PAPER 102: PROGRAMMING & PROBLEM SOLVING THROUGH C

Bidur Bhushan Handique
Dept. of Computer Science
St. Anthony's College

THE C PREPROCESSOR

- It is a program that processes a source program before it is passed to the compiler.
- This is done by a program called the Preprocessor
- The preprocessor offers several features called preprocessor directives.
- Each of the directives start with a hash (#) sign.
- The line at the beginning of every C programs

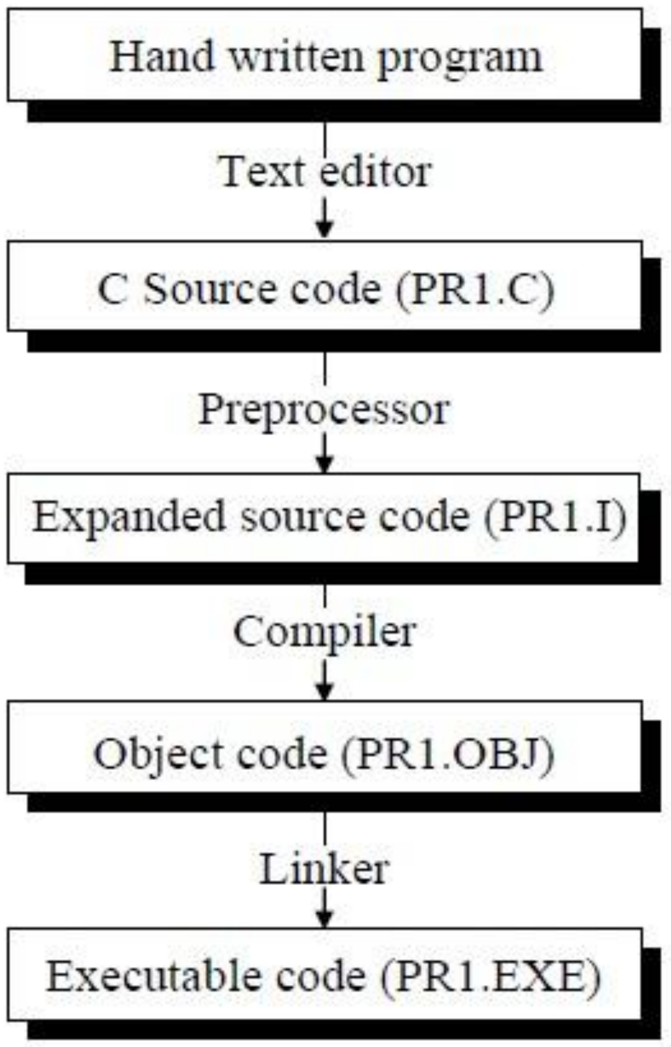
  #include <stdio.h>

  Is a preprocessor directive.

CONTINUED

- All lines preceding with a hash(#) sign are processed by the preprocessor.
- The # include directive causes the preprocessor to effectively insert the *stdio.h* file into the C program, followed by the rest of the program.

STEPS OF A PROGRAM EXECUTION



TYPES OF PREPROCESSOR DIRECTIVE

- Macro Expansion
- File Inclusion
- Conditional Compilation
- Miscellaneous

MACRO EXPANSION

- The C preprocessor permits you to define simple macros that are evaluated and expanded prior to compilation

```
#define UPLIMIT 10
void main()
{       int i;
        for(i=1;i<=UPLIMIT;i++)
                printf("\n%d",i);}
```

```
#define UPLIMIT 10
```

- Is called as 'macro definition' or most commonly known as 'macro'.
- The preprocessor replaces every occurrence of UPLIMIT in the program with 10.
- UPLIMIT is known as 'macro template' and 10 is known as 'macro expansion'.

```
#define PI 3.1415
main( )
{
float r = 6.25 ;
float area ;
area = PI * r * r ;
printf ( "\n Area of circle = %f", area ) ;
}
```

```
#define AND &&
#define OR ||
main( )
{
int f = 1, x = 4, y = 90 ;
if ( ( f < 5 ) AND ( x <= 20 OR y <= 45 ) )
printf ( "\n Your PC will always work fine..." ) ;
else
printf ( "\n In front of the maintenance man" ) ;
}
```

MACRO WITH ARGUMENTS

```
#define AREA(x) ( 3.14 * x * x )
main( )
{
float r1 = 6.25, a ;
a = AREA ( r1 ) ;//Preprocessor expands the phrase to (a=3.14 * r1 * r1)
printf ( "\nArea of circle = %f", a ) ;
}
```

```
#define SQUARE(n) (n*n)

void main()

{

int a;

a=64/SQUARE(4);

printf("\n a=%d",a);

}
```

```
#define ISDIGIT(y) ( y >= 48 && y <=
57 )

main( )

{

char ch ;

printf ( "Enter any digit " ) ;

scanf ( "%c", &ch ) ;

if ( ISDIGIT ( ch ) )

printf ( "\nYou entered a digit" ) ;

else

printf ( "\nIllegal input" ) ;
```

POINTS TO REMEMBER WHILE WRITING MACRO WITH ARGUMENTS

- Be careful not to leave a blank between the macro template and its argument while defining the macro.

```
#define AREA(x) not #define AREA (x)
```

- The entire macro expansion should be enclosed within parentheses.

```
#define SQUARE(n) (n * n) not #define SQUARE(n) n * n
```

- Macros can be split into multiple lines, with a '\' (back slash) present at the end of each line.

```
#define HLINE for ( i = 0 ; i < 79 ; i++ ) \
                    printf ( "%c", 196 ) ;
```

PROGRAM

- Write a macro to check the greatest of three numbers. Use this macro in a program to determine the greatest of three numbers input by the user and display the result.

FILE INCLUSION

- This directive causes one file to be included in another.
- The preprocessor command for file inclusion looks like this:

#include “filename”

➢ used to include user-defined header files

➢ the preprocessor searches in the same directory as the file containing the directive.

#include<filename>

➢ used to include standard library header files

➢ the preprocessor searches in an implementation dependent manner, normally in search directories pre-designated by the compiler

CONTINUED

- It can be used in two cases:

I) If we have a very large program, the code is best divided into several different files, each containing a set of related functions.

II) Commonly needed functions and macro definitions can be stored in a file, and that file can be included in every program we write.

EXAMPLE

<u>FIRST PROGRAM (FIR.C)</u>

```
#include<stdio.h>
#include<conio.h>
#include"SEC.C"
{
	display();
	getch();}
```

<u>SECOND PROGRAM (SEC.C)</u>

```
void display()
{
	printf("\nHello File Inclusion.\n");
}
```

CONDITIONAL COMPILATION

- Using special preprocessing directives, you can include or exclude parts of the program according to various conditions.
- Three reasons to use CONDITIONAL COMPILATION:

  - A program may need to use different code depending on the machine or operating system it is to run on.

  - You may want to be able to compile the same source file into two different programs, where the difference between the programs is that one makes frequent time-consuming consistency checks on its intermediate data, or prints the values of those data for debugging, while the other does not.

  - A conditional whose condition is always false is a good way to exclude code from the program but keep it as a sort of comment for future reference.

THE PREPROCESSING COMMANDS: **#IFDEF** AND **#ENDIF**

```
main()
{
#ifdef OKAY
statement 1 ;
statement 2 ;
#endif
statement 3 ;
}
```

- “ifdef” means if defined.
- Statements 1and 2 would get compiled only when macro OKAY is defined.
- Why is it better compared to comment line?

THE PREPROCESSING COMMANDS: **#IFDEF**, **#ELSE** AND **#ENDIF**

```
#ifdef INTEL
code suitable for a INTEL
#else
code suitable for a AMD
#endif
code common to both the computers
```

- To make programs portable and make them work in two totally different computers

THE PREPROCESSING COMMANDS: **#IFNDEF**

```
#ifndef INTEL
code suitable for a AMD
#else
code suitable for a INTEL
#endif
code common to both the computers
```

- “#ifndef” means if not defined.

THE PREPROCESSING COMMANDS: **#IF #ELSE** AND **#ENDIF**

```
main( )

{

#if TEST <= 5

statement 1 ;

#else

statement 2 ;

#endif}
```

- If **TEST <= 5** evaluates to true then statements 1is compiled otherwise statements 2 is compiled.

THE PREPROCESSING COMMANDS: **#ELIF**

```
#if ADAPTER == VGA
code for video graphics array
#elif ADAPTER == SVGA
code for super video graphics array
#else
code for extended graphics adapter
#endif
```

MISCELLANEOUS DIRECTIVES

- Preprocessor directives are:

  #undef

  #pragma

  #error

  #warning

#undef

- On some occasions it may be desirable to cause a defined macro to become ‘undefined’.
- #undef PENTIUM

#pragma

- This directive is another special-purpose directive that you can use to turn on or off certain features. Pragmas vary from one compiler to another.

CONTINUED

#pragma startup:

directives allow us to specify functions that are called upon program startup (before **main( )**)

#pragma exit:

program exit (just before the program terminates).

Example:

```
void fun1( ) ; void fun2( ) ;
#pragma startup fun1
#pragma exit fun2
main( )
{printf ( "\nInside maim" ) ;}
void fun1( )
{printf ( "\nInside fun1" ) ;}
void fun2( )
{printf ( "\nInside fun2" ) ;}
```

CONTINUED

#error

The directive #error causes the preprocessor to report a fatal error. The rest of the line that follows #error is used as the error message.

#warning

The directive #warning is like the directive #error, but causes the preprocessor to issue a warning and continue preprocessing. The rest of the line that follows #warning is used as the warning message.

PROGRAM

- Define macros for calculating the area of a circle, square and a triangle and store in a separate file called “areas.h”. WAP to allow the user to choose which area to be computed and take appropriate inputs from the user to compute and display the result. Use “areas.h” in this program.

  Area of a Circle = πr^2

  Area of a Square = a^2

  Area of a triangle= $(hb)/2$