Difference between const char \*p, char \* const p and const char \* const p

* Difficulty Level : [Easy](https://www.geeksforgeeks.org/easy/)
* Last Updated : 09 Oct, 2018

Prerequisite: [Pointers](https://www.geeksforgeeks.org/pointers-in-c-and-c-set-1-introduction-arithmetic-and-array/)  
There is a lot of confusion when char, const, \*, p are all used in different permutaions and meanings change according to which is placed where. Following article focus on differentiation and usage of all of these.

The qualifier [const](https://www.geeksforgeeks.org/const-qualifier-in-c/) 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.

|  |
| --- |
| // C program to illustrate  // char const \*p  #include<stdio.h>  #include<stdlib.h>    int main()  {      char a ='A', b ='B';      const char \*ptr = &a;        //\*ptr = b; illegal statement (assignment of read-only location \*ptr)        // ptr can be changed      printf( "value pointed to by ptr: %c\n", \*ptr);      ptr = &b;      printf( "value pointed to by ptr: %c\n", \*ptr);  } |

1. Output:
2. value pointed to by ptr:A
3. value pointed to by ptr:B
4. **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.
5. **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.**

|  |
| --- |
| // C program to illustrate  // char\* const p  #include<stdio.h>  #include<stdlib.h>    int main()  {      char a ='A', b ='B';      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 statement (assignment of read-only variable ptr)        // changing the value at the address ptr is pointing to      \*ptr = b;      printf( "Value pointed to by ptr: %c\n", \*ptr);      printf( "Address ptr is pointing to: %d\n", ptr);  } |

1. Output:
2. Value pointed to by ptr: A
3. Address ptr is pointing to: -1443150762
4. Value pointed to by ptr: B
5. Address ptr is pointing to: -1443150762
6. **NOTE:** Pointer always points to same address, only the value at the location is changed.
7. **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 statement (assignment of read-only variable ptr)      // \*ptr = b; illegal statement (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**.