

A computer science portal for geeks		
Custom Search		
	Geeks Classes	Login
	Write an Article	
N.	lacros vs Functions	
		ne macros would be processed before your program
	mpiles. However, functions are not preprocessed b	
Se	e the following example of Macro:	
	nclude <stdio.h></stdio.h>	
	efine NUMBER 10 t main()	
ι	<pre>printf("%d", NUMBER); return 0;</pre>	
}		Run on IDE
		Rull off IDE
Οι	itput:	
1	0	
Se	e the following example of Function:	
	nclude <stdio.h></stdio.h>	
	t number()	
}	return 10; t main()	
{	<pre>printf("%d", number());</pre>	
}	return 0;	
		Run on IDE
O۱	ıtput:	
J	itput.	
1	0	
Now compile them using the command:		
	•	

gcc -E file_name.c

This will give you the executable code as shown in the figure:

```
# 943 "/usr/include/stdio.h" 3 4

# 2 "example.c" 2
int main(){
  printf("%d", 10);
  return 0;
}
pranjal@ubuntu:~/Desktop$
```

```
# 2 "example.c" 2
int number(){
  return 10;
  }
int main(){
  printf("%d", number());
  return 0;
}
pranjal@ubuntu:~/Desktop$ []
```

This shows that the macros are preprocessed while functions are not.

In macros, no type checking(incompatible operand, etc.) is done and thus use of micros can lead to errors/side-effects in some cases. However, this is not the case with functions. Also, macros do not check for compilation error (if any). Consider the following two codes:

Macros:

```
#include<stdio.h>
#define CUBE(b) b*b*b
int main()
{
    printf("%d", CUBE(1+2));
    return 0;
}
```

Run on IDE

Output: Unexpected output

7

Functions:

```
#include<stdio.h>
int cube(int a)
{
    return a*a*a;
}
int main()
{
    printf("%d", cube(1+2));
    return 0;
}
```

Run on IDE

Output: As expected

27

- Macros are usually one liner. However, they can consist of more than one line, Click here to see the usage. There are no such constraints in functions.
- The speed at which macros and functions differs. Macros are typically faster than functions as they don't involve actual function call overhead.

Conclusion:

Macros are no longer recommended as they cause following issues. There is a better way in modern compilers that is inline functions and const variable. Below are disadvantages of macros:

- a) There is no type checking
- b) Difficult to debug as they cause simple replacement.
- c) Macro don't have namespace, so a macro in one section of code can affect other section.
- d) Macros can cause side effects as shown in above CUBE() example.

See following for more details on macros:

Interesting facts about Macros and Preprocessors

This article is contributed by **Pranjal Mathur**. If you like GeeksforGeeks and would like to contribute, you can also write an article and mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

Practice Tags : C CPP

Article Tags : C C++ CPP Functions cpp-macros

Login to Improve this Article

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Recommended Posts:

Interesting Facts about Macros and Preprocessors in C

Multiline macros in C

Inline Functions in C++

How to print size of array parameter in C++?

Data Type Ranges and their macros in C++

Message encryption and decryption using UDP server

Storage of integer and character values in C

How will you print numbers from 1 to 100 without using loop? | Set-2

Common Memory/Pointer Related bug in C Programs

Format specifiers in C