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

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 **switch**

2. Conocimientos previos

La instrucción **break** se utiliza con la instrucción condicional **switch**.

En una instrucción **switch**, la instrucción **break** hace que el programa ejecute la siguiente instrucción que hay fuera de la instrucción **switch**. Sin una instrucción **break**, se ejecutan todas las instrucciones que hay desde la etiqueta **case** coincidente hasta el final de la instrucción **switch**, incluida la cláusula **default**.

3. Desarrollo

Se trata de una alternativa a la bifurcación if elseif else cuando se compara la misma expresión con distintos valores. Su forma general es la siguiente:

```
switch (expression) {
   case value1: statements1;
   case value2: statements2;
   case value3: statements3;
   case value4: statements4;
   case value5: statements5;
   case value6: statements6;
   default: statements7;
}
```

Las características más relevantes de switch son las siguientes:

1. Cada sentencia **case** se corresponde con un único valor de **expression**. No se pueden establecer rangos o condiciones sino que se debe comparar con valores concretos.
2. Los valores no comprendidos en ninguna sentencia **case** se pueden gestionar en **default**, que es opcional.
3. En ausencia de **break**, cuando se ejecuta una sentencia **case** se ejecutan también todas las **case** que van a continuación, hasta que se llega a un **break** o hasta que se termina el **switch**.

4. Implementación

La impletemación de esta estructura es similar en Java y C++ la única direrencia es que en Java permite que la variable que sea utilizada en switch sea de tipo string algo que no sucede en C++.

4.1. Java

```
//determina cual calificacion se introdujo
switch ( calificacion ) //instruccion switch
{
   case 'A': //calificacion fue A mayuscula
   case 'a': //or a minuscula
      aCuenta++; // incrementa aCuenta
      break; // es necesario salir del switch
   case 'B': // calificacion fue B mayuscula
   case 'b': // o b minuscula
      bCuenta++; // incrementa bCuenta
      break; // sale del switch
   case 'C': //calificacion fue C mayuscula
   case 'c': // o c minuscula
      cCuenta++; // incrementa cCuenta
      break; // sale del switch
   case 'D': //calificacion fue D mayuscula
   case 'd': //o d minuscula
      dCuenta++; // incrementa dCuenta
      break; // sale del switch
   case 'F': // calificacion fue F mayuscula
   case 'f': // o f minuscula
      fCuenta++; // incrementa fCuenta
      break; // sale del switch
   case '\n': // ignora caracteres de nueva linea,
   case '\t': // tabuladores
   case ' ': // y espacios en la entrada
      break; // sale del switch
   default: // atrapa todos los demas caracteres
      cout << "Se introdujo una letra de calificacion incorrecta." << endl;
      break; // opcional; saldra del switch de todas formas
} // fin de switch
```

4.2. C++

```
//determina cual calificacion se introdujo
switch ( calificacion ) //instruccion switch
{
   case 'A': //calificacion fue A mayuscula
   case 'a': //or a minuscula
      aCuenta++; // incrementa aCuenta
```

```
      break; // es necesario salir del switch
   case 'B': // calificacion fue B mayuscula
   case 'b': // o b minuscula
      bCuenta++; // incrementa bCuenta
      break; // sale del switch
   case 'C': //calificacion fue C mayuscula
   case 'c': // o c minuscula
      cCuenta++; // incrementa cCuenta
      break; // sale del switch
   case 'D': //calificacion fue D mayuscula
   case 'd': //o d minuscula
      dCuenta++; // incrementa dCuenta
      break; // sale del switch
   case 'F': // calificacion fue F mayuscula
   case 'f': // o f minuscula
      fCuenta++; // incrementa fCuenta
      break; // sale del switch
   case '\n': // ignora caracteres de nueva linea,
   case '\t': // tabuladores
   case ' ': // y espacios en la entrada
      break; // sale del switch
   default: // atrapa todos los demas caracteres
      cout << "Se introdujo una letra de calificacion incorrecta." << endl;
      break; // opcional; saldra del switch de todas formas
} // fin de switch
```

5. Complejidad

El tiempo de ejecución de la sentencia $CASE \quad C \quad OF \quad v1 : S1|v2 : S2|...|vn : Sn \quad END$; es $T = T(C) + \text{máx}\{T(S_1), T(S_2), ..., T(S_n)\}$. Obsérvese que T(C) incluye el tiempo de comparación con v_1 , v_2 ,..., v_n donde:

- $T(C)$: Complejidad de chequear si el valor de la variable se corresponde con el calor del case.
- $T(S_1)$: Complejidad de las instrucciones que coforman el bloque que se ejecutará en caso que la variable tome el valor del caso 1.
- $T(S_2)$: Complejidad de las instrucciones que coforman el bloque que se ejecutará en caso que la variable tome el valor del caso 2.
- $T(S_n)$: Complejidad de las instrucciones que coforman el bloque que se ejecutará en caso que la variable tome el valor del caso n.

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.

Aunque cabe destacar que la estructura swicth presenta limitaciones frente a la estructura if por lo que su uso es solo para situaciones muy puntuales.

7. Ejercicios propuestos

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

- Par o impar.
- Las Notas de Ork
- A - An easy task I
- A - An easy task II