



typedef versus #define

typedef: The typedef is used to give data type a new name. For example,

```
// C program to demonstrate typedef
#include <stdio.h>

// After this line BYTE can be used
// in place of unsigned char
typedef unsigned char BYTE;

int main()
{
    BYTE b1, b2;
    b1 = 'c';
    printf("%c ", b1);
    return 0;
}
```

Output:

c

#define in C is a directive which is used to #define alias.

```
// C program to demonstrate #define
```

```
#include <stdio.h>

// After this line HYD is replaced by
// "Hyderabad"
#define HYD "Hyderabad"

int main()
{
    printf("%s ", HYD);
    return 0;
}
```

Output:

Hyderabad

Difference between typedef and #define:

1. typedef is limited to giving symbolic names to types only, whereas #define can be used to define an alias for values as well, e.g., you can define 1 as ONE, 3.14 as PI, etc.
2. typedef interpretation is performed by the compiler where #define statements are performed by preprocessor.
3. #define should not be terminated with a semicolon, but typedef should be terminated with semicolon.
4. #define will just copy-paste the definition values at the point of use, while typedef is the actual definition of a new type.
5. typedef follows the scope rule which means if a new type is defined in a scope (inside a function), then the new type name will only be visible till the scope is there. In case of #define, when preprocessor encounters #define, it replaces all the occurrences, after that (No scope rule is followed).

```
// C program to demonstrate importance
// of typedef over #define for data types
#include <stdio.h>
typedef char* ptr;
#define PTR char*
int main()
{
    ptr a, b, c;
    PTR x, y, z;
    printf("sizeof a:%u\n", sizeof(a));
    printf("sizeof b:%u\n", sizeof(b));
    printf("sizeof c:%u\n", sizeof(c));
    printf("sizeof x:%u\n", sizeof(x));
    printf("sizeof y:%u\n", sizeof(y));
    printf("sizeof z:%u\n", sizeof(z));
    return 0;
}
```

Output:

sizeof a:8
sizeof b:8
sizeof c:8

```
sizeof x:8  
sizeof y:1  
sizeof z:1
```

From the output of the above program size of "a" which is a pointer is 8 (on a machine where pointers are stored using 8 bytes). In the above program, when the compiler comes to

```
typedef char* ptr;  
ptr a, b, c;
```

the statement effectively becomes

```
char *a, *b, *c;
```

This declares a, b, c as char*.

In contrast, #define works like this:

```
#define PTR char*  
PTR x, y, z;
```

the statement effectively becomes

```
char *x, y, z;
```

This makes x, y and z different, as, x is pointer-to-a char, whereas, y and z are char variables. When we declare macros with pointers while defining if we declare more than one identifier then the actual definition is given to the first identifier and for the rest non-pointer definition is given. In the above case x will be declared as char*, so its size is the size of a pointer, whereas, y and z will be declared as char so, their size will be 1 byte.

This article is contributed by **HARISH KUMAR**. If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](https://www.geeksforgeeks.org/contribute) or 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:

[Is there any equivalent to typedef of C/C++ in Java ?](#)

[For Versus While](#)

[Scope Resolution Operator Versus this pointer in C++?](#)

[Difference between #define and const in C?](#)

["static const" vs "#define" vs "enum"](#)

[<complex.h> header file in C with Examples](#)

[tgamma\(\) method in C/C++ with Examples](#)

[Modulo Operator \(%\) in C/C++ with Examples](#)

[fpclassify\(\) method in C/C++ with Examples](#)

[getch\(\) function in C with Examples](#)

[Chain of Pointers in C with Examples](#)

[Pointer Expressions in C with Examples](#)

[Speed up Code executions with help of Pragma in C/C++](#)

[Introduction to the C99 Programming Language : Part I](#)

Improved By : [code_master5](#), [yash_09](#)

Article Tags : [C](#) [C-Macro & Preprocessor](#)

Practice Tags : [C](#)



32

2.7

☐ To-do ☐ Done

Based on **26** vote(s)

[Feedback/ Suggest Improvement](#)[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.

[Comments](#)[Community](#)[Privacy Policy](#)[Login](#)[Recommend](#) 2[Tweet](#)[Share](#)[Sort by Newest](#)

LOG IN WITH

OR SIGN UP WITH DISQUS [?](#)**Vincent Balcita** • 2 months ago

can someone explain why the value returned is 3 and not pi
#define PI 22/7

[^](#) | [v](#) • [Reply](#) • [Share](#)**Samyu** • 2 years ago

Can anyone explain what is exactly happening?
#define PTR char*

PTR x, y, z;

[^](#) | [v](#) • [Reply](#) • [Share](#)**Rahul Birari** → Samyu • 2 years ago

It should have been this way
ptr expanded declaration as char *x, * y, *z;
whereas PTR expanded as char *x, y, z; <--here * is
associated only with the x
So, Define acted just as a place holder whereas,
typedef did a smart job of initializing each variable as
per the defined type which is char*

Hope this helps.

[1](#) [^](#) | [v](#) • [Reply](#) • [Share](#)**jyoti ranjan** • 2 years ago

Hello,

can anyone able to understand the below code ?

```
#include<stdio.h>
#define sqr(x) (x*x)
int main()
{
    int a;
```

```
... ,  
int b = 4;  
a=sqr(b+2);  
printf("%d",a);  
return 0;  
}
```

output is 14

please post your solution. i am unable to get

^ | v • Reply • Share ›



Prakash → jyoti ranjan • 2 years ago

Here, `sqr(b+2)` has been replaced at preprocessing time. So, the code will be like:

`a = sqr(b+2) -> a = (b+2*b+2) => (4+2*4+2) => (4+8+2)`
`=> 14`

That's why the value is 14.

1 ^ | v • Reply • Share ›



cool_shark • 3 years ago • edited

I understand this article is for C, but for C++11 or C++14,
'using' is preferred to typedef. Example:
`using ptr = char*;`

A computer science portal for geeks

5th Floor, A-118,
Sector-136, Noida, Uttar Pradesh - 201305
feedback@geeksforgeeks.org

COMPANY

About Us
Careers
Privacy Policy
Contact Us

PRACTICE

Courses
Company-wise
Topic-wise
How to begin?

LEARN

Algorithms
Data Structures
Languages
CS Subjects
Video Tutorials

CONTRIBUTE

Write an Article
Write Interview Experience
Internships
Videos

@geeksforgeeks, Some rights reserved