GUÍA DE APRENDIZAJE PARA CONCURSANTES ICPC Y IOI: INSTRUCCION ALTERNATIVA IF

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 de decidir que cojunto de instrucciones se debe realizar o que conjunto de instrucciones se debe omitir. 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 decisión que realizan una pregunta la cual retorna verdadero o falso (evalúa una condición) y selecciona la siguiente instrucción a ejecutar dependiendo la respuesta o resultado. Especificamente veremos la instrucción **if**

2. Conocimientos previos

2.1. Operadores relacionales

Los operadores relacionales sirven para realizar comparaciones de igualdad, desigualdad y relación de menor o mayor. El resultado de estos operadores es siempre un valor boolean (true o false) según se cumpla o no la relación considerada.

Operador	Significado	Utilización	Es verdadero
$<$	Menor que.	op1 $<$ op2	si op1 es menor que op2
$<=$	Menor o igual que.	op1 $<=$ op2	si op1 es menor o igual que op2
==	Igual a.	op1 $==$ op2	si op1 y op2 son iguales
$>$	Mayor que.	op1 $>$ op2	si op1 es mayor que op2
$>=$	Mayor o igual que.	op1 $>=$ op2	si op1 es mayor o igual que op2
$!=$	Distinto que	op1 $!=$ op2	si op1 y op2 son diferentes

2.2. Operadores lógicos

Los operadores lógicos && (and), || (or) y ! (not) son utilizados cuando evaluamos expresiones que poseen varios operandos relacionales para obtener un resultado simple (verdadero o falso) que las relaciona. El operador && es el operador and de la lógica booleana. Esta operación resulta verdadera si todos los operandos son verdaderos y falsos en caso contrario. El operador || es el operador or de la lógica booleana y resulta verdadera si al menos uno de los operandos es verdadero. Por último el operador ! es el operador not de la lógica booleana que niega(si es verdadero lo convierte en falso y viceversa) el resultado de la expresión que le sucede.

Operador	Nombre	Utilización	Resultado
&&	AND	op1 && op2	true si op1 y op2 son true. Si op1 es false ya no se evalúa op2
||	OR	op1 | | op2	true si op1 u op2 son true. Si op1 es true ya no se evalúa op2
!	NOT	!op	true si op es false y false si op es true
&	AND	op1 & op2	true si op1 y op2 son true. Siempre se evalúa op2
|	OR	op1 | op2	true si op1 u op2 son true. Siempre se evalúa op2

3. Desarrollo

3.1. Bifurcación if

Esta estructura permite ejecutar un conjunto de sentencias en función del valor que tenga la expresión de comparación (se ejecuta si la expresión de comparación tiene valor true). Tiene la forma siguiente:

```
if (booleanExpression) {
   statements;
}
```

Las llaves {} sirven para agrupar en un bloque las sentencias que se han de ejecutar, y no son necesarias si sólo hay una sentencia dentro del if.

3.2. Bifurcación if else

Análoga a la anterior, de la cual es una ampliación. Las sentencias incluidas en el else se ejecutan en el caso de no cumplirse la expresión de comparación (false),

```
if (booleanExpression) {
   statements1;
} else {
   statements2;
}
```

3.3. Bifurcación if elseif else

Permite introducir más de una expresión de comparación. Si la primera condición no se cumple, se compara la segunda y así sucesivamente. En el caso de que no se cumpla ninguna de las comparaciones se ejecutan las sentencias correspondientes al else.

```
if (booleanExpression1) {
   statements1;
```

```
} else if (booleanExpression2) {
   statements2;
} else if (booleanExpression3) {
   statements3;
} else {
   statements4;
}
```

En esta estructura se puede omitir del bloque **else** de ser necesario.

4. Implementación

La implementación de esta estructura es similar tanto en C++ y Java asi que vamos a ver algunos ejemplos el uso de esta estructura.

4.1. Java

Si la calificación es igual o superior a 60 el estudiante esta aprobado

```
if ( calificacionEstudiante >= 60 )
   System.out.println( "Aprobado" );
```

Si la calificación es igual o superior a 60 el estudiante esta aprobado sino esta reprobado

```
if ( calificacion >= 60 )
   System.out.println( "Aprobado" );
else
   System.out.println( "Reprobado" );
```

Acorde al valor de la calificación y en el rango que esta ubicado dicha calificación será la repuesta.

```
if ( calificacionEstudiante >= 90 )
   System.out.println( "A" );
else if ( calificacionEstudiante >= 80 )
   System.out.println( "B" );
else if ( calificacionEstudiante >= 70 )
   System.out.println( "C" );
else if ( calificacionEstudiante >= 60 )
   System.out.println( "D" );
else
   System.out.println( "F" );
```

4.2. C++

Si la calificación es igual o superior a 60 el estudiante esta aprobado

```
if ( calificacionEstudiante >= 60 )
   cout<<"Aprobado";
```

Si la calificación es igual o superior a 60 el estudiante esta aprobado sino esta reprobado

```
if ( calificacion >= 60 )
   cout<<"Aprobado";
else
   cout<<"Reprobado";
```

Acorde al valor de la calificación y en el rango que esta ubicado dicha calificación será la repuesta.

```
if ( calificacionEstudiante >= 90 )
   cout<<"A";
else if ( calificacionEstudiante >= 80 )
   cout<<"B";
else if ( calificacionEstudiante >= 70 )
   cout<<"C";
else if ( calificacionEstudiante >= 60 )
   cout<<"D";
else
   cout<<"F" ;
```

5. Complejidad

El tiempo de ejecución de la sentencia del tipo $IF \quad C \quad THEN \quad S_1 \quad ELSE \quad S_2 \quad END$; es $T = T(C) + \text{máx}(T(S_1), T(S_2))$ donde:

- $T(C)$: Complejidad de la expresión que establece la condición de la instrucción if.
- $T(S_1)$: Complejidad de las instrucciones que coforman el bloque que se ejecutará en caso que la expresión que se establece en la condición de la instrucción if sea verdadera.
- $T(S_2)$: Complejidad de las instrucciones que coforman el bloque que se ejecutará en caso que la expresión que se establece en la condición de la instrucción if sea falsa.

6. Aplicaciones

No todos los problemas pueden resolverse empleando estructuras secuenciales. Cuando hay que tomar una decisión aparecen las estructuras condicionales. En multiples ocasiones en los ejercicios se nos presentan situaciones donde debemos decidir y las estructuras de control condicionales son útiles.

7. Ejercicios propuestos

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

- Par o impar.
- Alex y la IOI
- Las Notas de Ork
- N - Arithmetic Mean
- A - An easy task I
- A - An easy task II