Lecture 8: Dealing with characters in C++

ASCII Code

ASCII printable characters (character code 32-127)

Codes 32-127 are common for all the different variations of the ASCII table, they are called printable characters, represent letters, digits, punctuation marks, and a few miscellaneous symbols. You will find almost every character on your keyboard. Character 127 represents the command DEL.

DEC	OCT	HEX	BIN	Symbol	Description
32	040	20	00100000		Space
33	041	21	00100001	!	Exclamation mark
34	042	22	00100010	"	Double quotes (or speech marks)
35	043	23	00100011	#	Number
36	044	24	00100100	$	Dollar
37	045	25	00100101	%	Per cent sign
38	046	26	00100110	&	Ampersand
39	047	27	00100111	'	Single quote
40	050	28	00101000	(	Open parenthesis (or open bracket)
41	051	29	00101001	)	Close parenthesis (or close bracket)
42	052	2A	00101010	*	Asterisk
43	053	2B	00101011	+	Plus
44	054	2C	00101100	,	Comma
45	055	2D	00101101	-	Hyphen
46	056	2E	00101110	.	Period, dot or full stop
47	057	2F	00101111	/	Slash or divide
48	060	30	00110000	0	Zero

DEC	OCT	HEX	BIN	Symbol	Description
49	061	31	00110001	1	One
50	062	32	00110010	2	Two
51	063	33	00110011	3	Three
52	064	34	00110100	4	Four
53	065	35	00110101	5	Five
54	066	36	00110110	6	Six
55	067	37	00110111	7	Seven
56	070	38	00111000	8	Eight
57	071	39	00111001	9	Nine
58	072	3A	00111010	:	Colon
59	073	3B	00111011	;	Semicolon
60	074	3C	00111100	<	Less than (or open angled bracket)
61	075	3D	00111101	=	Equals
62	076	3E	00111110	>	Greater than (or close angled bracket)
63	077	3F	00111111	?	Question mark
64	100	40	01000000	@	At symbol
65	101	41	01000001	A	Uppercase A
66	102	42	01000010	B	Uppercase B
67	103	43	01000011	C	Uppercase C
68	104	44	01000100	D	Uppercase D
69	105	45	01000101	E	Uppercase E
70	106	46	01000110	F	Uppercase F
71	107	47	01000111	G	Uppercase G
72	110	48	01001000	H	Uppercase H
73	111	49	01001001	I	Uppercase I
74	112	4A	01001010	J	Uppercase J
75	113	4B	01001011	K	Uppercase K
76	114	4C	01001100	L	Uppercase L
77	115	4D	01001101	M	Uppercase M
78	116	4E	01001110	N	Uppercase N
79	117	4F	01001111	O	Uppercase O
80	120	50	01010000	P	Uppercase P
81	121	51	01010001	Q	Uppercase Q
82	122	52	01010010	R	Uppercase R
83	123	53	01010011	S	Uppercase S
84	124	54	01010100	T	Uppercase T
85	125	55	01010101	U	Uppercase U
86	126	56	01010110	V	Uppercase V
87	127	57	01010111	W	Uppercase W
88	130	58	01011000	X	Uppercase X
89	131	59	01011001	Y	Uppercase Y
90	132	5A	01011010	Z	Uppercase Z
91	133	5B	01011011	[	Opening bracket
92	134	5C	01011100	\	Backslash

DEC	OCT	HEX	BIN	Symbol	Description
93	135	5D	01011101	]	Closing bracket
94	136	5E	01011110	^	Caret - circumflex
95	137	5F	01011111	_	Underscore
96	140	60	01100000	`	Grave accent
97	141	61	01100001	a	Lowercase a
98	142	62	01100010	b	Lowercase b
99	143	63	01100011	c	Lowercase c
100	144	64	01100100	d	Lowercase d
101	145	65	01100101	e	Lowercase e
102	146	66	01100110	f	Lowercase f
103	147	67	01100111	g	Lowercase g
104	150	68	01101000	h	Lowercase h
105	151	69	01101001	i	Lowercase i
106	152	6A	01101010	j	Lowercase j
107	153	6B	01101011	k	Lowercase k
108	154	6C	01101100	l	Lowercase l
109	155	6D	01101101	m	Lowercase m
110	156	6E	01101110	n	Lowercase n
111	157	6F	01101111	o	Lowercase o
112	160	70	01110000	p	Lowercase p
113	161	71	01110001	q	Lowercase q
114	162	72	01110010	r	Lowercase r
115	163	73	01110011	s	Lowercase s
116	164	74	01110100	t	Lowercase t
117	165	75	01110101	u	Lowercase u
118	166	76	01110110	v	Lowercase v
119	167	77	01110111	w	Lowercase w
120	170	78	01111000	x	Lowercase x
121	171	79	01111001	y	Lowercase y
122	172	7A	01111010	z	Lowercase z
123	173	7B	01111011	{	Opening brace
124	174	7C	01111100	|	Vertical bar
125	175	7D	01111101	}	Closing brace
126	176	7E	01111110	~	Equivalency sign - tilde
127	177	7F	01111111		Delete

Printing ASCII Value

As stated above, each character is interpreted as ASCII character. It's possible for you to get the ASCII value of any character. You simply pass the character to the int() function. This process is called type casting. Let's demonstrate this:

Example:

```
#include <iostream>
using namespace std;
int main()
{
   char ch;
   cout << "Enter any character: ";
   cin >> ch;
   cout << "The ASCII Value of " << ch << " is " << int(ch);
   return 0;
}
```

Printing Char Value

Given an ASCII value, the C++ compiler can return the corresponding character. You declare a char variable and assign it an integer value. It will be converted to the corresponding character value.

Example:

```
#include <iostream>
using namespace std;

int main()
{
    char x = 65, y = 66, z = 79;
    cout << x << endl;
    cout << y << endl;
    cout << z << endl;

    return 0;
}
```

Output:

A

B

O

Since x was declared as a char, the char with ASCII value of 65 will be returned, that is, **A.**

Example

```
#include <iostream>
using namespace std;

int main()
{
   char num1 = '1', num2 = '2';
   cout << num1 + num2 << endl;
   return 0;
}
```

The output of this code is 99. The ASCII value of '1' is 49, and the ASCII value of '2' is 50. Thus, 49 + 50 = 99.

Converting Character to String

There exist a number of ways that we can use to convert characters to strings.

#1: Using Constructor given by a String Class

This can be done using the following syntax:

```
string st(int n,char x);
```

The parameter n denotes the size of the string that is to be generated.

The parameter x is the character to convert to a string.

The function returns a string.

Example:

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
   string st(1, 'C');
   cout << "The resulting string is : " << st;
   return 0;
}
```

Output:

```
The resulting string is : C
```

#2) Using the std::string Operators = and +=

The = and += operators are already overloaded with characters. The two can be used to convert a particular character to a string.

Example

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    string st;
    char b = 'B';
    st = 'A';
    st += b;
    cout << "The resulting string is : " << st;
    return 0;
}
```

#3: Using std::string Methods

- **push_back**

  This function assigns a particular character to a string's end. It is overloaded for characters.

  It takes the following syntax:

```
void push_back(char ch)
```

  The parameter ch is the character that is to be changed to a string.

- **append**

  It assigns many copies of a particular character to a string.

  The function takes the following syntax:

```
string &append(size_t n, char ch)
```

  The parameter n denotes the times that the character will be appended.

  The parameter ch is the character to append to string.

- **assign**

  This function replaces the current contents of the string with n copies of the specified character.

  It takes the following syntax:

```
string& assign(size_t n,char ch);
```

  The parameter n denotes the total copies for the character.

  The parameter ch is the character to copy into the string.

- **insert**

  The insert function inserts n copies of a character at the starting position of the string, as specified in the arguments.

  It takes the following syntax:

```
string& insert(size_t p,size_t n,char ch);
```

The p parameter denotes the position from the start where characters will be inserted.
The parameter n denotes the total copies for the character.
The parameter ch is the character to be insert in the string.

Example:

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    string st;
    st = "university";
    st.push_back('A');
    cout << "push_back A returns : " << st << endl;

    st = "university";
    st.append(3, 'C');
    cout << "append C returns : " << st << endl;

    st = "university";
    st.assign(2, 'D');
    cout << "assign D returns : " << st << endl;

    st = "university";
    st.insert(0, 1, 'E');
    cout << "insert single character returns : " << st << endl;
    return 0;
}
```

Output:

```
push_back A returns : universityA
append C returns : universityCCC
assign D returns : DD
insert single character returns : Euniversity
```