

C_146 \Rightarrow Introduction to Pre-processor in C

Preprocessor Directive (#include)

* Preprocessor is a program; it is not a part of compiler.

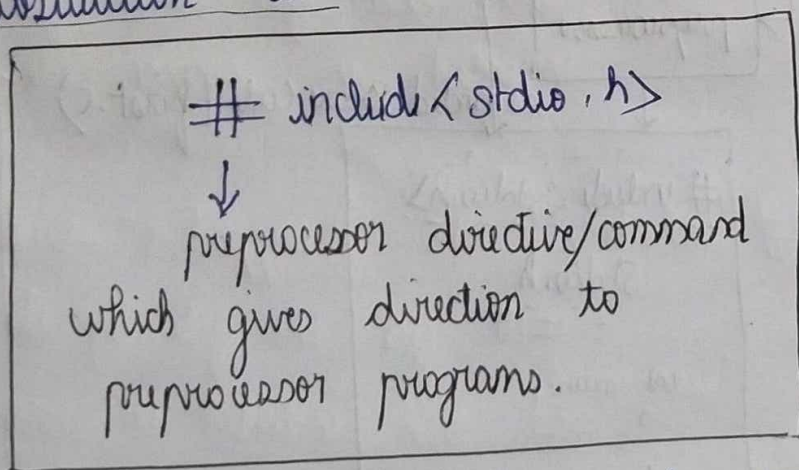
* But it is a step in compilation process.

* The function of preprocessor program comes before the compilation process for any (.c) files.

* When you compile any C file (filename.c) by clicking compile button; then before compiling process the function of preprocessor takes place.

* So preprocessing is just a step in compilation process but it is not a part of compilation process.


* Preprocessor is also called as text substitution tool.



* '#' will be understood only by preprocessor but compiler do not understand.

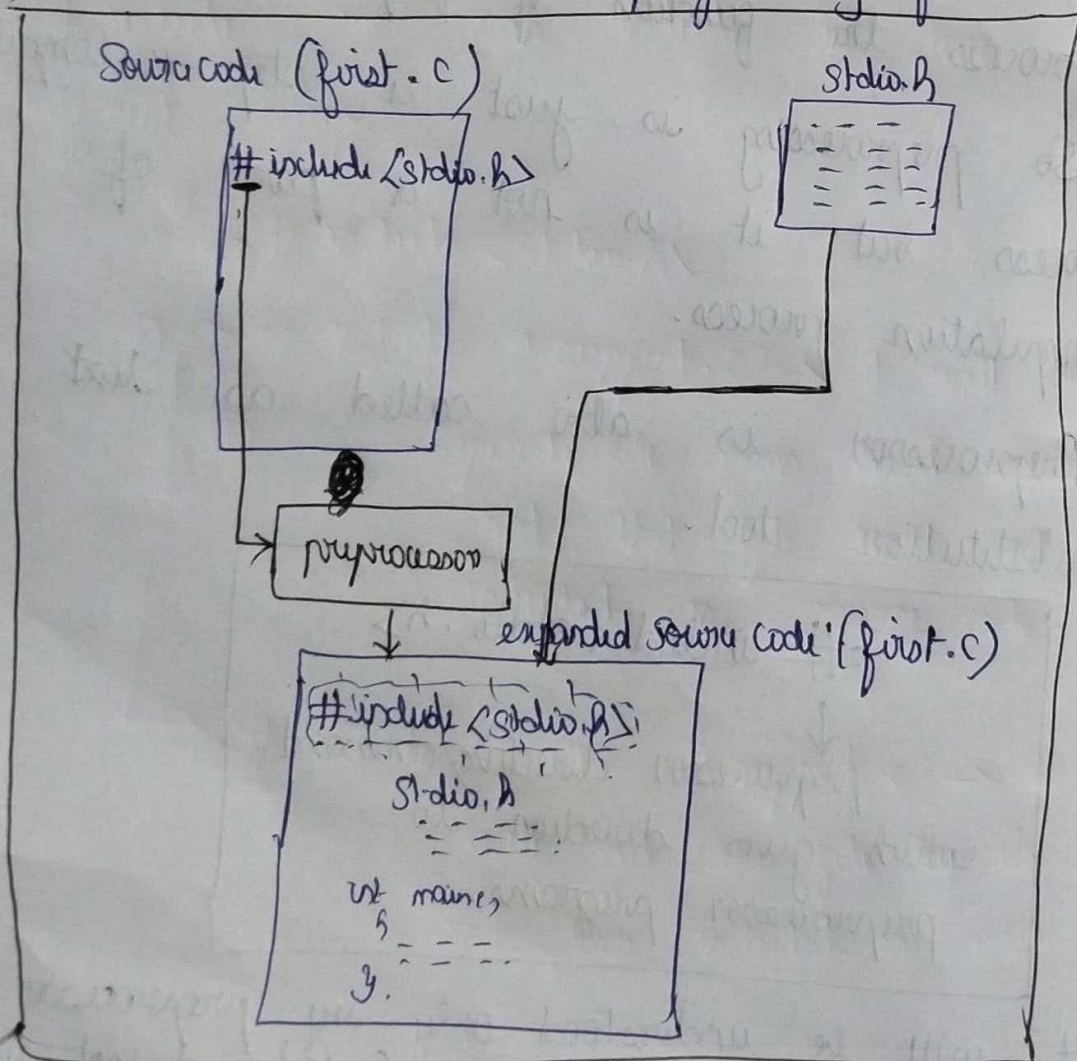
* So before compilation process the action of preprocessor takes place.

* So what will this preprocessor do? It will expand our source code (first.c) by including the (stdio.h) file to our source file.

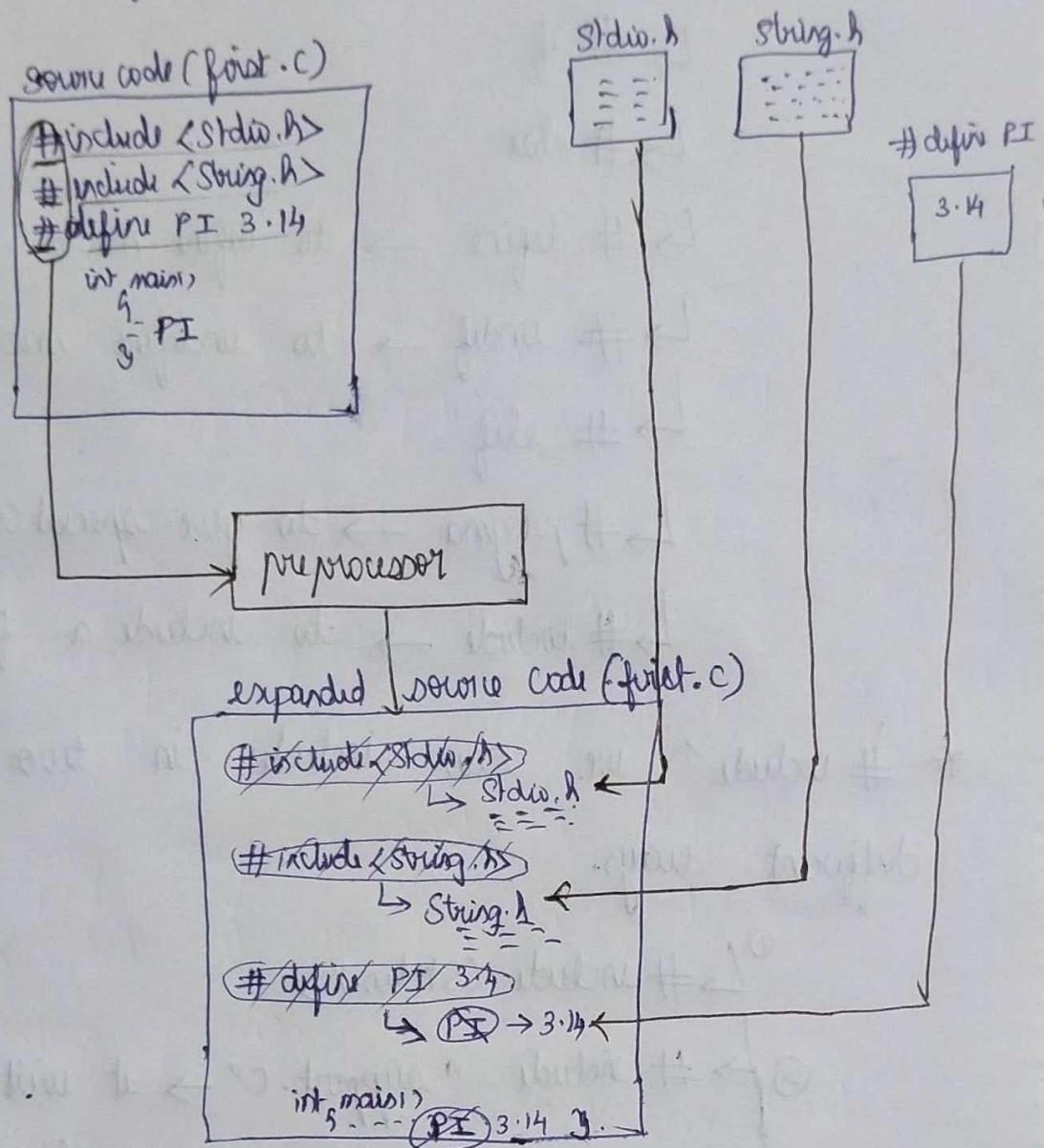
→ Eg:  include <stdio.h>

preprocessor gives direction to preprocessor which will expand the source code by including stdio.h file into the source code.

→ Source Code is our C program eg:- first.c.



eg. #include <stdio.h>
 #include <string.h>
 #define PI 3.14 → macro substitution.



* So the preprocessor program function will be done before compilation and the compiler will compile only the expanded source code and hence we tell preprocessor as text substitution tool which substitutes the '#' with files like `stdio.h`, `string.h` etc.

* There are different types of preprocessor directives like given below;

↳ #if

↳ #else

↳ #define → to define macro

↳ #undef → to undefine macro

↳ #elif

↳ #pragma → to give special commands

↳ #include → to include a file

* '#include' we can include in two different ways.

① ↳ #include <stdio.h>

② ↳ #include "support.c" → it will get

① Separate file from local directory and that file will get included in source code.

↳ #include "C:\ ---- \support.c" → it will get the file by its path and then included in source code.

The screenshot shows the Code::Blocks IDE with the file `main.c` open. The code includes `<stdio.h>` and `<stdlib.h>`. It defines a function `display()` and a `main` function that calls `display()` and returns 0. A comment explains that the `support.c` file is located outside the project folder and must be included using its full path. The status bar at the bottom indicates the file path: `D:\1. C C++\NOTEBOOK\C LANGUAGE\C Jennys Lectures\PART 11_JENNY'S LECTURE_MISCELLANEOUS TOPICS\47_Preprocess Directives in C - Code::Blocks 20.03`.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 /** 47 - Preprocessor directives in C **/
5
6 #include "D:\1. C C++\NOTEBOOK\C LANGUAGE\C Jennys Lectures\PART 11_JENNY'S LECTURE_MISCELLANEOUS TOPICS\47_Preprocess Directives in C - Code::Blocks 20.03\support.c"
7
8 int x=10;
9 extern void display();
10 int main()
11 {
12     display();
13     return 0;
14 }
```

The screenshot shows the Code::Blocks IDE with the file `support.c` open. The code includes `<stdio.h>` and `<stdlib.h>`. It defines a function `display()` that prints "Hello from support file" and the value of `x`. The status bar at the bottom indicates the file path: `D:\1. C C++\NOTEBOOK\C LANGUAGE\C Jennys Lectures\PART 11_JENNY'S LECTURE_MISCELLANEOUS TOPICS\47_Preprocess Directives in C - Code::Blocks 20.03`.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 /** 43 - extern Storage Classes in C **/
5 /** support.c file **/
6
7 void display()
8 {
9     extern int x;
10    printf("Hello from support file\n");
11    printf("%d\n",x);
12 }
13
```

The screenshot shows the Windows command prompt with the output of the program. It displays "Hello from support file", the value of `x` (10), and the message "Process returned 0 (0x0) execution time : 0.053 s".

```
"D:\1. C C++\NOTEBOOK\C LANGUAGE\C Jennys Lectures\PART 11_JENNY'S LECTURE_MISCELLANEOUS TOPICS\47_Preprocess Directives in C - Code::Blocks 20.03"
Hello from support file
10
Process returned 0 (0x0)   execution time : 0.053 s
Press any key to continue.
```