**Another selection statement: switch.**

The syntax of the *switch* statement is a bit peculiar. Its purpose is to check for a value among a number of possible constant expressions. It is something similar to concatenating if-else statements, but limited to constant expressions. Its most typical syntax is:

*switch (expression)*

*{*

*case constant1:*

*group-of-statements-1;*

*break;*

*case constant2:*

*group-of-statements-2;*

*break;*

*.*

*.*

*.*

*default:*

*default-group-of-statements*

*}*

It works in the following way: *switch* evaluates expression and checks if it is equivalent to *constant1*; if it is, it executes *group-of-statements-1* until it finds the *break* statement. When it finds this *break* statement, the program jumps to the end of the entire *switch* statement (the closing brace).

If expression was not equal to *constant1*, it is then checked against *constant2*. If it is equal to this, it executes *group-of-statements-2* until a *break* is found, when it jumps to the end of the *switch*.

Finally, if the value of expression did not match any of the previously specified constants (there may be any number of these), the program executes the statements included after the *default:* label, if it exists (since it is optional).

Both of the following code fragments have the same behavior, demonstrating the *if-else* equivalent of a *switch* statement:

|  |  |
| --- | --- |
| **switch example** | **if-else equivalent** |
| switch (x) {  case 1:  cout << "x is 1";  break;  case 2:  cout << "x is 2";  break;  default:  cout << "value of x unknown";  } | if (x == 1) {  cout << "x is 1";  }  else if (x == 2) {  cout << "x is 2";  }  else {  cout << "value of x unknown";  } |

The *switch* statement has a somewhat peculiar syntax inherited from the early times of the first C compilers, because it uses labels instead of blocks. In the most typical use (shown above), this means that *break* statements are needed after each group of statements for a particular *label*. If *break* is not included, all statements following the *case* (including those under any other labels) are also executed, until the end of the *switch* *block* or a jump statement (such as *break*) is reached.

If the example above lacked the *break* statement after the first group for *case* *one*, the program would not jump automatically to the end of the *switch* block after printing *x* is 1, and would instead continue executing the statements in *case* *two* (thus printing also *x* is 2). It would then continue doing so until a *break* statement is encountered, or the end of the switch block. This makes unnecessary to enclose the statements for each *case* in braces {}, and can also be useful to execute the same group of statements for different possible values. For example:

|  |  |  |
| --- | --- | --- |
| 1 2 3 4 5 6 7 8 9 | switch (x) {  case 1:  case 2:  case 3:  cout << "x is 1, 2 or 3";  break;  default:  cout << "x is not 1, 2 nor 3";  } |  |

Notice that switch is limited to compare its evaluated expression against labels that are constant expressions. It is not possible to use variables as labels or ranges, because they are not valid C++ constant expressions.

To check for ranges or values that are not constant, it is better to use concatenations of *if* and *else if* statements.