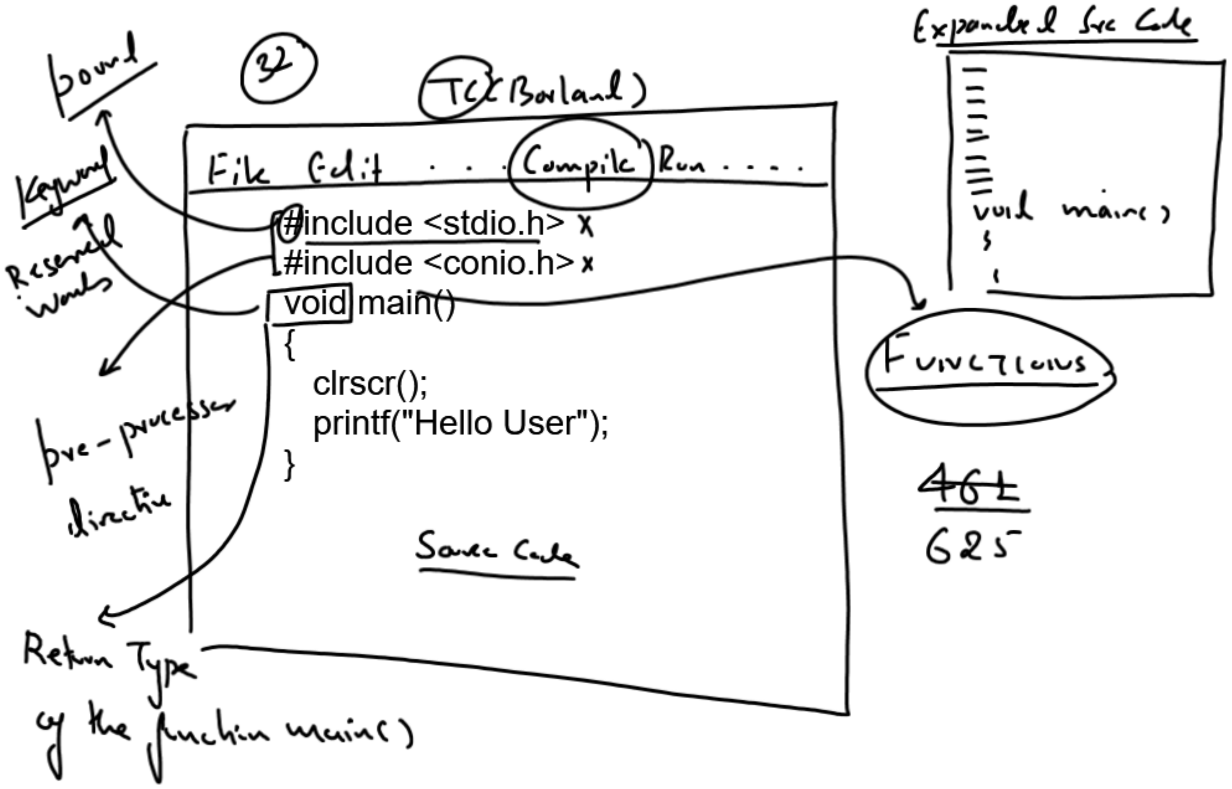


```
#include <stdio.h>
#include <conio.h>
void main()
{
   clrscr();
   printf("Hello User");
}
```

What is the symbol of # called and why do we write #include?
==

In C and C++ languages, the symbol of # is better known as **pound** and any statement which begins with # is called as **pre-processor-directive.**

These pre-processor-directives are not handled by the compiler, rather they are handled by another special software in C language called as the **pre-processor.**

This pre-processor reads our program even before the compiler reads it but it handles only those lines which begin with # i.e. pound.

Following are some popular pre-processor-directives and their meaning:

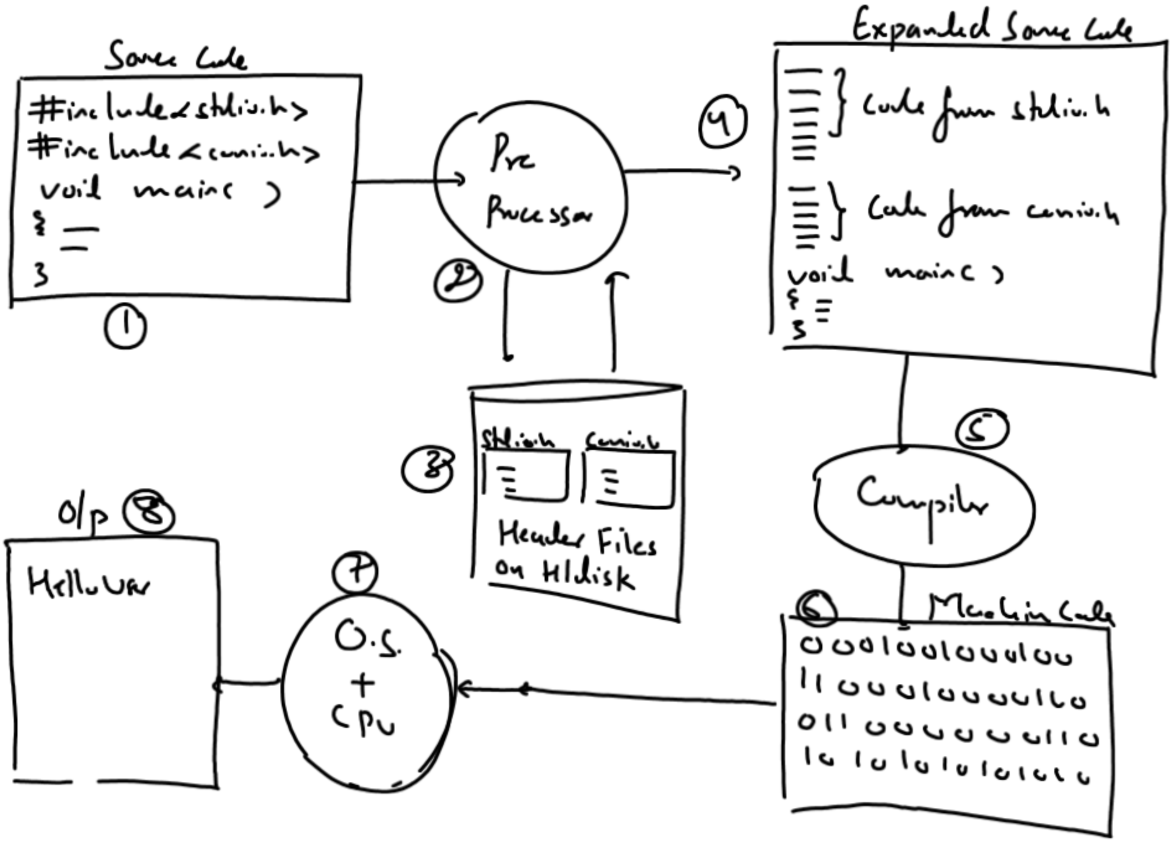
1. #include : Called as **file inclusion directive**
2. #define : Called as **macro creation directive**
3. #undef : Called as **macro removal directive**
4.
    a) #if
    b) #elif
    c) #else
    d) #endif
    e) #ifdef
    f) #ifndef

Called as **conditional compilation directive**

Amongst all of them, the most popular is the #include directive which is also called as **file inclusion directive** and as the name indicates the use it for adding header files in our program. Whenever the pre-processor finds a #include statement in our program then it takes following actions:

1. It reads the name of the header file mentioned in **<>.**
2. It copies the complete coding of the mentioned header file, removes the #include statement and in its place, paste's the header file code.

Due to this no. of lines in our program are increased and a seperate copy of our code gets created which is called as **expanded source code.** This expanded source code is then passed to the compiler which converts it into **machine code.**



```
int   main()
{
    —
    —
    —
    —
    —

    return 0;
}
```

```
void  main()
{
    —
    —
    —
    —

}
```

```
main( )
{

    —
    —
    —

}
```