**Macros vs Functions in C**

The macros are pre-processed, so it means that all the macros will be preprocessed while it is compiled. The functions are not preprocessed, but compiled.

In macros no type checking is done, so it may occur some problems for different types of inputs. In the case of functions, this is not done. Also for macros if the inputs are not properly maintained, then it may generate some invalid results. Please check the following program to get the idea about the problem.

Example

#include <stdio.h>

#define SQUARE(x) x \* x

int sqr(int x) {

   return x\*x;

}

main() {

   printf("Use of sqr(). The value of sqr(3+2): %d", sqr(3+2));

   printf("Use of SQUARE(). The value of SQUARE(3+2): %d", SQUARE(3+2));

}

Output

Use of sqr(). The value of sqr(3+2): 25

Use of SQUARE(). The value of SQUARE(3+2): 11

The function and macro, we want both will do the same task, but here we can see that the output are not same. The main reason is when we are passing 3 + 2 as function argument, it converts into 5, then calculate 5 \* 5 = 25. For macro it is doing 3 + 2 \* 3 + 2 = 3 + 6 + 2 = 11.

So the macros are not recommended for the following problems −

* There is no type checking
* Default to debug, as they cause simple replacement
* Macro don’t have the namespace. So if the macro is defined in one section, it can be used at another section.
* Macro increases the code length as it is added before the code while preprocessing.
* Macro does not check any compile time errors.