Chapter 17

By St Joseph

WORD COUNT	1440	TIME SUBMITTED	09-JUL-2016 11:26PM
		PAPER ID	23989572

17. C Preprocessor directives:

17.1 Introduction

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 in C language.

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.

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

Program process flow	***File name in each steps***	***Description***
Source code ↓	test.c	->Preprocessor replaces #define(macro) #include(files),conditional compilation codes like #ifdef, #ifndef by their Respective value & source codes in source file
Expanded source code ↓	test.i	->Expand source code to assembly source code
Assembly source code ↓	test.s	->Converts assembly source code to object code
Object code ↓	test.o	-> This is a program that converts object code to executable code and also combines all object codes.
Executable code ↓	test.exe	
Execution		->Executable code is loaded and executed by loader

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

#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.

Example:

```
#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 : ?
```

17.2 File inclusion:

An external file containing functions or Marco definitions can be included as a part of a program.

Syntax:

#include"filename"

Or

#include<filename>

Example program for conditional compilation directives:

17.3 CONDITIONAL COMPILATION

17.3.1. 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.

Example:

```
#include <stdio.h>

#define RAVI 100

 int main()

{

  #ifdef RAVI

  printf("RAVI is defined. So, this line will be added in " \

         "this C file\n");

  #else

  printf("RAVI is not defined\n");

  #endif
  return 0;
}
```

Output:

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

17.3.2 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.

Example:

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

  #endif
```

```
   return 0;

}
```

Output:

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

17.3.3 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.

Example:

```
#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.

Example:

```
#include <stdio.h>

#define height 120

void main()

{

   printf("First defined value for height    : %d\n",height);

   #undef height          // undefining variable

   #define height 500     // redefining the same for new value

   printf("value of height after undef \& redefine:%d",height);

}
```

Output:

```
First defined value for height : 120

value of height after undef & redefine : 500
```

17.4 OTHER DIRECTIVES

1

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

Example:

```
#include <stdio.h>


void fun1( );

void fun2( );


#pragma startup fun1

#pragma exit fun2
```

```
int main( )
{
   printf ( "\n Now we are in main function" ) ;
   return 0;
}
void fun1( )
{
   printf("\nFunction1 is called before main function call");
}

void fun2( )
{
   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
```

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

17.5 Predefined Macros

ANSI C defines a number of macros. Although each one is available for use in programming, the predefined macros should not be directly modified.

Macro	Description
__DATE__	The current date as a character literal in "MMM DD YYYY" format.
__TIME__	The current time as a character literal in "HH:MM:SS" format.
__FILE__	This contains the current filename as a string literal.
__LINE__	This contains the current line number as a decimal constant.

__STDC__	Defined as 1 when the compiler complies with the ANSI standard.

Example:

```
#include <stdio.h>

main() {

   printf("File :%s\n", __FILE__ );

   printf("Date :%s\n", __DATE__ );

   printf("Time :%s\n", __TIME__ );

   printf("Line :%d\n", __LINE__ );

   printf("ANSI :%d\n", __STDC__ );

}
```

Output:

```
File :test.c

Date :Jun 2 2016

Time :04:15:15

Line :8

ANSI :1
```

17.6 Preprocessor Operators:

The C preprocessor offers the following operators to help create macros –

17.6.1 The Macro Continuation (\) Operator:

A macro is normally confined to a single line. The macro continuation operator (\) is used to continue a macro that is too long for a single line

Example:

```
#define  message_for(a, b)  \

   printf(#a " and " #b ": CSE SJIT!\n")
```

17.6.2. The Stringize (#) Operator:

The stringize or number-sign operator ('#'), when used within a macro definition, converts a macro parameter into a string constant. This operator may be used only in a macro having a specified argument or parameter list.

Example:

```
#include <stdio.h>

#define  message_for(a, b)

   printf(#a " and " #b ": Welcome!\n")

int main(void) {

   message_for(Ram,Vijay);

   return 0;

}
```

Output:

```
Ram and Vijay: Welcome!
```

17.6.3 The Token Pasting (##) Operator:

The token-pasting operator (##) within a macro definition combines two arguments. It permits two separate tokens in the macro definition to be joined into a single token.

Example:

```
#include <stdio.h>

#define tokenpaster(n) printf ("token" #n " = %d", token##n)

int main(void) {
  int token34 = 40;

  tokenpaster(34);

  return 0;

}
```

Output:

```
token34 = 40
```

17.6.4. The defined() Operator:

The preprocessor defined operator is used in constant expressions to determine if an identifier is defined using #define. If the specified identifier is defined, the value is true (non-zero). If the symbol is not defined, the value is false (zero).

Example:

```
#include <stdio.h>

#if !defined (MESSAGE)

  #define MESSAGE “Help!"

#endif

int main(void) {

  printf("Here is the message: %s\n", MESSAGE);

  return 0;

}
```

Output:

```
Here is the message: Help!
```

17.7 Parameterized Macros:

One of the powerful functions of the C is the ability to simulate functions using parameterized macros.

Example:

```
#include<stdio.h>
#define MAX(x,y) ((x) > (y) ? (x) : (y))

int main(void) {

   printf("Max between 20 and 10 is %d\n", MAX(10, 20));

   return 0;

}
```

Output:

```
Max between 20 and 10 is 20
```

Chapter 17

ORIGINALITY REPORT

84%

SIMILARITY INDEX

PRIMARY SOURCES

1	fresh2refresh.com Internet	799 words — 55%
2	www.tutorialspoint.com Internet	294 words — 20%
3	www.alphagroup.edu Internet	112 words — 8%
4	Klein, Gerwin. "Operating system verification--An overview", Sadhana/02562499, 20090201 Publications	7 words — < 1%

EXCLUDE QUOTES	OFF	EXCLUDE MATCHES	OFF
EXCLUDE BIBLIOGRAPHY	OFF		