**Macros at a glance**

**Macro** is a fragment of code which has been given a name. Whenever the name is used, it is replaced by the contents of the **macro**. Two types of macros:

-object-like **macros** resemble data objects when used,

- function-like **macros** resemble function calls.

**Macros** can be used to make tasks less repetitive by representing a complicated sequence of keystrokes, mouse movements, commands, or other types of input. In computer **programming**, **macros** are a tool that allows a developer to re-use cod

The #define preprocessor directive creates symbolic constants. The symbolic constant is called a **macro** and the general form of the directive is

Macro tutorial:

https://bogotobogo.com/cplusplus/preprocessor\_macro.php

The formal syntax of a macro is:

*#define name(dummy1[,dummy2][,...]) token string*

*A* ***token*** *is* ***the*** *smallest element of a* ***C++*** *program that is meaningful to* ***the*** *compiler.* ***The C++*** *parser recognizes these kinds of* ***tokens****: identifiers, keywords, literals, operators, punctuators, and other separators. A stream of these* ***tokens*** *makes up a translation unit.* ***Tokens*** *are usually separated by white space*

Macro evils

<https://arne-mertz.de/2019/03/macro-evil/>

Writing C/C++ Macros: Rules, Tricks and Hint

<http://www.ebyte.it/library/codesnippets/WritingCppMacros.html>

https://en.cppreference.com/w/cpp/preprocessor/replace

The preprocessor supports text macro replacement. Function-like text macro replacement is also supported.

**Syntax**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | | |
| **#define** *identifier* *replacement-list*(optional) | (1) |  |  |  |  |  |  |  |  |
|  | | | | | | | | | |
| **#define** *identifier***(** *parameters* **)** *replacement-list*(optional) | (2) |  |  |  |  |  |  |  |  |
|  | | | | | | | | | |
| **#define** *identifier***(** *parameters***, ... )** *replacement-list*(optional) | (3) | (since C++11) |  |  |  |  |  |  |  |
|  | | | | | | | | | |
| **#define** *identifier***( ... )** *replacement-list*(optional) | (4) | (since C++11) |  |  |  |  |  |  |  |
|  | | | | | | | | | |
| **#undef** *identifier* | (5) |  |  |  |  |  |  |  |  |

Example1:

#define macro-name replacement-text

When this line appears in a file, all subsequent occurrences of macro in that file will be replaced by replacement-text before the program is compiled. For example −

#include <iostream>

using namespace std;

#define PI 3.14159

int main () {

cout << "Value of PI :" << PI << endl;

return 0;

}

Example2:

**#define** pred(x) ((x)-1)

What this macro expands to depends on what [argument](https://en.wikipedia.org/wiki/Argument_(computer_science)) *x* is passed to it. Here are some possible expansions:

pred(2) → ((2) -1)

pred(y+2) → ((y+2) -1)

pred(f(5)) → ((f(5))-1)

Counter-example:

#define sqr(x) (x\*x)

sqr(1+2) = sqr(1+2\*1+2)= 5; //macro is a text replacement

however:

# define sqr(x) ((x)\*(x))

sqr(1+2)= ((1+2)\*(1+2))= 9;

<http://www.cplusplus.com/doc/tutorial/preprocessor/>

When the preprocessor encounters this directive, it replaces any occurrence of identifier in the rest of the code by replacement. This replacement can be an expression, a statement, a block or simply anything. The preprocessor does not understand C++ proper, it simply replaces any occurrence of identifier by replacement.

|  |  |  |
| --- | --- | --- |
| 1 2 3 | #define TABLE\_SIZE 100  int table1[TABLE\_SIZE];  int table2[TABLE\_SIZE]; |  |

After the preprocessor has replaced TABLE\_SIZE, the code becomes equivalent to:

|  |  |  |
| --- | --- | --- |
| 1 2 | int table1[100];  int table2[100]; |  |

#define can work also with parameters to define function macros:

|  |  |  |
| --- | --- | --- |
|  | #define getmax(a,b) a>b?a:b |  |

This would replace any occurrence of getmax followed by two arguments by the replacement expression, but also replacing each argument by its identifier, exactly as you would expect if it was a function:

|  |  |  |  |
| --- | --- | --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 | // function macro  #include <iostream>  using namespace std;  #define getmax(a,b) ((a)>(b)?(a):(b))  int main()  {  int x=5, y;  y= getmax(x,2);  cout << y << endl;  cout << getmax(7,x) << endl;  return 0;  } | 5  7 | [Edit & Run](http://www.cplusplus.com/doc/tutorial/preprocessor/) |