CSCI 1301 - Lab 19

Clément Aubert

March 20, 2018

Part I - 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 >= -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);
```

Part II - 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 + " ");
```

Part III - A Simple User-Controlled Loop

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 at the screen (i.e., $1 + 2 + 3 + 4 + 5 = 15$).

Answer the following questions:

a. Whitout running your program, can you tell what will happen if the user enter a negative value?
b. Do you think you could have written the same program using a `while` loop?
c. How could you change your program so that it would compute the product instead of the sum (i.e., for 5, $1 \times 2 \times 3 \times 4 \times 5 = 120$)?
d. How could you change your program so that it would display on the screen the divisors of the integer entered (i.e., for 5, only 1 and 5).

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

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

Part IV – Pushing Further

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 harder 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.

a. 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--); }
```

Can you rewrite them four as `for` loops with exactly one initialization and one update?

b. 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?