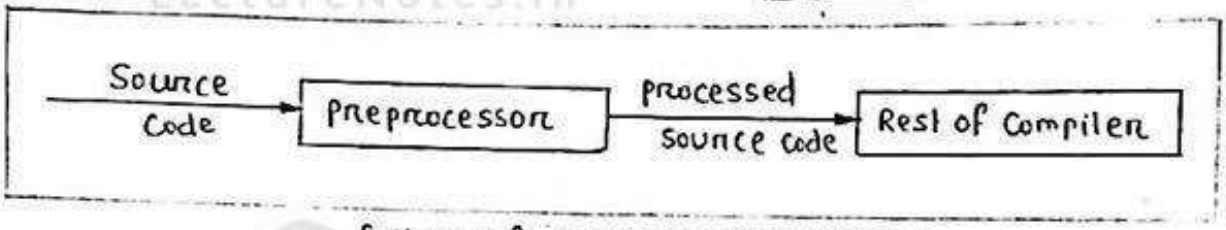
C Preprocessor : Lesson Number 17

→ Using preprocessor C is different from other programming languages.

→ The C preprocessor is a tool which filters the source code before it is compiled.

→ Before compilation, the source codes are passed to the C preprocessor and preprocessor scans source codes and modifies it. At the time of compilation, itself, some processes can be done in C. The command which invoke such processes are called preprocessor directives.



(Stage of preprocessing work)

Features of preprocessor directives:

1. Each preprocessor directive start with # symbol.
2. There can be one directive on a line.
3. There is no semicolon present at the end of directive.
4. It can present at the beginning of a program.
5. A directive is active at the end of the program.

Preprocessor Directives:

Basically, the preprocessor directives are classified into 3 types
1. File inclusion directives
2. Macro substitution directives
3. Compiler control directives.

1) File inclusion directive:

An external file containing functions or macro definitions can be included as a part of the program, so that we need not rewrite these functions or macro definitions.

Exp: #include : include a source file.

[Reference : programming with ANSI and Turbo C, Ashok N. kamthane]

2. Macro substitution directives :

→ The macro substitution directives defines an identifier and a string that will be substituted for the identifier each time the identifier is encountered in the source file.

→ The most two common forms of macro substitutions are ;

1. Simple macro substitution
2. Augumented macro substitution.

→ Example : # define : define a macro.

3. Compiler Control Directives :

→ A number of directives control conditional compilation, which allows certain portions of a program to be selectively compiled on ignored depending upon specified conditions. This C pre-processor feature is known as conditional compilation.

→ Example : #if : Conditional Compilation
ifdef : Conditional Compilation
ifndef : Conditional compilation
elif : Conditional Compilation (ANSI addition)
else : Conditional Compilation.
line : Control error reporting.
: null directive ; no effect (ANSI addition)

Advantages using preprocessors :

The advantages using preprocessors are :

1. Readability of the program is increased.
2. program modification becomes easier.
3. Makes program efficient.

Types of C Pre-processor :

→ The C pre-processor includes ;

1) The #include Directive : [File Inclusion (#include)]

→ C programs are divided into modules or functions.

→ Some functions are written by users like us and many others are stored in the C library.

→ Library functions are grouped category-wise and stored in different files known as header files.

→ If we want to access the functions stored in the library, it is necessary to tell the compiler about the files to be accessed.

→ This is achieved by using the preprocessor directive #include as follows.

#include "filename" Example: #include "stdio.h"
or #include <filename> Example : #include <stdio.h>

filename is the name of the library file that contains the required function definition. Preprocessor directives are placed at the beginning of a program.

2) The #define Directive : [Macro substitution (#define)]

→ The #define is a preprocessor compiler directive, not a statement.

→ Therefore, #define lines should not end with a semicolon.

→ Symbolic constants are generally written in uppercase so that they are easily distinguished from lowercase variable names.

→ #define instructions are usually placed at the beginning before the main() function.

→ Symbolic constants are not declared in declaration section.

→ Syntax : #define Symbolic Constant Value } Simple macro substitution
Example : #define period 10

→ The preprocesson #define also permits to define more complex and more useful form of replacements.

→ The general form is;

.#define identifier(arg1, arg2, ···, argn)

where arg1, arg2, ··argn are formal macro arguments.

→ Example: #define SQR(x) (x*x)
#define LOOP(n) for(i=1; i<=n'; i++)
} argumented macro substitution.

3) The #if Directive : [compiler control (#if)]

→ The #if constructs takes a single integral constant expression as their argument.

→ Syntax : #if condition

→ Example : #if ULONG-MAX != 0

4) The #ifdef Directive : [Compiler Control (#ifdef)]

→ The #ifdef directive is used to test the definition.

→ For example; #ifdef Name
/* compiles these lines if NAME is defined */
#endif

5) The #ifndef Directive : [Compiler Control (#ifndef)]

→ The #ifndef directive is used to test the definition.

→ For example : #ifndef Name
/* Compile these lines if NAME is not defined */
#else
/* Compile these lines if NAME is defined */
#endif.

6) The #elif Directive : [Compiler control (#elif)]

→ Same as #if directive.

Standard Library Functions :

1) Standard input output Library functions : <stdio.h>

Name of the Functions	Description
getchar()	Reads a character from the keyboard; waits for carriage return.
getch()	Reads a character without an echo to the screen.
getche()	same as getch(), but echoes a character
putchar()	writes a character to the screen.
putch()	writes a character to the screen.
gets()	Reads a string of characters.
puts()	write a string to the screen.
printf()	write data in various formats to output device.
scanf()	Read data in various format to input device.

2) Standard time Library functions : <time.h>

Functions	data type returned	Task
difftime(t1,t2)	double	Return the time difference t1~t2, where t1 & t2 represents elapsed time beyond a designated base time.
time(t)	long int	Return the no. of seconds elapsed beyond a designated base time.

3) Standard math library functions: <math.h>

Functions	Data Type returned	Task
acos(d)	double	Return the arc cosine of d.
asin(d)	double	Return the arc sine of d.
Cos(d)	double	Return the cosine of d.
Cosh(d)	double	Return the hyperbolic Cosine of d.
exp(d)	double	Raise e to the power d.
fabs(d)	double	Return the absolute value of d.
fmod(d1,d2)	double	Return the remaider of d1/d2.
pow(d1,d2)	double	Return d1 raised to the d2 power.

4) standard character testing & Conversion Library functions : <ctype.h>

Functions	Data Type returned	Task
isalnum(c)	int	Determine if argument is alphanumeric. Return nonzero value if true, 0 otherwise.
isalpha(c)	int	Determine if argument is alphabetic. Return nonzero value if true, 0 otherwise.
isspace(c)	int	Determine if argument is a white space character. Return non-zero value if true, 0 otherwise.
tolower(c)	int	Convert letter to lowercase.
toupper(c)	int	Convert letter to uppercase

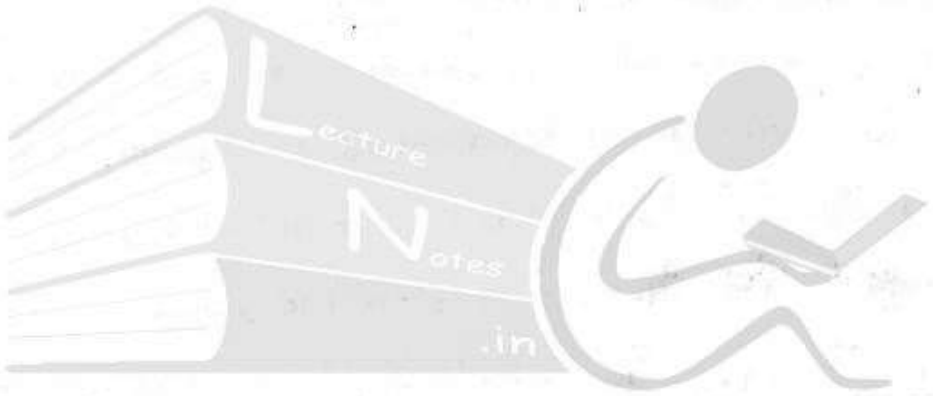
5) Standard string library functions : <string.h>

Functions	Data Type Returned	Task
Strcmp (S1, S2)	int	Compare two strings lexicographically. Return a negative value if S1 < S2 ; 0 if S1 and S2 are identical and a positive value if S1 > S2.
Strcpy (S1, S2)	char *	Copy string S2 to string S1
Strlen (S)	int	Return the number of characters in string S.
Strset (S, C)	char*	Set all characters within S to C. (Excluding the terminating null character '\0').

6) Standard library functions : <stdlib.h>

Functions	Data Type Returned	Task
abs (i)	int	Return the absolute value of i.
atoi (S)	int	Convert string S to an integer.
atol (S)	long	Convert string S to a long integer.
rand (void)	int	Return random positive integer.
System (S)	int	pass Command string S to the OS. Return 0 if Command is Successfully executed, otherwise return a nonzero value typically -1.

7) Standard Console input output functions : <conio.h>



Assignment : what is preprocessor directives ?

→ The C preprocessor is a tool which filters the source code before it is compiled.

→ Before compilation, the source codes are passed to the C preprocessors and preprocessor scans source codes and modifies it. At the time of compilation, itself, some processes can be done in C. The command which invoke such processes are called preprocessor directives.

Assignment : write down the features of a preprocessor directive.

1. Each preprocessor directive starts with # symbol.
2. There can be one directive on a line.
3. There is no semicolon (;) present at the end of directive.
4. It can present at the beginning of a program.
5. A directive is active at the end of the program.

Assignment : what is the function of a file inclusion directive ?

Sol? File inclusion directive : An external file containing functions or macro definitions can be included as a part of the program, so that we need not rewrite these functions or macro definitions.

Example : #include : include a source file.

→ C programs are divided into modules.
→ some functions are written by users like us and many others are stored in the C library.
→ The library functions are grouped category-wise and stored in different files known as header files.
→ If we want to access the functions stored in the library, it is necessary to tell the compiler about the files to be accessed.

→ This is achieved by using the preprocessor directive #include as follows.

#include "filename" or

#include <filename>

Example: #include "stdio.h" or

#include <stdio.h>

Assignment: write the library functions which are included in the standard input output header file or <stdio.h>.

Soln: The library functions included in the standard input output header files are;

getchar() : Reads a character from the keyboard; waits for carriage return.

getch() : Reads a character without an echo to the screen.

getche() : Same as getch(), but echnoes a character.

putchar() : writes a character to the screen.

putch() : writes a character to the screen.

gets() : Reads a string of characters.

puts() : write a string to the screen.

printf() : write data in various formats to output device.

scanf() : Read data in various format to input device.

Assignment: what is the function of a library function isalpha(), and isalnum()?

Soln

isalpha(c) : Determine if argument is alphabetic. Return nonzero value if true, otherwise 0(zero).

isalnum(c) : Determine if argument is alphanumeric. Return nonzero value if true, 0(zero) otherwise.