

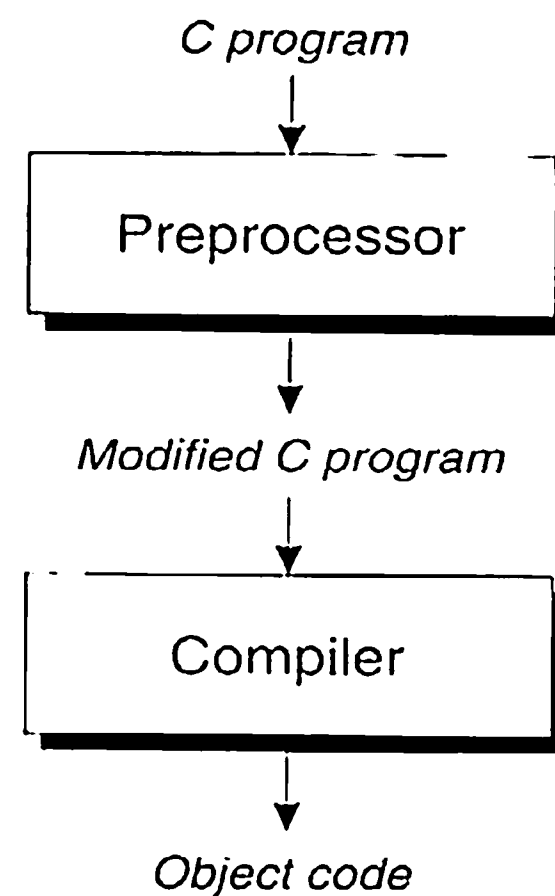
When the macro is used later in the program, the preprocessor “expands” the macro, replacing it by its defined value.

The `#include` directive tells the preprocessor to open a particular file and “include” its contents as part of the file being compiled. For example, the line

```
#include <stdio.h>
```

instructs the preprocessor to open the file named `stdio.h` and bring its contents into the program. (Among other things, `stdio.h` contains prototypes for C’s standard input/output functions.)

The following diagram shows the preprocessor’s role in the compilation process:



The input to the preprocessor is a C program, possibly containing directives. The preprocessor executes these directives, removing them in the process. The output of the preprocessor is another C program: an edited version of the original program, containing no directives. The preprocessor’s output goes directly into the compiler, which checks the program for errors and translates it to object code (machine instructions).

To see what the preprocessor does, let’s apply it to the `celsius.c` program of Section 2.6. Here’s the original program:

```

/* Converts a Fahrenheit temperature to Celsius */

#include <stdio.h>

#define FREEZING_PT 32.0f
#define SCALE_FACTOR (5.0f / 9.0f)

int main(void)
{
    float fahrenheit, celsius;

    printf("Enter Fahrenheit temperature: ");
    scanf("%f", &fahrenheit);

    celsius = (fahrenheit - FREEZING_PT) * SCALE_FACTOR;

```