C# Loops

What is a loop?

- A developer will use a loop when it is necessary to execute a block of code over and over again.
- Loops will continue to execute while a condition holds true
- Loops have their own scope
- There are different kinds of loops
  - While different loops are all interchangeable, certain loops are often better for specific jobs

For Loop

- for loops are ideal in situations where you know how many times you would like the loop to run

```
for(start; condition; iteration) {
    // do stuff
}
```

- start – Initial value for the iterator (loop counter)
- condition – Tested each iteration, loop completes when condition becomes false
- Iteration – Executed each iteration, the pattern by which the iterator changes

What a for loop looks like:

```
for (int index = 0; index < 10; index++)
{
    Console.WriteLine(index);
}
```

//Output: 0 1 2 3 4 5 6 7 8 9

While Loop

- A while loop allows code to be executed repeatedly based on a given Boolean condition.
- A while loop always checks the condition before entering the loop.
- This means that it is possible for a while loop to never run if the condition is initially false

What a while loop looks like:

```
int numberOfplayers = 12;
while(numberOfplayers < 16)
{
    numberOfplayers++;
    Console.WriteLine(numberOfplayers);
}
```

//Output:
13
14
15
16

Do-While Loop

- Similar to a while loop
- The difference is it will test the condition at the end of the loop body, instead of testing the condition before executing the loop body.
- Practical examples:
  - Command line interpreter
  - Read data from a file
  - Shipping applications

What a do-while loop looks like:

```
double input;
do
{
    Console.Write("Please enter positive number:\n");
    input = double.Parse(Console.ReadLine());
}
while (input < 0);
```

Nested Loops

- It is possible to use one or more loops inside each other.

What a nested loop looks like:

```
for (int i = 0; i < 5; i++)
{
    for (int j = 0; j < 5; j++)
    {
        if (i == j)
        {
            Console.WriteLine("We have a match");
        }
        else
        {
            Console.WriteLine(i + "," + j);
        }

    }
    Console.WriteLine("-------------");
}
```

OUTPUT:

```
We have a match
0,1
0,2
0,3
0,4
-------------
1,0
We have a match
1,2
1,3
1,4
-------------
2,0
2,1
We have a match
2,3
2,4
-------------
3,0
3,1
3,2
We have a match
3,4
-------------
4,0
4,1
4,2
4,3
We have a match
-------------
```

Continue

- Instead of breaking out of a while loop or for loop, the continue key word will skip a part of the loop and continue on with the next iteration of the loop.
- Essentially, the continue keyword will skip the current iteration in the loop.

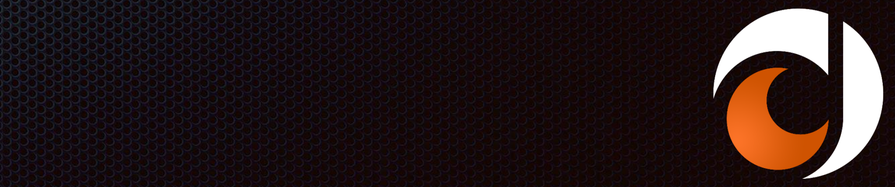
What the continue keyword looks like:

```
for(int i = 0; i < 10; i++)
{
    if (i== 2)
    {
        continue;
    }

    Console.WriteLine(i);
}
```

Output:

```
0
1
3
4
5
6
7
8
9
```



Review Questions

- What is a while loop? Give an example.
- What is a for loop? Give an example.
- Why would you use a while loop instead of a for loop?
- When is it appropriate to use a do…while loop?
- When is it appropriate to use the continue keyword?

Assignment

Fizzbuzz

Write a program that prints every number from 0 to 100 to the console

If a number is divisible by 3, print ‘fizz’ instead of the number

If a number is divisible by 5, print ‘buzz’ instead of the number

If a number is divisible by 3 AND 5, print ‘fizzbuzz’ instead of the number