


# Difference between `const char *p`, `char * const p` and `const char * const p`

Prerequisite: [Pointers](#)

There is a lot of confusion when `char`, `const`, `*`, `p` are all used in different permutations and meanings change according to which is placed where. Following article focus on differentiation and usage of all of these.

The qualifier `const` can be applied to the declaration of any variable to specify that its value will not be changed. `const` keyword applies to whatever is immediately to its left. If there is nothing to its left, it applies to whatever is immediately to its right.

1. **`const char *ptr`** : This is a pointer to a constant character. **You cannot change the value pointed by `ptr`, but you can change the pointer itself.** "`const char *`" is a (non-const) pointer to a const char.



```
1 // C program to illustrate
2 // char const *p
3 #include<stdio.h>
4 #include<stdlib.h>
5
6 int main()
7 {
8     char a='A', b='B';
9     const char *ptr = &a;
10
11     /*ptr = b; illegal (assignment of read-only location *ptr)
12
13     // ptr can be changed
14     printf( "value pointed to by ptr: %c\n", *ptr);
15     ptr = &b;
16     printf( "value pointed to by ptr: %c\n", *ptr);
17 }
18
```

Output:

```
value pointed to by ptr:A
value pointed to by ptr:B
```

**NOTE:** There is no difference between `const char *p` and `char const *p` as both are pointer to a const char and position of `*`(asterik) is also same.

2. **char \*const ptr** : This is a constant pointer to non-constant character. **You cannot change the pointer p, but can change the value pointed by ptr.**

```
1 // C program to illustrate
2 // char* const p
3 #include<stdio.h>
4 #include<stdlib.h>
5
6 int main()
7 {
8     char a='A', b='B';
9     char *const ptr = &a;
10    printf( "Value pointed to by ptr: %c\n", *ptr);
11    printf( "Address ptr is pointing to: %d\n\n", ptr);
12
13    //ptr = &b; illegal (assignment of read-only variable ptr)
14
15    // changing the value at the address ptr is pointing to
16    *ptr = b;
17    printf( "Value pointed to by ptr: %c\n", *ptr);
18    printf( "Address ptr is pointing to: %d\n", ptr);
19 }
20
```

Output:

```
Value pointed to by ptr: A
Address ptr is pointing to: -1443150762
```

**NOTE:** Pointer always points to same address, only the value at the location is changed.

3. **const char \* const ptr** : This is a constant pointer to constant character. **You can neither change the value pointed by ptr nor the pointer ptr.**

```
// C program to illustrate
//const char * const ptr
#include<stdio.h>
#include<stdlib.h>

int main()
{
    char a='A', b='B';
    const char *const ptr = &a;

    printf( "Value pointed to by ptr: %c\n", *ptr);
    printf( "Address ptr is pointing to: %d\n\n", ptr);

    // ptr = &b; illegal (assignment of read-only variable ptr)
    // *ptr = b; illegal (assignment of read-only location *ptr)
}
```

Output:

```
Value pointed to by ptr: A  
Address ptr is pointing to: -255095482
```

**NOTE:** `char const * const ptr` is same as `const char *const ptr`.