Function Interposition in C with an example of user defined malloc()

int (1 sign bit + 31 data bits) keyword in C

Program error signals

Why array index starts from zero ?

TCP Server-Client implementation in C

How to return multiple values from a function in C or C++?

Dynamic Memory Allocation in

C using malloc(), calloc(), free() and realloc()

Commonly Asked C Programming Interview Questions | Set 3

Applications of Pointers in C/C++

Pre-increment and Post-increment in C/C++

Sum of array Elements without using loops and recursion

“static const” vs “#define” vs “enum”

Comments in C/C++

How to find Segmentation Error in C & C++ ? (Using GDB)

Why strcpy and strncpy are not safe to use?

time() function in C

Loader in C/C++

GDB (Step by Step Introduction)

Communication between two process using signals in C

tolower() function in C

Passing Reference to a Pointer in C++

Difference between const int*, const int * const, and int const *

How does a C program executes?

Difference

between Call by Value and Call by Reference

How to avoid Structure Padding in C?

strrev() function in C

Interesting facts about C Language

Difference between fundamental data types and derived data types

C++: Methods of code shortening in competitive programming

Inline function in C

Macros vs Functions

Macros are **pre-processed** which means that all the macros would be processed before your program compiles. However, functions are **not preprocessed but compiled**.

See the following example of Macro:

```c
#include<stdio.h>
#define NUMBER 10
int main()
{
     printf("%d", NUMBER);
     return 0;
}
```

Output:

```
10
```

See the following example of Function:

```c
#include<stdio.h>
int number()
{
    return 10;
}
int main()
{
    printf("%d", number());
    return 0;
}
```

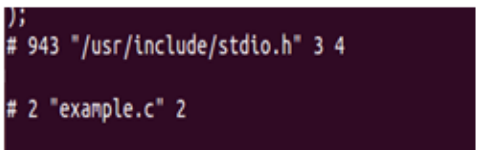
Output:

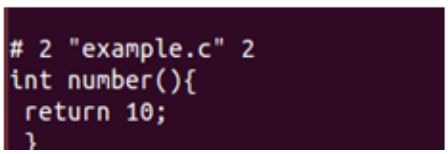
```
10
```

Now compile them using the command:

gcc –E file_name.c

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





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

```
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:

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

Output: Unexpected output

```
7
```

Functions:

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

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.

MACRO	FUNCTION
Macro is Preprocessed	Function is Compiled
No Type Checking is done in Macro	Type Checking is Done in Function
Using Macro increases the code length	Using Function keeps the code length unaffected
Use of macro can lead to side effect at later stages	Functions do not lead to any side effect in any case
Speed of Execution using Macro is Faster	Speed of Execution using Function is Slower
Before Compilation, macro name is replaced by macro value	During function call, transfer of control takes place
Macros are useful when small code is	Functions are useful when large code

MACRO	FUNCTION
repeated many times	is to be written
Macro does not check any Compile-Time Errors	Function checks Compile-Time Errors

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.

Recommended Posts:

Importance of macros in C++
Multiline macros in C
Branch prediction macros in GCC
Hygienic Macros : An Introduction
Output of C++ programs | Set 25 (Macros)
Data Type Ranges and their macros in C++
Interesting Facts about Macros and Preprocessors in C
Variable length arguments for Macros
Output of the Program | Use Macros Carefully!
Functions in C/C++
C | Functions | Question 8
C | Functions | Question 2
C | Functions | Question 11
C | Functions | Question 7

C | Functions | Question 4

Improved By : RishabhPrabhu

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

Practice Tags : C CPP

2

To-do Done

1.8

Based on **5** vote(s)

Feedback Add Notes Improve Article

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

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments Share this post!