

## **char \*s="Hello" - Error**

char \*s="Hello" may not be supported in all compilers.

change it to char s[]="Hello";

## **String Literal - char \*s="Hello";**

"Hello" is a literal.

Literal means direct values used in a program. Like int x=10;

Literals are stored in code section.

Literal cannot be modified. Like s[2]='k'; is invalid.

## **char vs string**

**char** type of variable can store a single character.

### **example:**

```
char c='A';
```

**string** is an array of characters used for storing a name, word , sentence etc.

It is terminated by '\0'

### **Example:**

```
char str[6]={'H','e','l','l','o','\0'};
```

```
char str[6]="Hello"; // '\0' will be included at the end.
```

## **class string vs char str[]**

**string** is a builtin class in C++ to store a string.

Internally it contains an array of characters.

It will create array of characters in heap.

It has many functions for performing string operations

**example:**

```
string str="Hello" // this will create a string object.
```

**char str[10]**="Hello"; is a C style string. It can contain set of characters terminated by '\0'

## **char \*s vs char s[10]**

**char s[10];** is an array of characters. It can contain string

**Example:**

```
char str[6]={ 'H','e','l','l','o','\0'}; // '\0' also takes space
```

**char \*s;** is pointer of type character. It can point on a char array or string

Example:

```
1.char *s="Hello"; // s is pointing to a string literal
```

```
2.char str[]={ 'H','e','l','l','o','\0'};
```

```
char *s=str; // s is pointing to a string in array.
```

## **getline(cin,str) vs cin.getline(str,100)**

**getline(cin,str)** is used for reading a string object. It will not work for char array.

**cin.getline(str,100)** is used for reading a string in char array. It will not work for string class object

### **example:**

1. string str;

getline(cin,str); it is used with string class.

2. char str[10];

cin.getline(str,100); it is used with char array.

## **sizeof() vs strlen()**

**sizeof** is used for finding size of a data type or variable. It cannot find string length

**strlen(str)** is a function used for finding length of a string.

## **strtok()**

It is used for tokenising a string

In first call we pass a string from second call we pass NULL, it will use same string given in first call.

### **First call**

**strtok("hello:how:are:you",":");** this is the first call

- In first call it will return first token "hello" as a string

- `char *t;` is used for taking tokens.
- In first call, it will store the string internally.

## Second call

**`strtok(NULL,":");`** this is the second call.

- NULL means, it will use the same string given in first call
- From second call onwards it will give second token onwards.
- Same call is repeated with NULL to get all the tokens.

## CHECKING PALINDROME (Common Mistakes)

```
#include <iostream>
```

```
#include <cmath>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
string str="MalayaLAm";
```

```
string rev;
```

```
int len=(int)str.length();
```

```
int i,j;
```

```
rev.resize(len); // resizing of reverse string must be done, rev will be empty.
```

```
for(i=0, j=len-1; i<len; i++, j--)
```

```
{
```

```
rev[i]=str[j];
```

```
}
```

```
rev[len]='\0';
```

```
for(i=0; str[i]!='\0' && rev[i]!='\0'; i++)
```

```
{
```

```
if(str[i]-rev[i]!=32 && str[i]-rev[i]!=-32 && str[i]-rev[i]!=0)
```

```
{  
break;  
}  
}  
if(str[i]=='\0' && rev[i]=='\0')  
cout<<"palindrome"<<endl;  
else cout<<"not a palindrome"<<endl;  
}
```