Lesson 2_1 - Basic Structure

Topics:
- ♦ Naming a program
- ♦ Writing a simple C program
- ♦ Using the printf() function to display text on the screen
- ♦ Structure of a simple C program
- ♦ Basic rules for writing a C program

The program illustrates the basic structure of a C program. When you execute the program, the statement:

Welcome to C!

appears on the screen and it will remain there until you have performed other tasks which will eliminate it.

Examine the program and the output carefully before you read the explanation. **Can you reason out specifically what the "printf" function does? If you wanted to display the statement, "I am in the process of learning a programming language" onto the screen, how would you write the "printf" statement?**

Source Code

```
main()
{
 print("Welcome to C!");
}
```

Output

```
Welcome to C!
```

1. *What is main()?*

- ♦ Every C program is divided into units, each unit is called a function. Every C program has a primary function that must be assigned the name mane. The name is mandatory and cannot be altered by the user.
- ♦ The expression main() identifies the program unit as a main function. In C, all functions start with a name, followed by a pair of parentheses. There may or may not be arguments inside the parentheses. In this example, there is nothing inside the parentheses in main.

2. *How do we start and end the program body of a C main function?*

- It must start with an opening brace {.
- It must end with a closing brace }.
- Between the opening and closing braces are the program body which consists of blanks (if you want to do nothing) or C statement(s), i.e., the structure of a typical C main program is as follows:

```
main ()
{                   Begin program body
 statement 1;
                    Program body
 statement 2;
 . . .
}                   End Program body
```

3. What is print("Welcome to C!")?

- The statement printf("Welcome to C") is a statement which calls the function printf which is located in the compiler library. The printf function in the library looks much like the main function in this program in that it has a name and is followed by a program body which is enclosed by braces. This particular library function as it is used in this program causes the phrase between the double quotes (called a character string) to be printed on the screen. The printf function is one of many standard functions contained in the library of the C compiler. These functions are available to you as a programmer and may be activated (called) by using a statement in your program body.
- In C, we use the following structure to call a standard C function from the main function:

  function_name(Argument1,Argument2,...);

  where Argument1, Argument2... are the argument values that we want to pass from the main function to the function to be called. If you have more than one argument to pass, a comma shall be used to separate one argument from the other. Some functions may require no arguments at all (called void arguments). In any case, whether we have argument(s) or not, a pair of parentheses must be used to enclose the argument list. In addition, the calling statement must end with a semicolon.

- The simplest way to call printf() is

```
printf(format string);
```

  where the format string represents an argument being passed from main to the printf function. The calling statement is begun with printf and a left parenthesis. This is followed by a double quote, the character string to be printed, and closed by another double quote and a right parenthesis. The statement must be ended with a semicolon.

4. *How do we terminate a C statement?*

♦ A C statement must be terminated by a semicolon ;. The semicolon at the end of a C statement acts much like a period at the end of a sentence.

5. *Can we use both upper and lower case to write a C statement?*

♦ The C language distinguishes between lower case and upper case. Thus, printf is different from PRINTF or Printf or PrIntF. It is, thus, said that C is considered to be case sensitive. In this lesson, both main() and printf() must be written in lower case. For other self-developed functions, you may use whatever case you consider appropriate.

6. *When do we need blank space(s) in a C statement?*

♦ A C statement consists of a number of “words” and characters. A C “word” can be a function name, such as main, or a C reserved words which will be discussed later in this Chapter. All C words should be written continuously. For example, the expression

```
ma    in()
```

is not legal because no blank characters are allowed between the two characters a and I in the word main.

♦ Between “words”, white-space characters (such as blank, tab, and the carriage return) can be inserted, but this is optional. For example, the expression

```
main()
```

is equivalent to

```
main (              ) or main     (  )
```

In general, it is acceptable to add blanks between “words” but not acceptable to add blanks within “words”.

7. *Can we write a C statement at any location?*

♦ The white-space character is invisible to the C compiler. Therefore, you have the freedom to write a C statement at any row or column you like. The C compiler allows you, for example, to rewrite and pack program 2_1.C into one line

```
main(){printf(“Learn the basic structure of a C program!”);}
```

or rewrite it as:

```
main(              ) {
  printf
(  “Learn the basic structure of a C program!” )    ;               }
```

However, these styles will make your program more difficult to understand and should not be used.

Exercises

1. True or False:

```
a. In general, C statements are case sensitive.
b. By default, any C statement is location sensitive.
c. A C statement must be terminated with a period.
d. The name of the primary function of a C program must be
   MAIN.
e. main (){} is a complete and correct C program.
f. 12Test.C and %K12@.C are valid C source file names.
g. 1/2Test.C and “K12@.C are valid C source file names.
```

2. Find the error(s), if any, in each of the statements below:

```
a. main();
b. printf (“Do we need parentheses here?”);
c. printf(“Where do we need blank space?   “    )    ;
d. print (“Is anything wrong here?”)
```

3. Compile and run the correct program 2_1a.C.below:

```
main(             )
   { printf (“Good Morning! Paris”    )    ;    }
```

4. Correct, and then compile and run the wrong program 2_1b.C below

```
ma   in() PRINTF  *, (‘What is wrong?’)
```

Solutions

1.
```
A B C D E F G
T F F F T T F
```

2.
```
a. main()
b. No error
c. No error
d. print(“Is any thing wrong here?”)
```

Lesson 2_2 - Writing Comments

Topics:
- Structure of comments
- Location of comments
- Continuing comments
- Style of writing comments

If you have thumbed through later chapters in this book you have probably noticed that computer code (that is, the actual C statements) looks somewhat cryptic. There are times when even the most experienced programmers find it difficult to understand the tasks that a program or a portion of a program are to be performing. You will very likely find as you go on that if you pick up code that you have written yourself only a few days earlier that you have trouble understanding how it is supposed to be working.

To make your life easier, though, you can write “comments” to yourself within the code of your program. Comments are essentially notes describing what a particular portion of your program does and how it does it (or anything else that you would like to write in the middle of your code).

If comments are not written correctly, though, the C compiler may think that the words written in the comments are functions, such as printf, or other C expressions or statements. If this occurs, your program will get errors when you compile it. Look at the next example program and see if you can tell the way that you signify to the C compiler that a statement is a comment statement.

Source Code

```
/*2_2.C - In this book, this line is used to
          give the program name */

main()
{/*The purpose of this program is to print one
  statement to the screen */

 printf(“How  do we write comments in C?”); /*
 The printf function prints to the screen */

 /*This program performs the same task as
  program  2_1.C */
}/* A closing brace is needed to end the program*/
```

Output

```
How  do we write comments in C?
```

Explanation

1. *What is the structure of a comment?*

   - The syntax of a C comment is

     ```
     /* Any text, number, or character */
     ```

     where there should not be any blank between the slash and the asterisk. In addition, the /* and */ must form a couple.

   - The /* and */ must form a couple, but they don/t need to be in the same line. Therefore, a comment line may occupy more than one line. A multi-line comment starts with /*, followed by multi-line of texts, numbers, or character strings, and closed with a */. See the first comment for example.
   - Examples of incorrect comments:

     ```
     /* Wrong comment 1, no end asterisk and slash
     /* Wrong comment 2, no end slash *
     /  *Wrong comment 3, there is a blank between /and * * /
     ```

2. *Can we write comments before and after a program?*

   - Yes, a comment line can be written in the very first line of a program. It can also be written after the end of the program.

3. *Can a comment appear in a C statement?*

   - It depends. The C compiler treats comments like a single white space character. Therefore, comments can only appear in a C statement where a white-space character is allowed to be present.
   - Examples of correct comments:

     ```
     printf /*This comment is legal */(“Welcome to C!”);
     printf ( /*This comment is also legal*/”Welcome to C!”);
     ```

   - Examples of incorrect comments:

     ```
     print   /*  This  comment  is  illegal  because  it  splits  a  C
     function
     name  */ intf(“Welcome to C!”);
     ```

```
printf("This comment /* will be displayed*/ ");
```

The comment above becomes part of the format string and will be displayed. Therefore, the comment is not a comment any more.

4. *Can we write a nested comment?*

- No, comment statements cannot be nested (meaning that we cannot write a comment within a comment) in C, for example:

```
/*/* This is an illegal comment because it is */ nested
*/
```

5. *Style*

- The comments in the following statements are legal, but we recommend using the first one:

```
main () /* This is a good location to write a comment*/
main    /* This is a poor location to write a comment*/  ()
```

- Be wise in your use of comments. Remember that comments are used to enhance the understandibility of your programs. Make them pleasing to the eye and clear. With practice you will develop a style of writing comments that is of benefit to you. In this text, we do not use many comments primarily because we want you to interpret our code. Your programs should use comments far more frequently than we have used them.

Exercises

1. True or False:

```
a. A C comment line may appear on the first line of a
   program.
b. A C comment line may appear on the last line of a
   program.
c. We may write a comment at the end of a C statement.
d. We may write a comment line which  contains 120
   characters.
e. Never write a nested comment.
```

2. Compile and run the correct program 2_2a.C below:

```
/* Comment before the program */
   main /* We learn how to write comments*/ ()
{ printf  /*This is not a nested*/  /*comment*/
  ("Good Morning! Paris"  );
} /* Comment after the program is OK */
```

3. Correct, and then compile and run the wrong program 2_2b.C below:

```
ma/* This comment is illegal */ in()
{printf(“Good Morning! Paris”);
/*This /* is a nested*/ comment */ }
/* This comment does not have the closing ‘*’ and ‘/’
```

Solutions

1.
```
A B C D E
T T T T T
```

Lesson 2_3 - Line Feeding

Topics:
- Formatting output
- Line feeding

The first example program of this chapter showed how to print a single line to the screen. However, in most cases you will want to print multiple lines to the screen, and you will want to display these lines in such a way that you will never have proper spacing between them. Proper spacing can be achieved by what is called line feeding.

Read the next example and look at the output. See if you can determine how you can write your printf statements to get the line feeding to occur.

Source Code

```
main()
{
 printf(“Welcome to”);
 printf(“London!”);
 printf(“\nHow do we\njump\n\ntwo lines?\n”);
 print(“\n”);
 printf(“It will rain\ntomorrow\n”);
}
```

Output

```
Welcome toLondon!
How do we
jump

two lines?

It will rain
tomorrow
```

Explanation

1. *Can we use two printf() functions to linefeed a string?*

   - Suppose we want to display

```
Welcome to
London!
```

in two lines on a screen. Can we use two printf() functions

```
printf(“Welcome to”);
printf(“London!”);
```

to reach our goal? The answer is No. This is because the C compiler never automatically advances a line to the next line after each printf() is called. Therefore, the output from the first printf() function will be connected to the output of the next printf() function and that is the reason why. “Welcome to” and “London!” are printed at the same line.

2. *How do we linefeed a line?*

♦ The linefeed operation can be done easily with the printf() function using the linefeed symbol \n (called backslash n). The symbol \n consists of two characters, \ and n, with no blank in between. In C, the two character symbol \n is one of the character escape sequences. The escape sequence \n functions both as a carriage return and a linefeed only and will not be displayed on the screen. Any data behind this symbol is written at the beginning of the next line. You can use \n at any location in the string to be displayed. The \n can be at the beginning, in the middle, or at the end of a string. The number of \n can be more than one. For example, in the statement

```
printf(“\nHow do we\njump\n\ntwo lines? \n”);
```

the program uses the first \n to move the cursor to a new line, displays How do we, uses the second \n to jump to a new line, prints jump, use the next two \ns to jump another two lines, prints two lines? (thus only one blank lines were left), and use the last \n to jump one more line.

3. *Can we use the linefeed symbol by itself?*

♦ Yes. The symbol \n can be used by itself to advance line as long as it is contained in double quotes.

Exercises

1. True or false:

```
a. The function printf (\n\n\n); Will create 3 blank
   lines.
b. The function printf (“\nnn”);  will create 3 blank
   lines.
c. The function printf (“\n\n\n”); will create 3 blank
   lines.
d. The function printf (“\n \n \n”); will create  3 blank
   lines.
e. The function printf (“\ n\ n\ n”); will create 3 blank
   lines.
```

2. *Find errors, if any, in each of the statements below:*

```
a. printf(“I \n Love \n California \n”);
b. printf(“I \ n Love \ n California \ n”);
c. printf(“I   n Love   n California \ n”);
```

3. *Compile and run the correct program 2_3a.C below:*

```
main()
{printf(“I \n Love \n California \n”);
 printf(“I \ n Love \ n California \ n”);
 printf(“I n Love n California n”);
}
```

4. *Write a program to display a 10-line story on your screen.*

Solutions

1.
```
A B C D E
F F T T F
```

2.
```
a. No error
b. No error, but will not linefeed, character n will also
   be displayed
c. No error, but will not linefeed, character n will also
   be displayed
```

Lesson 2_4 - More Escape Sequences

Topics:
- Moving the cursor
- Generating sound
- Concatenating a C string

The \n escape sequence is one of many that can be used within the pair of double quotes in the printf function. Escape sequences shown in the next program are capable of generating a beep and moving the cursor to different locations on a line. By looking at the next program and its output, see if you can interpret what the escape sequences are and what they do Why do the words “I earned $50” not appear in the output?

Source Code

```
main()
{printf(“Listen to the beep now. \a”);
  print(“\nWhere is the ‘t’ in cat\b?\n\n”);

  printf(“I earned $50 \r Where is my money?\m);
  printf(“The rabbit jumps \t\t two tabs.\n\n”);

  printf(“Welcome to\
  New York!\n\n”);

  printf(“From “              “Russia\
  with “      “Love.\n”);
}
```

Output

```
Listen to the beep now.
Where is the ‘t’ in ca?

Where is my money?
The rabbit jumps         two tabs.

Welcome to New York!

From Russia with Love.
```

Explanation

1. *How do we generate beeps?*

- The escape sequence \a in the printf() format string

```
printf("Listen to the beep now. \a");
```

generates a beep after "Listen to the beep now." has been displayed.

2. How do we backspace?

- The escape sequence \b in the printf() format string

```
print("\nWhere is the 't' in cat\b ?\n");
```

moves the cursor back one space after the t in the cat has been displayed, so we do not see the t.

3. How do we move the cursor to the beginning of the current line?

- The escape sequence \r in the printf() format string

```
printf("I earned $50 \r Where is my money?\n");
```

will not display any character before \r. The escape sequence \r represents a carriage return and moves the cursor to the beginning of the current line.

4. How do we continue a C statement in multiple lines?

- **Method 1**: We use a backslash at the end of a line to indicate that this line has not finished and is to be continued on the next line. Since the C compiler disregards all blank characters behind a statement, the connection to the next line will start right at the end of the preceding statement. If you want to include blank characters in a statement which occupies two lines, you should either place them before the backslash in the first line or place them at the beginning of the second line. For example, the statement

```
printf("Welcome to New
York!")
```

is equivalent to

```
printf("Welcome to New        York!");
```

but not

```
printf("Welcome to New York!");
```

- **Method 2**: We enclose each unfinished string in double quotation marks, e.g., the statement

```
printf("From " "Russia "
        “with” “ love.\n”);
```

is equivalent to the statement

```
print (“From Russia with love\n”);
```

- **Method 3**: Combination of method 1 and method 2. For example, the statement above is equivalent to:

```
printf(“From “ “Russia \
with “ “love.\n”);
```

5. *Summary*

- Character escape sequences consist of a backslash followed by a letter or by a combination of digits. Each of them represents a character that has special meaning or specifies an action.printf() and other output functions may use the following character escape sequences in its format string (note: This is a complete list of escape sequences which you can use as a reference later. At this point you do not need to understand the meanings of all of these):

Escape sequence	Meaning	Result
\0	Null character	Ends a character string
\a	Bell	Generates an audible beep
\b	Backspace	Backs one space from the current position
\f	Form feed	Ejects printer page
\n	New line	Linefeeds to next line
\r	Carriage return	Moves the cursor to the beginning of the current line
\t	Horizontal tab	Moves the cursor to a tab horizontally. One tab is 8 spaces
\v	Vertical tab	Moves the cursor a tab vertically
\0ddd	Octal constant	Represents a character using a base 8 constant, where ddd = 1, 2, or 3 octal digits, e.g., \0340 is character alpha (see later section)
\xdd or \Xdd	Hexadecimal constant	Represents a character using base 16 constant, where dd = 1 to 2 hex value, e.g., \xE0 is character alpha
\\	Backslash	Displays a backslash
\’	Single quote	Displays a single quote, e.g., XY=‘\”;

Escape sequence	Meaning	Result
\”	Double quote	Displays a double quote
\%	Percent	Displays a percent character
\?	Question mark	Prevents the misinterpretation of trigraph-like character sequence, e.g., trigraph sequence ??= will display character #, but \?\?= will display ??= (see Lesson 2_5)
\	Continue	Continue an unfinished statement

Exercises

1. True or False:

```
a. The statement printf (“ABC\a\a”); will display ABC
   and generate two beeps.
b. The statement printf(“ABC\b\b”); will display A only.
c. The statement printf(“ABC\r\r”); will display nothing.
d. The statement printf(“ABC\t\t”); will display ABC.
```

2. Compile and run the correct program 2_4a.C below:

```
main()
{printf(“\n1. I Love \a\a\a California \n”);
 printf(“\n2. I Love \b\b\b California \n”);
 printf(“\n3. I Love \r\r\r California \n”);}
 printf(“\n3. I Love \t\t\t California \n”);}
```

Solutions

```
1.   A B C D
     T T T T
```

Lesson 2_5 - Special Characters

Topics:
- Displaying special characters

You can see that some of the characters used in the printf function have special meaning. For instance, double quotes enclose the characters to be printed. **So you may ask, “If I want to print double quotes, how do I do it?” or, “The backslash is used for escape sequences so how do I print a backslash on the screen?”.**

We will also find that the percent sign (%) is a special character. The next example illustrates how special characters can be printed. Before you read the explanation, see if you can deduce what needs to be done to be able to print special characters.

Source Code

```
main()
{printf(“Print 3 double quotes     -\” \” \” \n”);
 printf(“Print 3 back slashes -\\ \\ \\ \n”);
 printf(“Print 3 percents          -\% ,% ,% \n”);
 printf(“Print 3   percents   -%% %% %% \n”);
}
```

Output

```
Print 3 double quotes    - “ “ “
Print 3 back slashes          - \ \ \
Print 3 percents         - % % %
Print 3 percents         - % % %
```

Explanation

1. *How do we display double quotes?*

   - To display double quotes in C, we must put a back slash immediately in front of the double quote that we want to display. No space is allowed between the back slash and the double quote.

2. *How do we display a backslash?*

   - To display a backslash in C, we must put a back slash immediately in front of the back slash that we want to display. No space is allowed between the two back slashes.

3. *How do we display a percent?*

- To display a percent in C, we must put a percent immediately in from of the percent that we want to display. No space is allowed between the two percents. Some compilers allow us to put a back slash in place of the first percent.

4. How do we display a question mark or a single quote?

- You have two choices. You may display a question mark or a single quote directly or put a backslash before the question mark or the single quote. For example, the two statements:

  ```
  printf("\' and \?");
  printf("' and ?");
  ```

  are equivalent and both will display

  ```
  ' and  ?
  ```

- In some special occasions the first choice may not work. For example, if you use trigraph characters (which are sequences of three characters introduced by two consecutive question marks) such as ??= (which represents character # in the printf string) and you want to display the trigraph character ??=, then you need to use a backslash before the question mark. For example, the statement below

  ```
  printf("Does the trigraph \?\?= (??=) represent the"
         " character # ?\n");
  ```

  will display

  ```
  Does the trigraph ??= (#) represent the character # ?
  ```

  a list of trigraph characters is shown below:

Trigraph	Meaning
??=	#
??(	[
??/	\
÷÷)	]
÷÷'	^
??>	{
??!	|
??>	}
??-	~

Exercises

1. True or False:

```
a. We use 3 percents %%% to display a percent.
b. We use 3 double quotes “”” to display a double quote.
c. We use “” to display a double quote.
d. We use \” to display a double quote.
e. We use  \ “ to display a  double quote.
f. We use \\ to display a back slash.
g. We use \ , to display a back slash.
h. \\\\ is equivalent to \ \ \ \.
i. \\\\ will display two back slashes.
```

2. Compile and run the correct program 2_5a.C below:

```
main()
{printf(“Print 5 single quotes   - ‘ ‘ “ “‘ ‘ ‘\n”);
 printf(“Print 5 double quotes  - \” \””  “\” \” \”\n”);
 printf(“Print 5 back slashes   - \\ \\ “” \\ \\ \\\n”);
 printf(“Print 5 percents       - %% %%” “%% %% %%\n”);
}
```

3. Correct, and then compile and run the wrong program 2_5b.C below:

```
main()  {
 printf(“Print 5 single quotes  --- \’\’\’\’\’ \n”);
 printf(“Print 5 double quotes  --- “””””      \n”);
 printf(“Print 5 back slashes   --- \\\\\      \n”);
 printf(“Print 5 percents       --- %%%%%      \n”);
}
```

4 Use the printf() function to display the following sentences on the screen:

```
He said : “The interest rate is 5% now”.
In C, the symbol for division is ‘/’, not “\”.
“Can you print ‘“/\”%’ on the screen?”
```

Solutions

```
1.    A B C D E F G H I
      F F F T F T F F T
```

Lesson 2_6 - Review Chapter 2

Topics:
- ♦ Basic structure of a C program
- ♦ Writing comments
- ♦ Using character escape sequences
- ♦ Displaying special characters
- ♦ Concatenating a C statement

The next program uses the techniques you have learned in this chapter. Read the program. You should feel comfortable with understanding everything it does before proceeding to the next chapter.

Source Code

```
main()
{/*  The program body of every C main function
     must start with an opening brace { */

 printf(“Review”  “ Chapter 1 --- Getting\
 started!\n\n”);

/*printf() is a standard C function. The
  string to be displayed must be enclosed by
  two double quotes. The symbol \n means
  linefeed and carriage return, it represents
  one, but not two characters.

  To call a standard C function, we start with
  the function name, followed by a pair of
  parentheses. The argument value(s) to be
  passed are placed inside the parentheses.

  Please note the semicolon ‘;’ at the end of
  print(f) statements. A statement in C must be
  terminated with a semicolon.
*/
 printf(“We have learned\n\n\
 1.  How to write comments \n\
 2.  How to linefeed \n\
 3.  How to continue a line \n\
 4.  How to use tab \\t, beep \\a, and \n\
     carriage return ‘\\r’ \n\
 5.  How to print special characters %%, \\, \”\n”);
} /* The program body of a c function must end
     with  a closing brace ‘}’ */
```

Output

```
Review Chapter  1 --- Getting started!

We have learned

1.  How to write comments
2.  How to linefeed
3.  How to continue a line

4.  How to use tab \t, beep \t, and
    carriage return \r
5.  How to print special characters %, \ ,”
```

Explanation

1. *What are the important features of a C program?*

- The main function name must be main.
- The program body must start from {.
- The program body must end with }.
- A C statement must end with ;.
- A C statement is case sensitive.
- s location insensitive.
- In general, it is acceptable to add blank(s) between “words” in a C statement but not acceptable to add blank(s) within “word”.
- C uses character escape sequences which consists of a backslash followed by other character(s) to represent a special character which cannot be typed with one keystroke from the keyboard.

Exercises

1. True or False:

```
a. A C program must begin with a MAIN() statement.
b. We may use any name as the main program name.
c. A C statement is location sensitive but case
   insensitive.
d. No end mark is required to end a C statement.
e. We may use two single quotes to enclose a character
   string.
f. We use ‘/n’ to linefeed.
g. We  may use ‘/’ to continue the current line to next
   line.
```

2. Use a printf function to display the following message:

```
Please display “” %% ‘’\\ // and ‘’ in a line?

If you cannot do this, please review the chapter.
```

Solutions

```
1.   A B C D E F G
     F F F F F F F

2.
     main()
     {printf(“Please display \”\” %%%%’’ \\\\ //  and ‘’\
      in a line?”);
     }
```

Application Program 2_1

Application programs in this book are use to illustrate the value of some of the programming techniques taught in each chapter. This application program creates a letterhead and logo upon execution. You can use it as a guide to write a program to create your own letterhead and logo.

Source Code

```
main()         {
/*   This program creates a letterhead and
     border for notes */

 printf("\n\n\n\n"
 "





 \n\n\n\n");
}
```

Output

Application Exercises

1. Write a program which is capable of displaying your own name and address and a border onto the screen!
2. Design your own personal logo. Write a program which will display that logo onto the screen.