Break & Continue

This lesson will introduce you to the break and continue keywords and their usage.

We'll cover the following

- Break
- Continue

Break

There is a special statement in C called `break` that allows you to exit from a loop. It overrides your loop condition at the top or bottom and “breaks” the loop when you want. As we saw in switch statements, we also use `break` to exit from each case.

```c
#include <stdio.h>

int main() {
  // Check whether 80 is in the first 20 multiples of 5
  int mult = 0;
  for(int i = 0; i <= 20; i++){
    mult = 5 * i;
    if( mult == 80 ){
      printf("80 is in the first 20 multiples of 5!\n5 x %d = %d\n", i, mult);
      break;
      printf("The loop does not reach this point\n");
    }
  }
  printf("Loop has ended");
}
```

As you can see in the code above, even though our loop’s ending condition hasn’t been fulfilled, the break statement does not allow it to move forward. The `printf` statement in line 10 isn’t executed either. Instead, the program jumps directly to line 14.

Continue

The `continue` statement is not used very often. It causes the next iteration of a loop to begin without executing the code still left in the current iteration. You can think of it as a sort of instruction to “do nothing, just continue”.

```
#include <stdio.h>

int main() {
  for( int i = 0; i < 11; i++ ){
    if( i == 5 ){
      continue;
    }
    printf("%d\n", i);
  }
}
```

Check out the output from the code we just wrote above. Does anything look different? The `continue` statement in the iteration when `i == 5` allowed us to skip the `printf` command.

This brings us to the end of our discussion on control flow. Time to try out some challenges. Good luck!

The next section will be all about one of the most fundamental aspects of C programming: Functions.