# For Loops

https://csci-1301.github.io/about # authors

October 18, 2021 (01:53:41 PM)

### **Contents**

Э	Pushing Further (Optional) 5.1 Multiple Initializations and Updates	3
	Using continue and break	3
3	Implementing for Loops	2
2	From for to while	2
1	From while to for	1

#### 1 From while to for

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

```
int a = 0;
while (a != 10)
    Console.WriteLine(a);
    a++;
}
int b = 3;
while (b \ge -2)
    Console.WriteLine(b);
    b = 2;
}
int c = 10;
while(c <= 100) {
    Console.WriteLine(c);
    c += 10;
}
int d = 1;
do
{
    Console.WriteLine(d);
```

```
d *= 2;} 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 (int h = 0; h > -30; h -= 1)
{
    Console.Write(h + " ");
}
```

# 3 Implementing for Loops

This exercise is to practice for loops.

Write a program that asks the user to enter a positive integer, and then uses a for loop to compute the sum of all the integers between 1 and the integer given by the user. For instance, if the user enters 5, your program should display 15 on the screen (i.e., 1 + 2 + 3 + 4 + 5 = 15).

Then, answer the following questions:

- 1. Without running your program, can you tell what will happen if the user enters a negative value?
- 2. Do you think you could have written the same program using a while loop?
- 3. How would you change the program to make it compute the product instead of the sum (i.e., for 5,  $1 \times 2 \times 3 \times 4 \times 5 = 120$ )?
- 4. How would you change the program to make it display on the screen the divisors of the integer entered? Examples:
  - divisors of 5 are: 1, 5
  - divisors of 10 are: 1, 2, 5, 10?

You can modify your program to check your answers to the previous questions. Once you are done, modify your original program in these two respects:

- 1. Once the result of the computation is displayed on the screen, ask the user if (s)he wants to compute the sum using another integer or quit, and act accordingly.
- 2. Add some input validation: floating-point values, non-numeric strings and negative values should not be allowed (i.e., your program should ask for another value).

# 4 Using continue and break

Programmers can use two keywords in loops, continue and break, that modify the control flow. They can make the loop more confusing to read, but can sometimes be useful for reducing the number of nested if statements in a complex loop. Try executing the following code to see what these 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 + " ");
}</pre>
```

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

# 5 Pushing Further (Optional)

#### 5.1 Multiple Initializations and Updates

This section is about two modifications of for loops that are sometimes considered bad design: used poorly, they can make the code harder to read and to debug, and sometimes make 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 in your own code. The exact structure of for loops is actually more complex than discussed in class. It is

That is, there can be more than one initialization (but only if the variables all have the same datatype) and more than one update. This means there 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}");</pre>
```

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);
}</pre>
```

and

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

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