

## Chapter 9                      C-style String

We have learnt array, and in fact **string** is a **char array**.

But to deal with **string** we cannot use the usual assignment, addition etc. operators.  
We will use **char\*** , **char[]** or **char[N]** to represent a **C-style string**

### Syntax:

```
char Name[100]="ABCDEF";
```

```
char Name[]="ABCDEF";
```

*Notes:* Array without index meaning the index depends on the value given at right.  
So for the right one, the computer knows it means Name[7];

### Why is it called C-style string?

The computer knows only that Name can store 100 characters.

The array data type doesn't give any information of the length of a string.

Therefore, in C, we use a special character to indicate the ending of a string.

That character is called the NULL character, the zero character in the ASCII table.

For the example above, Name actually used 7 spaces to store "ABCDEF" the 7-th one is invisible to us, which is the '\0' character.

### Wrong usage of char\*:

```
Name="WKC"; // This is wrong
```

But we can still use Name[0]='W'; Name[1]='K'; Name[2]='C';

With these 3 statements, we force the computer to store the characters directly.

But when we try to use printf to display it, some strange output will be given.

*Notes:* At the format, use "%s" to display string.

```
char Name[100]="ABCDEF";  
Name[0]='W';  
Name[1]='K';  
Name[2]='C';  
printf("%s",Name);
```

The output is probably "WKCDEF" instead of "WKC" as we haven't changed the ending indicator.

To have a correct output, we need to have one more statement, Name[3]= '\0'; before printf.

But it is very tedious job to handle things like in this way.

So, we will use C-style string function instead.

**C-style string functions:**

To use **char\*** properly, we have to include the new library with the statement  
`#include <string.h>`

For detail: <http://www.cplusplus.com/reference/clibrary/cstring/>

It has the following functions to handle **char\***.

Like the scanf function which can change value of variables, so do the following.

**Assignment**

<code>char* strcpy(char* dest, char* source);</code>	Copy the string source to dest
--	--------------------------------

As strcpy is a function, it has a return value.

That value is essentially the same as dest.

*Notes:* In fact, any data type followed by a \* meaning a **pointer to that data type**.

**Examples:**

```
strcpy(Name, "12ala"); // Store "12ala" to Name
printf("%s\n", Name); // Use "%s" to display string
printf("%s\n", strcpy(Name, "Second Line"));
```

**Append**

<code>char* strcat(char* dest, char* source);</code>	Append the string source to dest
--	----------------------------------

Return value is the same as dest.

**Examples:**

```
strcpy(Name, "12ala"); // Store "12ala" to Name
strcat(Name, "21222"); // "21222" is added to the end of Name
printf("%s\n", Name); // Display "12ala21222"
```

**Length**

<code>int strlen(char* source);</code>	Return the length of source
--	-----------------------------

**Examples:**

```
printf("%s\n", strlen("12abc")); // Display 5
```

**Comparison**

<code>int strcmp(char* s1, char* s2);</code>	Compare both string
--	---------------------

If s1 is greater than s2 then return a positive integer, s1 equal to s2 return 0, otherwise a negative integer.

**HKOI past paper:**

2007HJE A: #9, #25

Whole Paper 2006HJE

**End of Chapter**