User Input and (more) Control Structures

Using a while loop to repeatedly check cin for an input allows us to create programs with actual user functionality. The easiest way to do this is to use cin with an integer to allow the user to select from a series of options, but strings can also be used (as long as you check for exact spelling and capitalization). Here's an example with the integer variable:

```
cout << "Enter an option from 1 to 3, 0 to exit" << endl;
int input = 0;
while (cin >> input){
      if(input == 0){return 1;}
      else if(input == 1) {//do thing}
      else if(input == 2) {//do thing}
      else if(input == 3) {//do thing}
      else{cout << "You have not selected an option, try again!"}
}
```

When working with while loops there are extra features that we can use to make our loops *fancier*. With the `continue` and `break` options, you can make better controls.

`continue;` End the current while loop and go back to the beginning.

`break;` End the current while loop and leave it completely without checking it again.

I won't include examples, but try putting these in while loops and see what happens. Feel free to use the example menu code above to test, as it will prevent you from getting infinite loops.