La instrucción condicional `switch`

J. Sam, J.-C. Chappelier, V. Lepetit

versión 1.0 de octubre de 2013.

En C++, podemos escribir la secuencia de varias condiciones de forma más concisa, en el caso de que estemos probando diferentes valores de una expresión que devuelve un entero .

1 La instrucción `switch`

Así que una secuencia de sentencias `if` como esta :

```
si (i == 1)
  Instrucciones 1
else if (i == 12)
  Instrucciones 2
  ...
else if (i == 36)
  N otras
instrucciones
  Instrucciones N+1
```

puede sustituirse ventajosamente por la siguiente forma de palabras:

```
switch (i) {
caso 1:
  Instrucciones 1;
  interrupción;
cuadro 12:
  Instrucciones 2;
  interrupción;
...
caso 36:
  Instrucciones N;
  break;
por defecto:
  Instrucciones
  N+1; break;
}
```

Cuando se ejecuta una instrucción `switch`, la expresión que sigue a la palabra clave `switch` se evalúa[1] . La ejecución continúa entonces en la `casilla` correspondiente al resultado de la evaluación.

Esta ejecución continúa en secuencia, sin ninguna evaluación de condición, hasta que se encuentra la palabra reservada `break` o se llega al final de la instrucción.

Por lo tanto, la primera parte del código anterior sólo es equivalente, desde el punto de vista de la ejecución, a un `switch` en el que cada caso termina con un `break` (como ocurre en el ejemplo anterior).

Si la evaluación de la expresión no devuelve un valor correspondiente a un caso, el programa ejecuta las instrucciones correspondientes al caso por defecto (precedidas por la palabra clave `default`).

2 Papel de la `ranchera`: ejemplo

Veamos el siguiente ejemplo:

```
switch (a+b) {
  caso 2:
  casilla 8: instrucción2; // cuando (a+b) es 2 u 8
  casilla 4:
  casilla 3: instrucción3; // cuando (a+b) es 2, 3, 4 u 8
    break;
  caso 0: instrucción1; // se ejecuta sólo cuando
    break;              // (a+b) es 0
  por defecto: instrucción4; // en todos los demás
    casos break;
}
```

Supongamos que al evaluar la expresión `(a + b)` se obtiene `8`. La ejecución procederá como sigue:

1. el programa se "conectará" al caso `8` y ejecutará la `instrucción instruction2` ;
2. como no hay instrucción `break` debajo, la ejecución continúa y maneja el caso 4 (sin probar ni hacer nada, ya que no hay instrucciones para ejecutar);
3. entonces se encuentra `la instrucción3` y se ejecuta;
4. Finalmente, encontramos la `ruptura` debajo de `la instrucción3` y podemos salir de la instrucción `switch`.

Esto demuestra que

– `la instrucción2` se ejecutará si `(a + b)` es 2 u 8 ;
– `La instrucción3 se` ejecutará si `(a + b)` es 2, 3, 4 u 8 ;
– `la instrucción1` sólo se ejecutará si `(a + b)` es 0.

1. en el ejemplo anterior, la expresión se reduce a la variable simple `i`

El hecho de no poner una instrucción delante de un caso simplemente permite implementar un "*o lógico*" entre varias condiciones: `la instrucción2` se ejecuta si `(a + b)` es 2 **u** 8, por ejemplo. Esto evita la repetición innecesaria de código. Pero probablemente sea mejor evitar este tipo de optimización en primera instancia, ya que no es fácil de entender en una primera lectura...

3 `switch` vs `if..else`

`switch` es menos general que `if..else` :

- el valor que se comprueba debe ser de tipo integral: `char`, `int`, `unsigned int`, etc.).
- la expresión comprobada es siempre la misma (la "`i`" o "`a+b`" en los ejemplos anteriores);
- los casos deben ser constantes (no variables).