define jabberwocky

AI generated article from Bing

---

c++ - What does ## in a #define mean? - Stack Overflow

In other words, when the compiler starts building your code, no #define statements or anything like that is left. A good way to understand what the preprocessor does to your code is to get hold of the preprocessed output and look at it.

How can I use #if inside #define in the C preprocessor?

How can I use #if inside #define in the C preprocessor? Asked 15 years, 8 months ago Modified 10 months ago Viewed 51k times

c - "static const" vs "#define" vs "enum" - Stack Overflow

Which one is better to use among the below statements in C? static const int var = 5; or #define var 5 or enum { var = 5 };

What is the purpose of the #define directive in C++?

0 in C or C++ #define allows you to create preprocessor Macros. In the normal C or C++ build process the first thing that happens is that the PreProcessor runs, the preprocessor looks though the source files for preprocessor directives like #define or #include and then performs simple operations with them.

How can I define a define in C? - Stack Overflow

The question is if users can define new macros in a macro, not if they can use macros in macros.

c++ - Why use #define instead of a variable - Stack Overflow

What is the point of #define in C++? I've only seen examples where it's used in place of a "magic number" but I don't see the point in just giving that value to a variable instead.

Why do most C developers use define instead of const?

#define simply substitutes a name with its value. Furthermore, a #define 'd constant may be used in the preprocessor: you can use it with #ifdef to do conditional compilation based on its value, or use the stringizing operator # to get a string with its value.

Good Programming Practices for Macro Definitions

(#define) in C

For example, never define a macro like this: #define DANGER 60 + 2 This can potentially be dangerous when we do an operation like this: int wrong_value = DANGER * 2; // Expecting 124 Instead, def...

What's the difference in practice between inline and #define?

2 Macros (created with #define) are always replaced as written, and can have double-evaluation problems. inline on the other hand, is purely advisory - the compiler is free to ignore it. Under the C99 standard, an inline function can also have external linkage, creating a function definition which can be linked against.

c - #Define VS Variable - Stack Overflow

#define WIDTH 10 is a preprocessor directive that allows you to specify a name (WIDTH) and its replacement text (10). The preprocessor parses the source file and each occurrence of the name is replaced by its associated text.