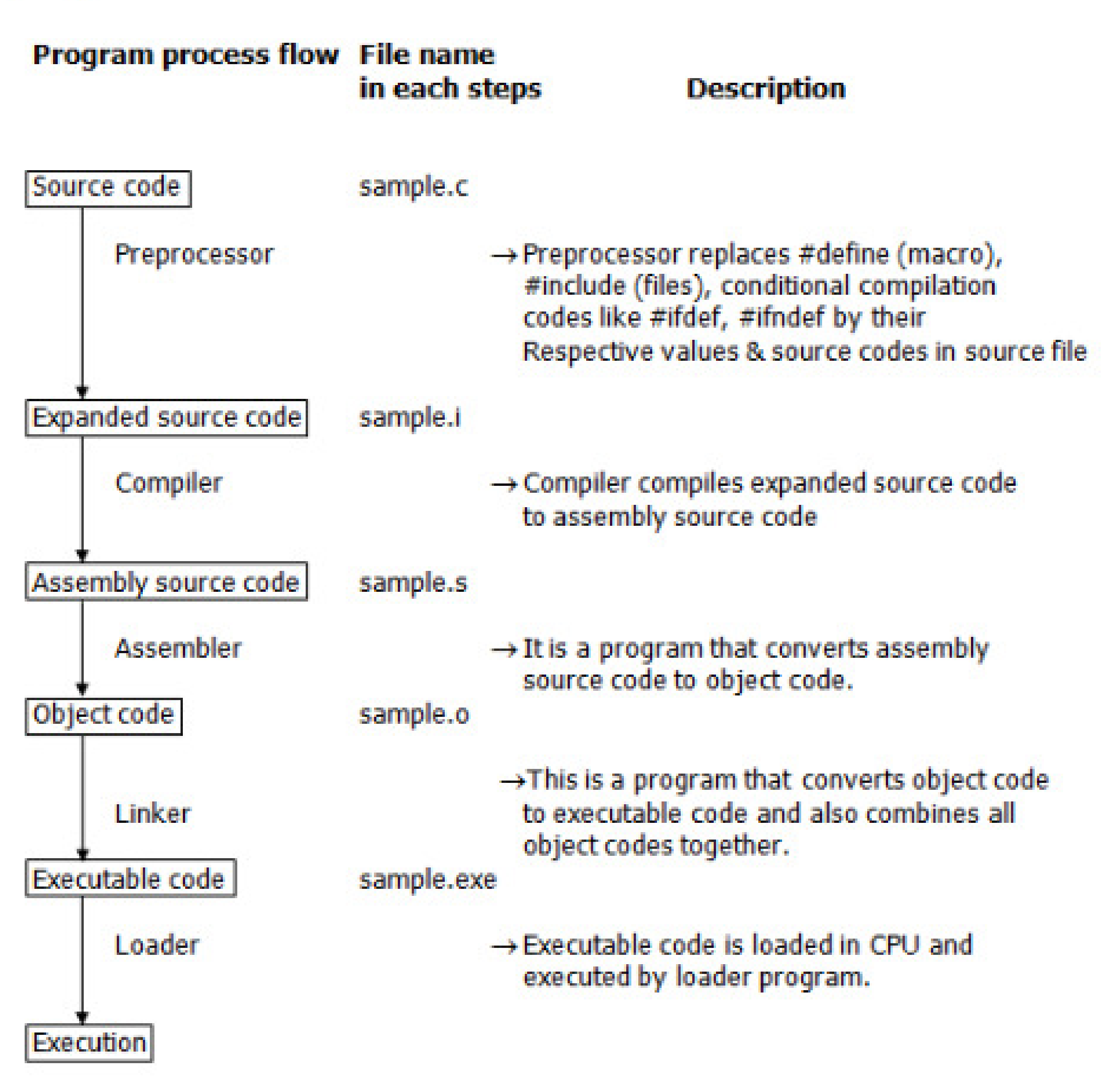
C Preprocessor directives:

- Before a C program is compiled in a compiler, source code is processed by a program called preprocessor. This process is called preprocessing.

- Commands used in preprocessor are called preprocessor directives and they begin with "#" symbol.
- Below is the list of preprocessor directives that C language offers.

S.no	Preprocessor	Syntax	Description
1	Macro	#define	This macro defines constant value and can be any of the basic data types.
2	Header file inclusion	#include <file_name>	The source code of the file "file_name" is included in the main program at the specified place
3	Conditional compilation	#ifdef, #endif, #if, #else, #ifndef	Set of commands are included or excluded in source program before compilation with respect to the condition
4	Other directives	#undef, #pragma	#undef is used to undefine a defined macro variable. #Pragma is used to call a function before and after main function in a C program

A program in C language involves into different processes. Below diagram will help you to understand all the processes that a C program comes across.


Program process flow
File name in each steps
Description
Source code
sample.c
Preprocessor
→Preprocessor replaces #define (macro), #include (files), conditional compilation codes like #ifdef, #ifndef by their Respective values & source codes in source file
Expanded source code
sample.i
Compiler
→Compiler compiles expanded source code to assembly source code
Assembly source code
sample.s
Assembler
→It is a program that converts assembly source code to object code.
Object code
sample.o
Linker
→This is a program that converts object code to executable code and also combines all object codes together.
Executable code
sample.exe
Loader
→Executable code is loaded in CPU and executed by loader program.
Execution

Example program for #define, #include preprocessors in C:

- #define - This macro defines constant value and can be any of the basic data types.
- #include <file_name> - The source code of the file "file_name" is included in the main C program where "#include <file_name>" is mentioned.

```
#include <stdio.h>

#define height 100
#define number 3.14
#define letter 'A'
#define letter_sequence "ABC"
#define backslash_char '\?'

void main()
{
   printf("value of height    : %d \n", height );
   printf("value of number : %f \n", number );
   printf("value of letter : %c \n", letter );
   printf("value of letter_sequence : %s \n", letter_sequence);
   printf("value of backslash_char  : %c \n", backslash_char);

}
```

Output:

```
value of height : 100
value of number : 3.140000
value of letter : A
value of letter_sequence : ABC
value of backslash_char : ?
```

Example program for conditional compilation directives:

a) Example program for #ifdef, #else and #endif in C:

- "#ifdef" directive checks whether particular macro is defined or not. If it is defined, "If" clause statements are included in source file.
- Otherwise, "else" clause statements are included in source file for compilation and execution.

```
#include <stdio.h>
#define RAJU 100

int main()
{
   #ifdef RAJU
   printf("RAJU is defined. So, this line will be added in " \
          "this C file\n");
   #else
   printf("RAJU is not defined\n");
   #endif
   return 0;
}
```

Output:

```
RAJU is defined. So, this line will be added in this C file
```

b) Example program for #ifndef and #endif in C:

- #ifndef exactly acts as reverse as #ifdef directive. If particular macro is not defined, "If" clause statements are included in source file.
- Otherwise, else clause statements are included in source file for compilation and execution.

```
#include <stdio.h>
#define RAJU 100
int main()
{
   #ifndef SELVA
   {
      printf("SELVA is not defined. So, now we are going to " \
             "define here\n");
      #define SELVA 300
   }
   #else
   printf("SELVA is already defined in the program”);

   #endif
   return 0;

}
```

Output:

```
SELVA is not defined. So, now we are going to define here
```

c) Example program for #if, #else and #endif in C:

- "If" clause statement is included in source file if given condition is true.
- Otherwise, else clause statement is included in source file for compilation and execution.

```
#include <stdio.h>
#define a 100
int main()
{
   #if (a==100)
   printf("This line will be added in this C file since " \
          "a \= 100\n");
   #else
   printf("This line will be added in this C file since " \
          "a is not equal to 100\n");
   #endif
   return 0;
}
```

Output:

```
This line will be added in this C file since a = 100
```

Example program for undef in C:

This directive undefines existing macro in the program.

```
#include <stdio.h>

#define height 100
void main()
{
   printf("First defined value for height    : %d\n",height);
   #undef height          // undefining variable
   #define height 600     // redefining the same for new value
   printf("value of height after undef \& redefine:%d",height);
}
```

Output:

```
First defined value for height : 100
value of height after undef & redefine : 600
```

Example program for pragma in C:

Pragma is used to call a function before and after main function in a C program.

```
#include <stdio.h>

void function1( );
void function2( );

#pragma startup function1
#pragma exit function2

int main( )
{
   printf ( "\n Now we are in main function" ) ;
   return 0;
}

void function1( )
{
   printf("\nFunction1 is called before main function call");
}

void function2( )
{
   printf ( "\nFunction2 is called just before end of " \
            "main function" ) ;"
}
```

Output:

```
Function1 is called before main function call
Now we are in main function
Function2 is called just before end of main function
```

More on pragma directive in C:

S.no	Pragma command	description
1	#Pragma startup <function_name_1>	This directive executes function named "function_name_1" before
2	#Pragma exit <function_name_2>	This directive executes function named "function_name_2" just before termination of the program.
3	#pragma warn – rvl	If function doesn't return a value, then warnings are suppressed by this directive while compiling.
4	#pragma warn – par	If function doesn't use passed function parameter , then warnings are suppressed
5	#pragma warn – rch	If a non reachable code is written inside a program, such warnings are suppressed by this directive.