## The Structure of `switch` statements

The `switch` statement is similar to an `if-else ladder`. It is used to choose from among many courses of action depending on the value of a particular variable (which must be an integer or a character). It cannot be used for inequalities, only when there is a finite set of possible values. The syntax is as follows:

```
switch(switchvariable)
{
  case firstvalue:  statement(s);
                    break;
  case secondvalue: statement(s);
                    break;
  [more cases optional]
  default:          statement(s);    //a default case is optional
}
```

Because this syntax may appear rather opaque, we present the following example:

```
int family_size;
bool allsame = false;
cout << "Enter the number of children in your family: ";
cin >> family_size;

switch (family_size)
{
  case 1:   cout << "You are an only child." << endl;
            break;
  case 2:
  case 3:   cout << "You have an average-sized family." << endl;
            break;
  case 4:   if(allsame)
              cout << "You have a team, not a family!" << endl;
            else
              cout << "You have a largish family!" << endl;
            break;
  case 0:   cout << "How are you answering this?" << endl;
            break;
  default:  cout << "You have a giant-sized family!!" << endl;
}
```

The `default` clause is optional. **The `break` statement is important**—without it, the appropriate statements ***and*** all statements that follow it are executed. In the example above, if there were no `breaks`, the "only child" would see all four messages. All code written after the case is listed will be executed up to the break, so loops, if_else etc. may be used. This also means that the same code will be used for case 2 and case 3 because there isn't a break until after the case 3 code. This is very useful if some cases would use the same code.

Below is another example using a different kind of variable.

```
char letter;
cout << "Enter a letter: ";
cin >> letter;

switch (letter)
{
  case 'A':
  case 'E':
  case 'I':
  case 'O':
  case 'U': cout << "Vowel." << endl;
            break;
  case 'Y': cout << "Sometimes a vowel, usually not." << endl;
            break;
  default:  cout << "Consonant." << endl;
}
```

Notice that the `switch` case again uses the stacking for code that can be used by other cases. Also, the cases do not need to be integers nor do they need to be in alphabetical or numerical order. The `default` is used to avoid the other 20 cases that are consonants. A failing of this particular piece of code is that there is no error checking for non-alphabet inputs so they will all count as consonants too.

Again, a `switch` case and an `if-else` ladder are very similar and can often be used interchangeably. For comparison, here is an example of an `if-else` ladder and a `switch` case that do the same thing.

If –Else Ladder

```
if (num == 6)
   cout << "Half dozen" << endl;

else if (num == 12)
   cout << "One dozen" << endl;

else if (num == 24)
   cout << "Two dozen" << endl;

else if (num == 13)
   cout << "Baker's dozen" << endl;

else
   cout << "Not bakery numbers" << endl;
```

Switch-Case

```
switch (num)
{
  case 6:  cout << "Half dozen" << endl;
           break;

  case 12: cout << "One dozen" << endl;
           break;

  case 24: cout << "Two dozen" << endl;
           break;

  case 13: cout << "Baker's dozen" << endl;
           break;

  default: cout << "Not bakery numbers" <<
endl;
}
```