C Preprocessors (Answers)

Sayak Haldar

IIEST, Shibpur

1. Answer is a) The predefined location then the current directory

2. Answer is b) C:COMPILERINCLUDE

Explanation: The order of searching by compilers when using #include

C:COMPILERINCLUDE->S:SOURCEHEADERS-> Current directory where program is saved

3. Answer is b) The user-defined library file will be selected

4. Answer is b) When former (#include<somelibrary.h>) is used, current directory is searched and when latter is used, standard library is searched

5. Answer is a) Yes

Explanation: Though implementing functions within header files is a bad practice it can be done in c

More Explanation:

Why do we consider defining functions within header files as a bad practice in c?

In C, if you define a function in a header file, then that function will appear in each module that is compiled that includes that header file, and a public symbol will be exported for the function. So if function additup is defined in header.h, and foo.c and bar.c both include header.h, then foo.o and bar.o will both include copies of additup.

When you go to link those two object files together, the linker will see that the symbol additup is defined more than once, and won't allow it.

More Explanation:

Difference between header files and libraries:

Generally, a header file notifies the *compiler* of certain things (mostly their existence or declarations) so that the compiler can correctly build a single translation unit (such as a single C file).

A library file is the actual *executable code* that does the work as specified in that header file. This is linked in by the *linker* to provide the actual functionality (the _definitions rather than just the declarations).

Header Files are the files that are included at the top of any program. If we use any function inside a program, then the header file containing declaration or definition of that function ,has to be included.Like printf() is defined in stdio.h.So, we must include it (by #include in order to use printf().

Library Files are the files which the compiler uses in order to define the functions which have been used in the program and had been declared inside the header file.Like, printf() has its complete definition ,like how it will work etc. in an I/O library! So, the compiler uses that library to get the machine code for printf.

Some of the differences between c header and library files (not user defined, but the real ones provided by c compiler):

- Header files generally contain the declarations of the function while libraries contain the definitions of the function.
- Header files are TEXT files while library files are BINARY. This means, we can read and modify the header file but not the library.
- Header file is in C language while the library is in machine language!
- Header file has to be included by the programmer while the compiler automatically relates the library file(s) with the program!

More Explanation:

Do's and Dont's to follow for creating a user-defined header files:

DO create one .h header file for each "module" of the system. A module may comprise one or more compilation units (e.g., .c or .asm source code files). But it should implement just one aspect of the system. Examples of well-chosen modules are: a device driver for an A/D converter; a communication protocol, such as FTP; and an alarm manager that is solely responsible for logging error conditions and alerting the user of the active errors.

DO include in the header file all of the function prototypes for the public interface of the module it describes. For example a header file adc.h might contain function prototypes for adc_init(), adc_select_input(), and adc_read().

DON'T include in the header file any other function or macro that may lie inside the module source code. It is desirable to hide these internal "helper" functions inside the implementation. If it's not called from any other module, hide it! (If your module spans several compilation units that need to share a helper function, then create a separate header file just for this purpose.) Module A should only call Module B through the public interface defined in moduleb.h.

DON'T include any executable lines of code in a header file, including variable declarations. But note it is necessary to make an exception for the bodies of some inline functions.

DON'T expose any variable in a header file, as is too often done by way of the 'extern' keyword. Proper encapsulation of a module requires data hiding: any and all internal state data in private variables inside the .c source code files. Whenever possible these variables should also be declared with keyword ‘static’ to enlist the linker’s help in hiding them.

DON’T expose the internal format of any module-specific data structure passed to or returned from one or more of the module’s interface functions. That is to say there should be no “struct { … } foo;” code in any header file. If you do have a type you need to pass in and out of your module, so client modules can create instances of it, you can simply “typedef struct foo moduleb_type” in the header file. Client modules should never know, and this way cannot know, the internal format of the struct.

6) Answer is d) It depends upon the compiler.

Explanation: In some compiler It will give compilation error while in some other compiler it will not cause anything.

7) Answer is a) 23

8) Answer is a) The preprocessor treats it as a user-defined file

9) Answer is b) The preprocessor treats it as a system-defined file

10) Answer is a) hello

11) Answer is a) #include

Explanation:

In #include C_IO_HEADER, C_IO_HEADER will be replaced by nothing

12) Answer is b) Compilation error if no file named “printf” is present in the current directory

a) hello if a file named “printf” is present in the current directory

13) Answer is c) Conditional Compilation

Explanation: A useful facility provided by the he preprocessor is **conditional compilation**; i.e. the selection of lines of source code to be compiled and those to be ignored. Conditional compilation can be used for many purposes including its use with debug statements.

Example of conditional compilation derivatives in c: ifdef,ifndef, if, elif, else

14) Answer is a) Preprocessor directive

15) Answer is a) True

Explanation: pragma: this preprocessor directive is a compiler specific feature.

More Explanation:

#pragma compiler specific extension
The pragma directive is used to access compiler-specific preprocessor extensions. A common use of #pragma is the #pragma once directive, which asks the compiler to include a header file only a single time, no matter how many times it has been imported:

```
#pragma once
// header file code
```

In this example, using #pragma once is equivalent to an include guard that prevents the file from being processed multiple times.

```
#ifndef _FILE_NAME_H_
#define _FILE_NAME_H_

/* code */

#endif // #ifndef _FILE_NAME_H_
```

#pragma once is available on many major compilers, including Clang, GCC, the Intel C++ compiler and MSVC.

The #pragma directive can also be used for other compiler-specific purposes. #pragma is commonly used to suppress warnings. For example, in MSVC

```
#pragma warning (disable : 4018 )
```

Can be used to disable warning 4018, warning of signed/unsigned mismatch. While you should be reluctant to suppress warnings sometimes it is necessary.

For more uses of the #pragma directive, consult your compiler's documentation.

16) Answer is a) In main.

Explanation: Preprocessor just replaces whatever is given compiler then checks for error at

the replaced part of the code. Here it is not replaced anywhere.

17) Answer is d) Line Control

18) Answer is d) They can include all types of files.

19) Answer is a) Preprocessor is a program that processes its input data to produce output that is used as input to another program

20) Answer is d) All of the mentioned.

21) Answer is b) It could be mentioned before any printf or scan f

Explanation: We generally write it before main to improve the readability of the program

22) Answer is c) Running a function at exiting the program

23) Answer is b) 1.2

Explanation: Since, next macro is not defined, line 6 and 7 of the program will not be executed

24) Answer is a) #

25) Answer is c) Both a & b

26) Answer is d) All of the mentioned.

27) Answer is a) The file is searched for in the standard compiler include paths

Explanation: #include might check the current directory but #include<filename.h> will never check the existence of the filename.h in the current directory.

28) Answer is c) Compile time error

29) Answer is d) m##n

Since, foo(k,l) will simply replaced by "m##n" after preprocesssing in the c code. So, m##n will be printed as a string

30) Answer is a) kl

31) Answer is c) -8

Explanation: foo(i + j, 3) at line 6, will be replaced by x+y/3+x+y in the program with x=-6 and y=3

Now, / operator has highest precedance among the present operators in the expression

So, -6+3/3-6+3 will be evaluated as the following:

=-6+1-6+3

Now, we get an expression consists of only – and + operator. Both operator have same precedence. So, asssociativity will effect the evaluation of the expression. Now, associativity of + and – is from left to right. So,

=-5-6+3

=-11+3

=-8

32) Answer is b) -4

Explanation: Evaluate it yourself. I know you can do it

33) Answer is a) 100

34) Answer is a) -8 -4

Explanation: check answers of questions a) and b)

35) Answer is a) -8 -4

Explanation: foo(i+j,3) at line 7 will be replaced by i+j/3+i+j

Whereas, foo(i+j,3) at line 9 will invoke the the function foo

37) Answer is a) Data type is flexible

38) Answer is b) Compile time error due to ';' at the end of line 4

39) Answer is a) 37

40) Answer is b) 32

Explanation: Though const is a keyword in both c and c++, it will work fine

41) Answer is c) Compile time error

Explanation: Compile time error since max is used as an identifier at line 5, but max is undeclared

Compile time error due to absence of lvalue at the left side of assignment operator at line 5.

42) Answer is d) hi

43) Answer is b) 11

Explanation: A*B at line at line 6 will be replaced by 1+2*3+4=11

44) Answer is d) None of the mentioned.

45) The answer is a) No error, it will show the output 20.

Explanation: Since printf("%d\n", var will simply become printf("%d\n",20); after preprocessing

46) The answer is b) World

Explanation: a needs to be a macro to execute line 7

47) Answer is d) Compile time error

Explanation: Compiler would say the following things:

a)Unterminated #ifdef

b)expected expression before ‘)’ token

printf("%d", Cprog); Since Cprog would expand to nothing thus causing the error

because %d would expect one integer type argument at printf.

Note: #ifdef must be terminated by #endif

48) Answer is d) elif

49) Answer is b) COLD.

50) Answer is c)

51) Answer is a)

Explanation: The rule is simple like normal if, else if and else.

52) The answer is d) All of the mentioned

53) The answer is a) True

54) Answer is d) All of the mentioned

55) Answer is a) Conditionally include source text if the previous #if, #ifdef, #ifndef, or #elif test fails.

56) Answer is b) Compile time error

Explanation: #if min Now, min is 0 so, the next statement which defines MAX macro would never be executed. Thus, MAX would be undeclared during its use in main.

Note: This is the basic difference between #ifdef and #if

If in line 3, #if MIN is replaced by #ifdef MIN then the program will not cause compilation error. It will print 10 0

57) Answer is a) 10 0

58) Answer is a) 10 0

Explanation:

Focus on

#if defined(MIN) + defined(MAX) at line 3

Now, defined will check if a macro is defined or not and returns 1 or 0 accordingly/respectively

Now, MIN macro is already defined and MAX macro is not defined

So, defined(MIN)+defined(MAX)=1+0

#if 1

So, line 4 will execute/will get a chance to be executed

59) Answer is b)

Explanation: Compile time error. Compiler would say, MAX is undeclared.

To understand it properly, check the last answer

60) Answer is a) 10 0

References:

1. http://www.sanfoundry.com/c-interview-questions-answers/
2. Different questions from http://stackoverflow.com/
3. http://embeddedgurus.com/barr-code/2010/11/what-belongs-in-a-c-h-header-file/
4. http://www.cprogramming.com/reference/preprocessor/