ASCII and Unicode

From Bit & Bytes to ASCII

ASCII Chart			
A	01000001	P	01010000
B	01000010	Q	01010001
C	01000011	R	01010010
D	01000100	S	01010011
E	01000101	T	01010100
F	01000110	U	01010101
G	01000111	V	01010110
H	01001000	W	01010111
I	01001001	X	01011000
J	01001010	Y	01011001
K	01001011	Z	01011010
L	01001100		
M	01001101		
N	01001110		
O	01001111		

- Bytes can represent any collection of items using a “look-up table” approach
- ASCII is used to represent characters

ASCII
American Standard Code for Information Interchange
http://en.wikipedia.org/wiki/ASCII

ASCII

- It is an acronym for the **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange.
- It is a standard seven-bit code that was first proposed by the American National Standards Institute or ANSI in 1963, and finalized in 1968 as ANSI Standard X3.4.
- The purpose of ASCII was to provide a standard code various to symbols (visible and invisible symbols)

ASCII

- In the **ASCII character set**, each binary value between 0 and 127 represents a specific character.
- Most computers extend the ASCII character set to use the full range of 256 characters available in a byte. The upper 128 characters handle special things like accented characters from common foreign languages.

- In general, ASCII works by assigning standard numeric values to letters, numbers, punctuation marks and other characters such as control codes.
- An uppercase "A," for example, is represented by the decimal number 65."

Bytes: ASCII

- By looking at the ASCII table, you can clearly see a one-to-one correspondence between each character and the ASCII code used.
- For example, 32 is the ASCII code for a space.
- We could expand these decimal numbers out to binary numbers (so 32 = 00100000), if we wanted to be technically correct -- that is how the computer really deals with things.

Bytes: ASCII

- Computers store text documents, both on disk and in memory, using these ASCII codes.
- For example, if you use Notepad in Windows XP/2000 to create a text file containing the words, "Four score and seven years ago," Notepad would use 1 byte of memory per character (including 1 byte for each space character between the words -- ASCII character 32).

- When Notepad stores the sentence in a file on disk, the file will also contain 1 byte per character and per space.
- Binary number is usually displayed as Hexadecimal to save display space.

- Take a look at a file size now.
- Take a look at the space of your p drive

Bytes: ASCII

- If you were to look at the file as a computer looks at it, you would find that each byte contains not a letter but a number -- the number is the ASCII code corresponding to the character (see below). So on disk, the numbers for the file look like this:
- F o u r a n d s e v e n
- 70 111 117 114 32 97 110 100 32 115 101 118 101 110

- Externally, it appears that human beings will use natural languages symbols to communicate with computer.
- But internally, computer will convert everything into binary data.
- Then process all information in binary world.
- Finally, computer will convert binary information to human understandable languages.

- When you type the letter A, the hardware logic built into the keyboard automatically translates that character into the ASCII code 65, which is then sent to the computer. Similarly, when the computer sends the ASCII code 65 to the screen, the letter A appears.

ascii

- ASCII stands for American Standard Code for Information Interchange
  - First published on October 6, 1960
- ASCII is a type of binary data

Ascii part 2

- ASCII is a character encoding scheme that encodes 128 different characters into 7 bit integers
- Computers can only read numbers, so ASCII is a numerical representation of special characters
  - Ex: ‘%’ ‘!’ ‘?’

Ascii part 3

- ASCII code assigns a number for each English character
- Each letter is assigned a number from 0-127
  - Ex: An uppercase ‘m’ has the ASCII code of 77
- By 2007, ASCII was the most commonly used character encoding program on the internet

ASCII Code: Character to Binary

0	0011 0000	O	0100 1111	m	0110 1101
1	0011 0001	P	0101 0000	n	0110 1110
2	0011 0010	Q	0101 0001	o	0110 1111
3	0011 0011	R	0101 0010	p	0111 0000
4	0011 0100	S	0101 0011	q	0111 0001
5	0011 0101	T	0101 0100	r	0111 0010
6	0011 0110	U	0101 0101	s	0111 0011
7	0011 0111	V	0101 0110	t	0111 0100
8	0011 1000	W	0101 0111	u	0111 0101
9	0011 1001	X	0101 1000	v	0111 0110
A	0100 0001	Y	0101 1001	w	0111 0111
B	0100 0010	Z	0101 1010	x	0111 1000
C	0100 0011	a	0110 0001	y	0111 1001
D	0100 0100	b	0110 0010	z	0111 1010
E	0100 0101	c	0110 0011	.	0010 1110
F	0100 0110	d	0110 0100	,	0010 0111
G	0100 0111	e	0110 0101	:	0011 1010
H	0100 1000	f	0110 0110	;	0011 1011
I	0100 1001	g	0110 0111	?	0011 1111
J	0100 1010	h	0110 1000	!	0010 0001
K	0100 1011	I	0110 1001	'	0010 1100
L	0100 1100	j	0110 1010	"	0010 0010
M	0100 1101	k	0110 1011	(	0010 1000
N	0100 1110	l	0110 1100	)	0010 1001
				space	0010 0000

revisit “char” data type

- In C, single characters are represented using the data type char, which is one of the most important scalar data types.

```
char achar;
achar='A';
achar=65;
```

Character and integer

- A character and an integer (actually a small integer spanning only 8 bits) are actually indistinguishable on their own. If you want to use it as a char, it will be a char, if you want to use it as an integer, it will be an integer, as long as you know how to use proper C++ statements to express your intentions.