

CSCI 1301 – Lab 27

November 3, 2019

1 From while to for

Rewrite the following while (or do...while) loops as for loops.

```
1  int a = 0;
2  while (a != 10)
3  {
4      Console.WriteLine(a);
5      a++;
6  }

1  int b = 3;
2  while (b >= -2)
3  {
4      Console.WriteLine(b);
5      b -= 2;
6  }

1  int c = 10;
2  while(c <= 100) {
3      Console.WriteLine(c);
4      c += 10;
5  }

1  int d = 1;
2  do
3  {
4      Console.WriteLine(d);
5      d *= 2;
6  } while (d <= 100);
```

2 From for to while

Rewrite the following for loops as while loops:

```
for (int e = 10; e <= 100; e += 10) Console.Write(e + " ");
for (double f = 150; f > 2; f/=2 ) Console.Write(f + " ");
for (char g = 'A' ; g != 'a'; g = (char)((int)g +1) ) Console.Write(g + " ");
for (int h = 0; h > -30; h -= 1) Console.Write(h + " ");
```

3 Pushing Further (Optional)

This lab's pushing further is about two modifications of `for` loops that are sometimes considered as bad design: used poorly, they can make the code harder to read, to debug, and sometimes makes it hard to follow the flow of control of your program. They are introduced because you may see them in your future, but, except for rare cases, should be avoided completely.

3.1 Multiple Initializations and Updates

The exact structure of `for` loops is actually more complex than what we discussed in class. It is

```
for(<initializations>; <condition>; <updates>)
{
    <statement block>
}
```

That is, there can be more than one initialization (but only if the variables all have the same datatype) and more than one update. That is, are legal statements like:

```
for(int z = 0, y = 10; z < y ; z++){ Console.WriteLine($"{z} + {y} =
    ↪ {z+y}"); }
```

or

```
for (int x = 0, y = 12 ; x != y; x++, y--)
    Console.WriteLine($"The difference between {x} and {y} is {x - y}");
```

Also, the initialization, as well as the update condition, are actually optional: we could have

```
int w = 0;
for (; w < 5; w++) { Console.WriteLine(w); }
```

and

```
for(int r = 10; r > 0;) { Console.WriteLine(r--); }
```

Try to rewrite the four `for` loops just given as four `for` loops with exactly one initialization and one update in the header of the `for` loop.

3.2 continue and break

Programmers can use two keywords in loops, `continue` and `break`, that modify the control flow. Looking at the following code, try to understand what those statements do.

```
for (int i = 1; i <= 5; i++)
{
    if (i == 3) continue;
    Console.Write(i + " ");
}
```

```
for (int i = 1; i <= 5; i++)
{
    if (i == 3) break;
    Console.Write(i + " ");
}
```

You can also use break and continue in while loops. Try to rewrite the previous two for as while loops: there is a trick to make the while loop using break works properly, can you spot it?

3.3 Default values

Execute the following:

```
int[] ar = new int[5];
ar[0] = 5;
for (int i = 0; i < ar.Length; i++)
    Console.WriteLine(ar[i]);
```

What can you conclude about the value of the cells that were not assigned?