

Forum:[Hack Forums](#)

Thread: [Help with #define](#)

SGUdyr - November 15, 2016, 1:35 pm

Picked up C recently, had a little problem with #define

I wanted to calculate circle's circumference, and I wanted to use #define to define Pi's value.

```
Code #include <stdio.h>
int main()
{
    int r; float C;
    #define Pi 3.14
    printf("r: ");
    scanf("%d", r);
    C=(2*r)*Pi;
    printf("%f", C);
    getch();
}
```

Program obviously crashes when I enter r, so what to do here? Ty ?

Yuuki - November 15, 2016, 1:46 pm

Seems to be working fine when I compile it (with some changes).

Can you try using my changes and see if it works? The problem may not be related to #define, but rather your scanf()

SGUdyr - November 15, 2016, 1:48 pm

My bad, forgot the & in &r.

Appreciate it.

Sajuuk - November 15, 2016, 2:52 pm

I don't know if anyone else has been told this, but according to my professors "C-style variable declarations, where they're all at the top of the code right below where the main function opens, make the code harder to read and makes it more difficult to trace errors back to their variables". I have a hard time understanding how that reference is even valid, as most compilers throw the line and variable name in which the error occurred (as well as the character in the line where the error begins).

As an alternative, we are instructed to initialize the variable right before it is to be used in the first function that uses it. I never have done that, and most likely will never do such a thing, it just seems ridiculous even when dealing with larger projects. The developer will know what variables initialize to what when he chooses to initialize them. Arithmetic operations later on in the program are a whole different story, but I don't know if I can relate this concept to defining preprocessor macros. I've never seen anyone define any keyword within the main function. I'm not sure if a keyword defined within a function acts the same as keywords up at the top with the include macros, or if it is limited to

that function in which it was defined.

Hash G. - November 15, 2016, 4:18 pm

Well I know you don't have this problem anymore, but just a general recommendation:

Always always put `#define` in a `.h` file and if you can't, put `#define` at the top of your `.c` file.
Macro's names always in uppercase, so it should be `PI` instead of `Pi`.

Sajuuk - November 15, 2016, 4:21 pm

(Nov 15, 2016 05:18 PM)Hash G. Wrote: [\[CITE\]](#) Well I know you don't have this problem anymore, but just a general recommendation:

Always always put `#define` in a `.h` file and if you can't, put `#define` at the top of your `.c` file.
Macro's names always in uppercase, so it should be `PI` instead of `Pi`.
I'd only say put defines in a header when a header is actually needed, which in most if not all large projects there will be at least one if not numerous custom headers. And yes, terms that are `#define`'d are constants, so (technically speaking) they should be uppercase.

RDCA - November 15, 2016, 7:36 pm

Quote:Always always put `#define` in a `.h` file and if you can't, put `#define` at the top of your `.c` file.

Eh, this isn't something you should say always about.

Sajuuk - November 15, 2016, 7:45 pm

(Nov 15, 2016 08:36 PM)RDCA Wrote: [\[CITE\]](#) Eh, this isn't something you should say always about.
With smaller projects it's fine so long as you're gonna end up using the same defines in one program. However when there are a massive number of `#define`'s required to compile such an app, not to mention the individual files you might be subjected to split the entire source into, in those cases it is essential to define everything in a separate header.

RDCA - November 15, 2016, 7:52 pm

(Nov 15, 2016 08:45 PM)Sajuuk Wrote: [\[CITE\]](#) With smaller projects it's fine so long as you're gonna end up using the same defines in one program. However when there are a massive number of `#define`'s required to compile such an app, not to mention the individual files you might be subjected to split the entire source into, in those cases it is essential to define everything in a separate header.
How big a project is doesn't really affect whether you should put something in a header file or not.

Sajuuk - November 15, 2016, 7:55 pm

(Nov 15, 2016 08:52 PM)RDCA Wrote: [\[CITE\]](#) How big a project is doesn't really affect whether you should put something in a header file or not.

"As big of a project" by what's supposed to be in headers. Definitions, universal includes, etc. I didn't mean to imply that I might be using the entire project's size, source code and all, when saying that. Just standard header stuff. How the fuck did all of this pop out of me noting a `#define` within his main function??

0xDEAD10CC - November 15, 2016, 9:40 pm

`system()` does not come from the `<Windows.h>` header either -- nobody mentioned this. It comes from the `<stdlib.h>` header (`<cstdlib>`), and calling `system("pause")` is undefined behavior. There are better platform-independent ways of waiting for user-input to synchronously halt execution. Therefore, the `<Windows.h>` header here is redundant.

Sajuuk - November 15, 2016, 9:50 pm

(Nov 15, 2016 10:40 PM)0xDEAD10CC Wrote: [\[CITE\]](#) `system()` does not come from the

<Windows.h> header either -- nobody mentioned this. It comes from the <stdlib.h> header (<cstdlib>), and calling system("pause") is undefined behavior. There are better platform-independent ways of waiting for user-input to synchronously halt execution. Therefore, the <Windows.h> header here is redundant.

I actually had to scroll back up to find the screenshot that actually had a system() call. Yuuki's code ran either because he compiled it with a different source (other than the one shown) before he took the screen or because his copy of <windows.h> includes <stdlib.h> for some reason (which makes no sense at all). Or, if not one of those two, he did include <stdlib.h> but the screenshot cut it off.

Infinity. - November 16, 2016, 2:41 am

(Nov 15, 2016 02:46 PM)Yuuki Wrote: [\[CITE\]](#)

Seems to be working fine when I compile it (with some changes).

Can you try using my changes and see if it works? The problem may not be related to #define, but rather your scanf()

Only MSVC picks up 'scanf' and doesn't accept it.

Codeblocks still accepts scanf. If the problem was scanf then OP wouldn't be able to compile the problem given that he is using VS.

(Nov 15, 2016 05:18 PM)Hash G. Wrote: [\[CITE\]](#) Well I know you don't have this problem anymore, but just a general recommendation:

Always always put #define in a .h file and if you can't, put #define at the top of your .c file.

Only if you do #pragma once for your includes. I maybe wrong though.

0xDEAD10CC - November 16, 2016, 10:04 am

(Nov 16, 2016 03:41 AM)Infinity. Wrote: [\[CITE\]](#) Only MSVC picks up 'scanf' and doesn't accept it.

Codeblocks still accepts scanf. If the problem was scanf then OP wouldn't be able to compile the problem given that he is using VS.

Only if you do #pragma once for your includes. I maybe wrong though.

What are you talking about?? :S

VS does "accept" scanf(), it has nothing to do with the IDE or the compiler really, just that Microsoft's compiler will provide warnings about those functions unless you disable them. This doesn't mean that it won't compile. Don't spread around lies like this...

And #pragma once also has nothing to do with #define's.

Infinity. - November 16, 2016, 11:25 am

(Nov 16, 2016 11:04 AM)0xDEAD10CC Wrote: [\[CITE\]](#) What are you talking about?? :S

VS does "accept" scanf(), it has nothing to do with the IDE or the compiler really, just that Microsoft's compiler will provide warnings about those functions unless you disable them. This doesn't mean that it won't compile. Don't spread around lies like this...

And #pragma once also has nothing to do with #define's.

oops sorry, yes it gives warning, i wasn't aware that you can disable them, i don't know much about VS, I am new. Show me that it compiles without disabling warnings.

Sajuuk - November 16, 2016, 11:29 am

(Nov 16, 2016 12:25 PM)Infinity. Wrote: [\[CITE\]](#) oops sorry, yes it gives warning, i wasn't aware that you can disable them, i don't know much about VS, I am new. Show me that it compiles without disabling warnings.

With GCC, my code compiles regardless of what warnings the compiler throws. It's just a warning, not an error.

0xDEAD10CC - November 16, 2016, 11:30 am

(Nov 16, 2016 12:25 PM)Infinity. Wrote: [\[CITE\]](#) oops sorry, yes it gives warning, i wasn't aware that you can disable them, i don't know much about VS, I am new. Show me that it compiles without disabling warnings.

You need to consider the difference between compiler warnings and compiler errors. You can compile code with warnings, but you can't compile code with errors...

Sajuuk - November 16, 2016, 11:31 am

(Nov 16, 2016 12:30 PM)0xDEAD10CC Wrote: [\[CITE\]](#) You need to consider the difference between compiler warnings and compiler errors. You can compile code with warnings, but you can't compile code with errors...

Which is what I was hinting to. However when you do get warnings from the compiler, note them and their possible effects.

Infinity. - November 16, 2016, 11:37 am

(Nov 16, 2016 12:29 PM)Sajuuk Wrote: [\[CITE\]](#) With GCC, my code compiles regardless of what warnings the compiler throws. It's just a warning, not an error.

I was specifically talking about cl.exe. Yes it compiles using codeblocks and MinGW.

(Nov 16, 2016 12:30 PM)0xDEAD10CC Wrote: [\[CITE\]](#) You need to consider the difference between compiler warnings and compiler errors. You can compile code with warnings, but you can't compile code with errors...

Ok thanks for correcting me.

Sajuuk - November 16, 2016, 11:39 am

(Nov 16, 2016 12:37 PM)Infinity. Wrote: [\[CITE\]](#) I was specifically talking about cl.exe. Yes it compiles using codeblocks and MinGW.

I understand that. The Microsoft compiler I have does the same thing as GCC with warnings. It compiles the source, unlike when it throws an error and does not compile the source.