Char and Int Conversion, Ordering of Characters

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

June 24, 2021 (10:27:06 AM)

Contents

0.1 Reading and Understanding

Characters are represented by integers: cf. https://en.wikipedia.org/wiki/ASCII#Printable_characters for a mapping between the glyphs (i.e., space, !, etc.) and **dec**imal values (to be read as "integer code", i.e., 32, 33, 34, etc.). Note that the characters are divided in groups, and that there are 95 printable characters.

0.2 Converting

Copy the following snippet of code in a `Main` method:

```
int intVar = (int)'C';
char charVar = (char)84;
Console.WriteLine($"'C' is represented as {intVar}\n"
    + $"{charVar} corresponds to the value 84");
```

And note that we can explicitely convert `int` into `char`, and `char` into `int`.

Actually, the conversion from `char` to `int` could be done implicitly by C#: replace the previous first line with

```
int intVar = 'C';
```

And note that your program would still compile. Can you also convert implicitly `int` into `char`?

0.3 Comparing

Exactly as 65 is less than 97, the character associated to 65, `A`, is less than the character associated with 97, `a`. You can convince yourself by executing the following code:

```
if ('A' > 'a')
    Console.Write("A is greater than a");
else
    Console.Write("A is less than a");
```

0.4 Testing for Equality

Note that you can also test if a character is equal to an other by using ==, as for integer values. This is particularly useful when we want to ask the user for a "yes" / "no" decision.

Write a snippet of code that

- Ask the user for a character,
- Display on the screen "The user said yes" if the user entered "Y" or "y",
- Display on the screen "The user said no" if the user entered "N" or "n",
- Display on the screen "The user entered an incorrect value" if the user entered any other character.

To read *a single character* (instead of a whole string), use

```
Console.WriteLine("Press y or Y for Yes, n or N for No:");
char answer = Console.ReadKey().KeyChar;
```

1 Pushing Further (Optional)

This lab's pushing further suggests to take some advance in two topics we will be covering soon: `for` loops and `string` comparison

1.1 `char` and `for` loop

Try to understand what the following code does:

```
for (int i = 32; i <= 126; i++)
    Console.Write((char)i);
```

Compile it, execute it, understand what its purpose is, and what its structure is.

1.2 String Comparison

Comparing strings cannot be done with > and < operators. To compare them, we have to use the CompareOrdinal[1] method of the String[2] class. It works as follow:

[1] https://docs.microsoft.com/en-us/dotnet/api/system.string.compareordinal?view=netframework-4.7.1
[2] https://msdn.microsoft.com/en-us/library/system.string(v=vs.110).aspx

```
if (String.CompareOrdinal("A", "a") > 0)
{
    Console.Write("A is greater than a");
}
else
{
    Console.Write("A is less than a");
}
```

Note that `CompareOrdinal` returns an integer, that we then compare with 0.

- If the value returned is 0, then the strings are the same,
- If the value returned is less than 0, then the first string is less than the second one,
- If the value returned is greater than 0, then the first string is greater than the second one.

In the previous example, we tested string made of only one character, but we can compare arbitrarily complex strings:

```
if (String.CompareOrdinal("Augusta", "Auguste") > 0)
{
    Console.Write("Augusta is greater than Auguste");
}
else
{
    Console.Write("Augsta is less than Auguste");
}
```

To conclude with this topic, note that the integer returned actually has a precise value: examine the following code to understand it.

```
if (String.CompareOrdinal("A", "a") == ((int)'A' - (int)'a'))
    Console.WriteLine("Ok, I get it now");

if (String.CompareOrdinal("Ab", "az") == (((int)'A' + (int)'b') - ((int)'a' + (int)'z')))
    Console.WriteLine("Yes, I really do.");
else if (String.CompareOrdinal("Ab", "az") == ((int)'A' - (int)'a'))
    Console.WriteLine("Or do I?");

if (String.CompareOrdinal("ABCDEf", "ABCDEF") == (int)'f' - (int)'F')
    Console.WriteLine("Ok, now I'm good.");
```

Do you understand how the returning value is computed for these strings?