Contents

Char Conversion and Ordering **1**
Warm Up 1
Converting Between Characters Representations 4
Testing for Equality 5
Comparing 6
Pushing Further (Optional) 7
String Comparison 7

Char Conversion and Ordering

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 of characters,
- (Optional) To illustrate the comparison of strings.

Warm Up

Characters are represented by integers: you can read on wikipedia[1] a mapping between the glyphs (e.g., space, `A`, `!`, etc.) and decimal values, to be read as "integer code" (e.g., 32, 33, 34, etc.).

In the referenced table on wikipedia[2], each character's integer code is given for different numeral systems[3]:

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

The *decimal system* is what we use every day, 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):

[1] https://www.wikiwand.com/en/ASCII#Printable_characters
[2] https://www.wikiwand.com/en/ASCII#Printable_characters
[3] https://www.wikiwand.com/en/Radix#In_numeral_systems

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

Decimal representation	Glyph (character)
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

Decimal representation	Glyph (character)
120	x
121	y
122	z
123	{
124	|
125	}
126	~

Note that the characters are divided into 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`, but 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 the decimal system) represent:

`int` value	`char` value
49	
104	
89	

Solution:

Your code could look like the following:

```
Console.WriteLine("int value | char value\n" +
  " | \n" +
  (int)'w' + "       | w\n" +
  (int)'A' + "        | A\n" +
  (int)'5' + "        | 5\n" +
  (int)'#' + "        | #\n" +
  " | \n" +
  "49        | " + (char)49 + "\n" +
  "104       | " + (char)104 + "\n" +
  "89        | " + (char)89 + "\n"
);
```

Testing for Equality

You can test if a character is equal to another 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. To read *a single character* (instead of a whole string), use the `ReadKey()` method: `Console.ReadKey().KeyChar` will return a `char` that you can then store into a variable and manipulate.
- 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.

Solution:

You can get started with this short program:

```
Console.WriteLine("Enter a character:");
// We ask the user for a character.
char input = Console.ReadKey().KeyChar;
// We read from the user.
Console.WriteLine();
// We are introducing a new line after the user input
if (input == 'Y')
{
    // The input is the letter Y, uppercase.
    Console.WriteLine("You entered 'Y'.");
}
```

It does not fit the description, though, as many elements are missing. Implement them all!

Comparing

Exactly as 65 is less than 97, the character associated with 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'.");
}
```

that displays " 'A' is less than 'a'.".

Implement the following short program to practice this concept.

1. Ask the 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 "The character 'X' is not a lowercase character", where X is replaced by the character they entered,
4. Otherwise, perform the following steps:
    - if the user enters character `'n'`, display "You entered 'n'."
    - if the character occurs before `'n'`, display "The character you entered is a lowercase letter before 'n'."
    - if the character occurs after `'n'`, display "The character you entered is a lowercase letter after 'n'."

Solution:

The following short program fits the description:

```
 Console.WriteLine("Enter a character:");
// We ask the user for a character.
char input = Console.ReadKey().KeyChar;
// We read from the user.
Console.WriteLine();
// We are introducing a new line after the user input
```

```
if (input >= 'a' && input <= 'z')
{
   // The input is a lowercase letter.
  if (input == 'n') { Console.WriteLine("You entered 'n'."); }
  else if (input < 'n') { Console.WriteLine("The character you entered is a lower ca
  else { Console.WriteLine("The character you entered is a lower case letter after '
}
else
{
   // The input is not a lowercase letter.
  Console.WriteLine("The character '" + input + "' is not a lowercase character.");
}
```

Pushing Further (Optional)

String Comparison

Comparing strings cannot be done with > and < operators (we can use ==, however). To compare them, we have to use the CompareOrdinal[4] method of the String[5] class.

It works as follows:

```
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 a 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\".");
```

[4]https://docs.microsoft.com/en-us/dotnet/api/system.string.compareordinal
[5]https://docs.microsoft.com/en-us/dotnet/api/system.string

```
} 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?