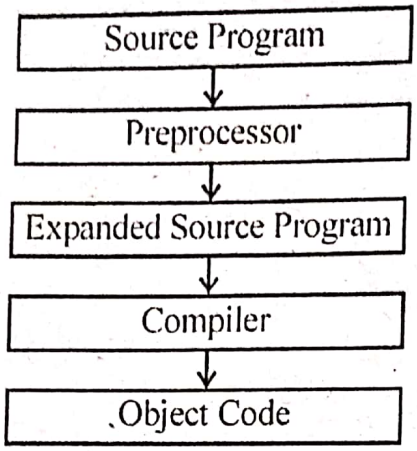
Q1. What is preprocessor?

Ans. The C preprocessor is a program that processes the source program before passing it to the compiler. It is used to define language within a language. It is also used for fast execution. The C preprocessor directly substitute the value or expression of a statements in source program so that the source program now becomes an expanded source program. That means it also used to expand the source program. After that the compiler compiles the program.



Q2. What is preprocessor in C? What is the difference between following two lines of code:

 # include <myfile.h>
 # include "myfile.h"

Ans. Preprocessor :

The preprocessor, as its name implies, is a program that process the source code before it passes through the compiler.

Preprocessor directives are placed in the source program before the main line: Before the source code passes through the compiler, it is examined by the preprocessor for any preprocessor directives.

 # include "myfile.h" : When the file name is included with in the double quotation marks the search for the file is made first in the current directory and then in the standard directory.

 # include < myfile.h > : In this case file is search only in the standard directories.

Q3. What are the features of C preprocessor?

Ans. The preprocessor offers several features called preprocessor directives. Each of these preprocessor directives begins with a # symbol. Normally, a control line appears in a single line with # as the first non-white space character and a new line as the terminating character. There is no semicolon to end any preprocessor facility.

Q4. List different types of directive?

Ans.

Directive Keyword	Meaning
#define	Defines a macro substitution.
#undef	Undefines a macro.
#include	Includes the specified files.
#ifdef	Tests whether a macro is defined.
#ifndef	Tests whether a macro is not defined.
#if	Tests an expression.
#else	Specifies alternatives if #if fails.
#endif	Marks the end of an #if block.
#elif	Specifies an if-else-if chain for multiple compilation options.
#error	Forces the compiler to stop compilation.
#line	Causes subsequent source code lines to be renumbered.
#pragma	Specifies various instructions to be given to the compiler.

Q5. What are the pros and cons of using a macro in place of a regular function?

Ans. Macro :

Macro substitution is a process where an identifier in a program is replaced by a predefined string composed of one or more tokens. The preprocessor accomplishes this task under the direction of # define statement. This statement is usually known as macro definition.

Its syntax : `#define identifier string`

Pros of using Macro :

1. It saves memory and time.
2. It enhance the speed of execution of program.
3. Reduce source code length.
4. Make the program more readable.
5. Any modification to instruction in macro reflects in every call.
6. No performance drawback by macros.

Cons of Using Macro :

1. Calling macro too many times in the code will increase the size of the code.
2. Simple operation can be performed which does not increase the size of the code.
3. Few operations are unpredictable.

Q6. What is macro substitution directive?

Ans. Macro expansion is a preprocessor statement, which is used to make program more readable and fast executable. Each preprocessor statement is defined with the word **define**. The word **define** begins with a # symbol and is not terminated by a semi-colon. It is normally written at the beginning of the source program. For example,

```
#define TRUE  1
#define FALSE 0
#define AND   &&
#define OR    ||
#define EQUALS ==
```

All preprocessor statements are handled by the compiler (or preprocessor) before the program is actually compiled. All preprocessor statements (i.e. starting with #) are processed first and the symbols,

which occur in the C program are replaced by their values. Once this substitution has taken place by the preprocessor, the program is then compiled.

Q7. What are file inclusion directive?

Ans. The C preprocessor define the concepts of file inclusion to include the content of one file into another file. Through this concept a very large program can be divided into several different files and each file contents a set of related functions. All the existing file which content you want in a new file or program are always included in the beginning of a new file by the following syntax:

```
#include "filename"
#include <filename>
```

The #include "filename" will search the file *filename* in the current directory as well as the specified list of directories as mentioned in the include search path that might have been set up.

The #include<filename> will search the file *filename* in only the specified list of directories as mentioned in the include search path that might have been set up.

Q8. What is conditional compilation directive?

Ans. The C preprocessor defines the concepts of conditional compilation to include a file or macro on some conditions. The C language support the following conditional compilation commands

```
#if
#ifdef
#else
#ifndef
#elif
#endif
```

Q9. What is # if statement?

Ans. The #if conditional statement is used to provide the selectional control structure that executes a set of statements based on a condition. The syntax for #if is

```
#if macro
    set of statements;
    #endif
    next statements
```

In this syntax if value of macro is non-zero then the compiler only compiles the set of statements that follow the #if directive otherwise the compiler skips the lines that follow until it encounters the matching #endif.

Q10. What is the meaning of # ifdef statement?

Ans. The meaning of the #ifdef is if the macro is defined then the compiler only compiles the set of statements that follow the #ifdef directive otherwise the compiler skips the lines that follow until it encounters the matching #endif. In this case the value of macro is either zero or non-zero does not matter.

Q11. What is the meaning of # ifndef statement?

Ans. The meaning of the #ifndef is if the macro is not defined then the compiler only compiles the set of statements that follow the #ifndef directive otherwise the compiler skips the lines that follow until it encounters the matching #endif. In this case the value of macro is either zero or non-zero does not matter.

Q12. What is # ifdef and #else statement?

Ans. The #ifdef #else conditional statement is used to select any one set of statements between two sets of statements. The syntax for **#ifdef #else** statements is executes a set of statements based on a condition. The syntax for **#ifdef #else** statements is

```
#ifdef macro
        First set of statements;
#else
        Second set of statements;
#endif
        next statements
```

In this syntax if the macro is defined then the compiler compiles the first set of statements that follow the #ifdef directive otherwise the compiler compiles the second set of statements that follow #else statement. In this case the value of macro is either zero or non zero does not matter.

Q13. What is nested conditional compilation directives?

Ans. The nested conditional compilation directives statements are used for the multi-way decision based on several conditions. The most general way of doing this is by using the following syntax.

```
#ifdef macro
        First set of statements;
#elif
        Second set of statements;
```