Char and Int Conversion, Ordering of Characters

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

June 30, 2021 (03:52:38 PM)

Table of Contents

## 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, A, !, etc.) and **dec**imal values, to be read as “integer code”, i.e., 32, 33, 34, etc.

In the references table, each character’s integer code is given for different [numeral systems](https://en.wikipedia.org/wiki/Radix#In_numeral_systems):

* Binary: base 2
* Oct: octal, base 8
* Dec: decimal, base 10
* Hex: hexadecimal, base 16

Note that the characters are divided in groups, and that there are 95 printable characters.

## Converting

Copy the following snippet of code in a Main method:

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

And note that we can explicitly 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 still compiles.

Can you also convert implicitly int into char?

Next write code to determine the int values for the following characters:

| char value | int value |
| --- | --- |
| w | 119 |
| A |  |
| 5 |  |
| # |  |
| p |  |

Also determine what characters the following integers (in decimal base) represent: 49, 104, 89.

| int value | char value |
| --- | --- |
| 59 |  |
| 112 |  |
| 89 |  |

## Comparing

Exactly as is less than , the character associated with , A, is less than the character associated with , 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");

Implement the following short program:

Ask user to enter a lowercase character.

1. First check that the alphabet is within a - z range
2. if it is not, display “not a lowercase character”
3. if it is, perform the following steps:
   * if user enters letter n, display “You entered n”
   * if the character occurs before n in the alphabet, display “Before n”
   * if the character occurs after n in the alphabet, display “After n”

* To read *a single character* (instead of a whole string), use ReadKey() method: Console.ReadKey().KeyChar

## 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.

# Pushing Further (Optional)

## String Comparison

Comparing strings cannot be done with > and < operators. To compare them, we have to use the [CompareOrdinal](https://docs.microsoft.com/en-us/dotnet/api/system.string.compareordinal) method of the [String](https://docs.microsoft.com/en-us/dotnet/api/system.string) class.

It works as follow:

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 .

* If the value returned is , then the strings are the same,
* If the value returned is less than , then the first string is less than the second one,
* If the value returned is greater than , 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", "August") > 0)  
{  
 Console.Write("Augusta is greater than August");  
}  
else  
{  
 Console.Write("Augusta is less than August");  
}

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?