**typedef vs #define**

The **#define** is a C-directive which is also used to define the aliases for various data types similar to **typedef** but with following differences:

* The **typedef** is limited to giving symbolic names to types only where as **#define** can be used to define alias for values as well, like you can define 1 as ONE etc.
* The **typedef** interpretation is performed by the compiler where as **#define** statements are processed by the pre-processor.

**definition**

Typedef is a keyword that is used to give a new symbolic name for the existing name in a C program. This is same like defining alias for the commands.

* Consider the below structure.

struct student  
{  
         int mark [2];  
         char name [10];  
         float average;  
}

typedef struct student status;

* When we use “typedef” keyword before struct <tag\_name> like above, after that we can simply use type definition “status” in the C program to declare structure variable.
* Now, structure variable declaration will be, “status record”.
* This is equal to “struct student record”. Type definition for “struct student” is status. i.e. status = “struct student”

**definition**

The #define creates a macro, which is the association of an identifier or parameterized identifier with a token string. After the macro is defined, the compiler can substitute the token string for each occurrence of the identifier in the source file.

## [Syntax](javascript:void(0))

* #define identifier token-stringopt
* #define identifier( identifieropt, ... , identifieropt ) token-stringopt