Char and Int Conversion, Ordering of Characters

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

January 11, 2023 (12:00:51 PM)

Table of Contents

This lab serves multiple goals:

* To introduce you to the char datatype,
* To introduce you to the different representations of characters,
* To exemplify how to convert between representations of characters,
* To introduce the order on characters,
* (Optional) To illustrate the comparison of strings.

# Warm Up

Characters are represented by integers: you can read [on wikipedia](https://en.wikipedia.org/wiki/ASCII#Printable_characters) a mapping between the glyphs (i.e., space, A, !, etc.) and decimal values, to be read as “integer code”, i.e., 32, 33, 34, etc.

In the referenced table [on wikipedia](https://en.wikipedia.org/wiki/ASCII#Printable_characters), 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

*Decimal system* is what we use everyday, but computer programs occasionally use other numerical systems. For that system, it gives (no need to memorize this information, this is simply for your general awareness):

| Decimal representation | Glyph (character) |
| --- | --- |
| 32 | space |
| 33 | ! |
| 34 | ” |
| 35 | # |
| 36 | $ |
| 37 | % |
| 38 | & |
| 39 | ’ |
| 40 | ( |
| 41 | ) |
| 42 | \* |
| 43 | + |
| 44 | , |
| 45 | - |
| 46 | . |
| 47 | / |
| 48 | 0 |
| 49 | 1 |
| 50 | 2 |
| 51 | 3 |
| 52 | 4 |
| 53 | 5 |
| 54 | 6 |
| 55 | 7 |
| 56 | 8 |
| 57 | 9 |
| 58 | : |
| 59 | ; |
| 60 | < |
| 61 | = |
| 62 | > |
| 63 | ? |
| 64 | @ |
| 65 | A |
| 66 | B |
| 67 | C |
| 68 | D |
| 69 | E |
| 70 | F |
| 71 | G |
| 72 | H |
| 73 | I |
| 74 | J |
| 75 | K |
| 76 | L |
| 77 | M |
| 78 | N |
| 79 | O |
| 80 | P |
| 81 | Q |
| 82 | R |
| 83 | S |
| 84 | T |
| 85 | U |
| 86 | V |
| 87 | W |
| 88 | X |
| 89 | Y |
| 90 | Z |
| 91 | [ |
| 92 | \ |
| 93 | ] |
| 94 | ^ |
| 95 | \_ |
| 96 | ` |
| 97 | a |
| 98 | b |
| 99 | c |
| 100 | d |
| 101 | e |
| 102 | f |
| 103 | g |
| 104 | h |
| 105 | i |
| 106 | j |
| 107 | k |
| 108 | l |
| 109 | m |
| 110 | n |
| 111 | o |
| 112 | p |
| 113 | q |
| 114 | r |
| 115 | s |
| 116 | t |
| 117 | u |
| 118 | v |
| 119 | w |
| 120 | x |
| 121 | y |
| 122 | z |
| 123 | { |
| 124 | | |
| 125 | } |
| 126 | ~ |

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

# Converting Between Characters Representations

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 |  |
| # |  |

Also determine what characters the following integers (in decimal system) represent:

| int value | char value |
| --- | --- |
| 49 |  |
| 104 |  |
| 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 to practice this concept. Note that to read *a single character* (instead of a whole string), use the ReadKey() method: Console.ReadKey().KeyChar will returns a char that you can then store into a variable and manipulate.

1. Ask user to enter a lowercase character,
2. Check that the character is within the **a - z** range (it *is* a lowercase character),
3. When it is not in this range, display “not a lowercase character”,
4. Otherwise, perform the following steps:
   * if user enters character 'n', display “You entered n”
   * if the character occurs before 'n', display “Before n”
   * if the character occurs after 'n', display “After n”

# Testing for Equality

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 program that

* Asks the user for a character,
* Displays on the screen “The user said yes” if the user entered 'Y' or 'y',
* Displays on the screen “The user said no” if the user entered 'N' or 'n',
* Displays 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?