

## String + Character:

In previous lessons, we seldom deal with string and char type.  
Now, we focus on them.

As mentioned before, computers can only store binary numbers like 0001, 1111.  
How do a computer store a character?

It uses number to represent a character following the below table.

### ASCII Code Table

*Copied from <http://www.asciitable.com/>*

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	<b>NUL</b> (null)	32	20	040	&#32;	Space	64	40	100	&#64;	@	96	60	140	&#96;	`
1	1	001	<b>SOH</b> (start of heading)	33	21	041	&#33;	!	65	41	101	&#65;	A	97	61	141	&#97;	a
2	2	002	<b>STX</b> (start of text)	34	22	042	&#34;	"	66	42	102	&#66;	B	98	62	142	&#98;	b
3	3	003	<b>ETX</b> (end of text)	35	23	043	&#35;	#	67	43	103	&#67;	C	99	63	143	&#99;	c
4	4	004	<b>EOT</b> (end of transmission)	36	24	044	&#36;	\$	68	44	104	&#68;	D	100	64	144	&#100;	d
5	5	005	<b>ENQ</b> (enquiry)	37	25	045	&#37;	%	69	45	105	&#69;	E	101	65	145	&#101;	e
6	6	006	<b>ACK</b> (acknowledge)	38	26	046	&#38;	&	70	46	106	&#70;	F	102	66	146	&#102;	f
7	7	007	<b>BEL</b> (bell)	39	27	047	&#39;	'	71	47	107	&#71;	G	103	67	147	&#103;	g
8	8	010	<b>BS</b> (backspace)	40	28	050	&#40;	(	72	48	110	&#72;	H	104	68	150	&#104;	h
9	9	011	<b>TAB</b> (horizontal tab)	41	29	051	&#41;	)	73	49	111	&#73;	I	105	69	151	&#105;	i
10	A	012	<b>LF</b> (NL line feed, new line)	42	2A	052	&#42;	*	74	4A	112	&#74;	J	106	6A	152	&#106;	j
11	B	013	<b>VT</b> (vertical tab)	43	2B	053	&#43;	+	75	4B	113	&#75;	K	107	6B	153	&#107;	k
12	C	014	<b>FF</b> (NP form feed, new page)	44	2C	054	&#44;	,	76	4C	114	&#76;	L	108	6C	154	&#108;	l
13	D	015	<b>CR</b> (carriage return)	45	2D	055	&#45;	-	77	4D	115	&#77;	M	109	6D	155	&#109;	m
14	E	016	<b>SO</b> (shift out)	46	2E	056	&#46;	.	78	4E	116	&#78;	N	110	6E	156	&#110;	n
15	F	017	<b>SI</b> (shift in)	47	2F	057	&#47;	/	79	4F	117	&#79;	O	111	6F	157	&#111;	o
16	10	020	<b>DLE</b> (data link escape)	48	30	060	&#48;	0	80	50	120	&#80;	P	112	70	160	&#112;	p
17	11	021	<b>DC1</b> (device control 1)	49	31	061	&#49;	1	81	51	121	&#81;	Q	113	71	161	&#113;	q
18	12	022	<b>DC2</b> (device control 2)	50	32	062	&#50;	2	82	52	122	&#82;	R	114	72	162	&#114;	r
19	13	023	<b>DC3</b> (device control 3)	51	33	063	&#51;	3	83	53	123	&#83;	S	115	73	163	&#115;	s
20	14	024	<b>DC4</b> (device control 4)	52	34	064	&#52;	4	84	54	124	&#84;	T	116	74	164	&#116;	t
21	15	025	<b>NAK</b> (negative acknowledge)	53	35	065	&#53;	5	85	55	125	&#85;	U	117	75	165	&#117;	u
22	16	026	<b>SYN</b> (synchronous idle)	54	36	066	&#54;	6	86	56	126	&#86;	V	118	76	166	&#118;	v
23	17	027	<b>ETB</b> (end of trans. block)	55	37	067	&#55;	7	87	57	127	&#87;	W	119	77	167	&#119;	w
24	18	030	<b>CAN</b> (cancel)	56	38	070	&#56;	8	88	58	130	&#88;	X	120	78	170	&#120;	x
25	19	031	<b>EM</b> (end of medium)	57	39	071	&#57;	9	89	59	131	&#89;	Y	121	79	171	&#121;	y
26	1A	032	<b>SUB</b> (substitute)	58	3A	072	&#58;	:	90	5A	132	&#90;	Z	122	7A	172	&#122;	z
27	1B	033	<b>ESC</b> (escape)	59	3B	073	&#59;	;	91	5B	133	&#91;	[	123	7B	173	&#123;	{
28	1C	034	<b>FS</b> (file separator)	60	3C	074	&#60;	<	92	5C	134	&#92;	\	124	7C	174	&#124;	
29	1D	035	<b>GS</b> (group separator)	61	3D	075	&#61;	=	93	5D	135	&#93;	]	125	7D	175	&#125;	}
30	1E	036	<b>RS</b> (record separator)	62	3E	076	&#62;	>	94	5E	136	&#94;	^	126	7E	176	&#126;	~
31	1F	037	<b>US</b> (unit separator)	63	3F	077	&#63;	?	95	5F	137	&#95;	_	127	7F	177	&#127;	DEL

Source: [www.LookupTables.com](http://www.LookupTables.com)

Dec: Decimal (10 進制)

Hex: Hexadecimal (16 進制)

Oct: Octal (8 進制)

Html: Web Page Special Code.

ASCII stands for American Standard Code for Information Interchange.

Therefore, a space is 32, Capital Letter A is 65.

From now on, you must accept the value of a space is 32, “A” is 65, “a” is 97 and so on.

\* This is a very important topic, Please pay more attention to it. \*

## String + Character Processing I:

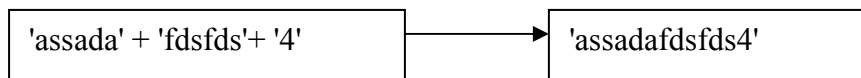
To revise, let's look at the below table, which is an operator list.

Expression	A Data type	B Data type	Result Data type	Meaning
A + B	Integer	Integer	Integer	Addition
	Integer	Real	""	""
	Real	Integer	Real	""
	Real	Real	""	""
	Char	Char	String	Concatenation (接合)
	Char	String	""	""
	String	Char	""	""
	String	String	""	""

Notes: You can never add a string to an integer, it is treated as syntax error.

You may notice that the + operator has different results in different situation. There are only 2 operations for string or char type, *Concatenation* and *Comparison*.

### Concatenation:



### Comparison:

Similar to numbers, we can compare character.

### String Comparison Operators:

A and B are character data type.

Expression	Result	Meaning
A = B	Boolean	Is A exactly equal to B?
A > B	Boolean	Compare the ASCII value

### Example:

'F' is greater than 'A' since 'F' is 70 and 'A' is 65.

'z' is greater than 'A'.

'F' is greater than '9'.

### Special value needed to memorize:

Space is 32. '0' is 48. 'A' is 65. 'a' is 97.

Notes: 'A' to 'Z' 'a' to 'z', '0' to '9' is sorted in order.

## String + Character Processing II:

Sometimes, we are interested in the ASCII value of a character.  
For example, the '@' character.  
Inversely, given a ASCII value, I am interested in what it represents.  
For example, a value 66.

We can check from the above table to get the value of it, but it's very tedious.

Here, I introduce two new Pascal functions to you.

### Ord (Ordinary value of..)

**ord(z)** : Return the order of z counting from zero.

#### Example:

Since for Boolean type, false comes first and followed by TRUE.

ord(true) will return 1, ord(false) will return 0.  
ord('a') will return 97.

*Notes that z must be ordinary.*

So a statement like ord(1.223) is definitely a syntax error.

### Chr (Character of that value..)

**chr(z)** : Return the character that z represents.

#### Example:

Chr(97) will return 'a'.  
Chr(65) will return 'A'.  
Chr(10) will return a backspace character.

You can also stick two character together using the + operator.  
Chr(79)+Chr(73) will return 'OI'.

Expression like this also works.

Chr(Ord('G') + 1) return 'H'. This kind is called function of function.

First, calculate Ord('G'), which gives 71.  
Second, add 1 to 71, gets 72.  
Finally calculate Chr(72), which is 'H'.

## String + Character Processing III:

Like integer variables, character variables can also perform self-incremented.

But, we don't use statement like `A:=a+1;` to increment or decrement a character type variable.

We use procedures called **inc** and **dec** to do so.

**Inc(x) : Self-increment variable x by 1.**  
**Dec(x) : Self-decrement variable x by 1.**

The example will give an output 'db'.  
After the statement `inc(x)`, x will store 'd'.  
After the statement `dec(y)`, y will store 'b'.

```
Var x,y:char;  
Begin  
X:='c';  
Y:='c';  
Inc(x);  
Write(x);  
Dec(y);  
Writeln(y);  
End.
```

**Inc(x,2) : Self-increment variable x by 2**  
**Inc(x,3) : Self-increment variable x by 3**

...

**Dec(x,2) : Self-decrement variable x by 2**  
**Dec(x,3) : Self-decrement variable x by 3**

...

This time, it outputs 'ea'.

```
Var x,y:char;  
Begin  
X:='c';  
Y:='c';  
Inc(x,2);  
Write(x);  
Dec(y,2);  
Writeln(y);  
End.
```

**Try to write 2 programs that generates output: (*Italic and bold : user-input.*)**

```
Enter number of lines:4  
A  
AC  
ACE  
ACEG
```

```
Enter number of lines:3  
A  
CAC  
ECACE
```

```
Enter number of lines:5  
A  
AC  
ACE  
ACEG  
ACEGI
```

```
Enter number of lines:4  
A  
CAC  
ECACE  
GECACEG
```

*Hints : Try to use ord and chr functions.*

## String + Character Processing IV:

String is a very special data type. It has a default maximum length 255.  
You can treat it as array [0..255] of char , but there is a few difference between them.

We can access the n<sup>th</sup> character of a string by using syntax like accessing an element of array.

```
var s:string;  
begin  
s:='asfbds';  
writeln(s[1]); {Output the 1st character of s}  
end.
```

Since a string default maximum length is 255, it may be very wasteful to declare one to store very short strings like 'aaa', 'bbb'.

We can control the maximum length of a string like the below example.

```
var s:string[20]; {Max length 20}  
begin  
s:='asfbds';  
writeln(s[1]); {Output the 1st character of s}  
end.
```

Special Case:

```
var s1:string[20]; {Max length 20}  
    s2:string[3]; {Max length 3}  
begin  
s1:='asfbds';  
s2:=s1;  
writeln(s2); {Output 'asf'}  
end.
```

Since s2 can store max 3 characters, when storing s1 to s2, ignore the remaining characters 'bds'.

Therefore The output is 'asf'.

## String + Character Processing V:

**String functions:**

Var s , s1 , s2 , s3 : string ; n , m : integer ;

Function	Meaning
Length(s)	Return the length of s.
Pos(s1,s)	Return the position of first string s1 in s. Zero for not found.
Copy(s, N , M)	Extract M characters from s starting from N <sup>th</sup> character.
Concat(s1,s2,s3)	Exactly the same as s1+s2+s3 , and so on.

**Example:**

Length('abcdef') return 6.

Pos('a','bccassa') return 4. {Because the first a is at position 4.}

Pos('asssa', 'asss333a') return 0. { Because there is no 'asssa' in 'asss333a' }

Copy('abcdefghi' , 2 , 5) return 'bcdef'. {Start from 2<sup>nd</sup> character , draw 5 characters.}

```

Var s1,s:string; I:integer;
Begin
Readln(s);
S1:="";
For I:=1 to length(s) do s1:=s[i]+s1;
writeln(s1);
End.
{This program reverse string s and store it to s1}

```

```

Var s1,s:string; I:integer;
Begin
Readln(s);
For I:=length(s) downto 1 do Write(s[I]);
Writeln;
End.
{This program reverse string s and store it to s1}

```

**Try to write 2 programs that generates output: (*Italic and bold : user-input.*)**

```

Enter a string:ACEG
A
AC
ACE
ACEG

```

```

Enter a string:ACE
A
CAC
ECACE

```

```

Enter number of lines:ACEGI
A
AC
ACE
ACEG
ACEGI

```

```

Enter number of lines:ACEG
A
CAC
ECACE
GECACEG

```

*Hints : Try to use length and copy functions.*

## String + Character Processing VI:

Try to write program to remove all spaces of a string: (*Italic and bold : user-input.*)

Enter a string: *ACEG d*  
ACEGd

Enter number of lines: *Hello world*  
Helloworld

Hints: Use the pos function and copy function to do so.

### String Procedures:

Var s,s2: string; n1,n2, x , e: integer; r : real;

Procedure	Meaning
Val(s,x,e)	Convert a string s to numeric and store to x, Store the error position to e.
Str(r,s)	Convert a real number to string, store to s. {Second colon operator is allow}
Str(x,s)	Convert an integer to string, store to s. {Only one colon is allow}
Delete(s, N , M)	Delete M characters from s starting from N <sup>th</sup> character.
Insert(s2,s,n)	Insert s2 to s before the n <sup>th</sup> character.

```

Var S: String;
begin
  S := 'Honest Lincoln';
  Insert('Abe ', S, 8);    { 'Honest Abe Lincoln' }
end.

```

```

Var S: String; R:real;
begin
  R:=1.247;
  Str(r:0:2,s) {S will store '1.24'}
end.

```

```

Var S: String;
begin
  S := 'abcdefgh';
  Delete(s,2,5);    { 'agh' }
end.

```

```

Var S: String; R:real;
begin
  R:=1.247;
  Str(r:0:3,s) {S will store '1.247'}
end.

```

```

Var S: String; R:real;
begin
  R:=1.247;
  Str(r,s) {S will store '∇1.247000000E+00'}
end.    {Never forget the space}

```

```

Var S: String; I:integer;
begin
  i:=10;
  Str(i,s) {S will store '10'}
end.

```

```

Var S: String; I , e : integer;
begin
  S := '123';
  Val (s,i,e); { Since '123' is a valid integer , no error}
end.    { So , I store 123 , e store 0 }

```

```

Var S: String; I:integer;
begin
  i:=10;
  Str(I:5,s) {S will store '∇∇∇10'}
end.

```

Exercise: string.doc , array ex\_1.doc