumar dos numeros) lo encapsulamos dentro de un while el cual nos va servir para controlar la cantidad de instancias del mismo problema que nuestro algoritmo debe ser capaz de resolver.

Cualquiera de las otras estructuras de control de tipo bucle puede ser convertidas o representadas utilizando la estructura while siempre con un menor o mayor grado de complejidad de implementación dependiendo de la situación.

7. Ejercicios propuestos

A continuación una lista de ejercicios que se resuelven utilizando esta instrucción:

- Interés Bancario.
- Sum

- A Plus B
- ¿Cuántas fichas de dominó? (I)
- Calculando áreas
- A - A + B
- B - A - B