Introduction to Computer Programming with C Language

BY

Dr. EMAD SAMI

Course Chapters

5. Reading and Writing Standard I / O

Chapter Objectives:

5-1 scanf ( ) function

5-2 printf ( ) function

5-3 Adding The Minimum Field Width

5-4 Aligning Output

5-5 The Precision Specifier

5.1 scant ( ) function

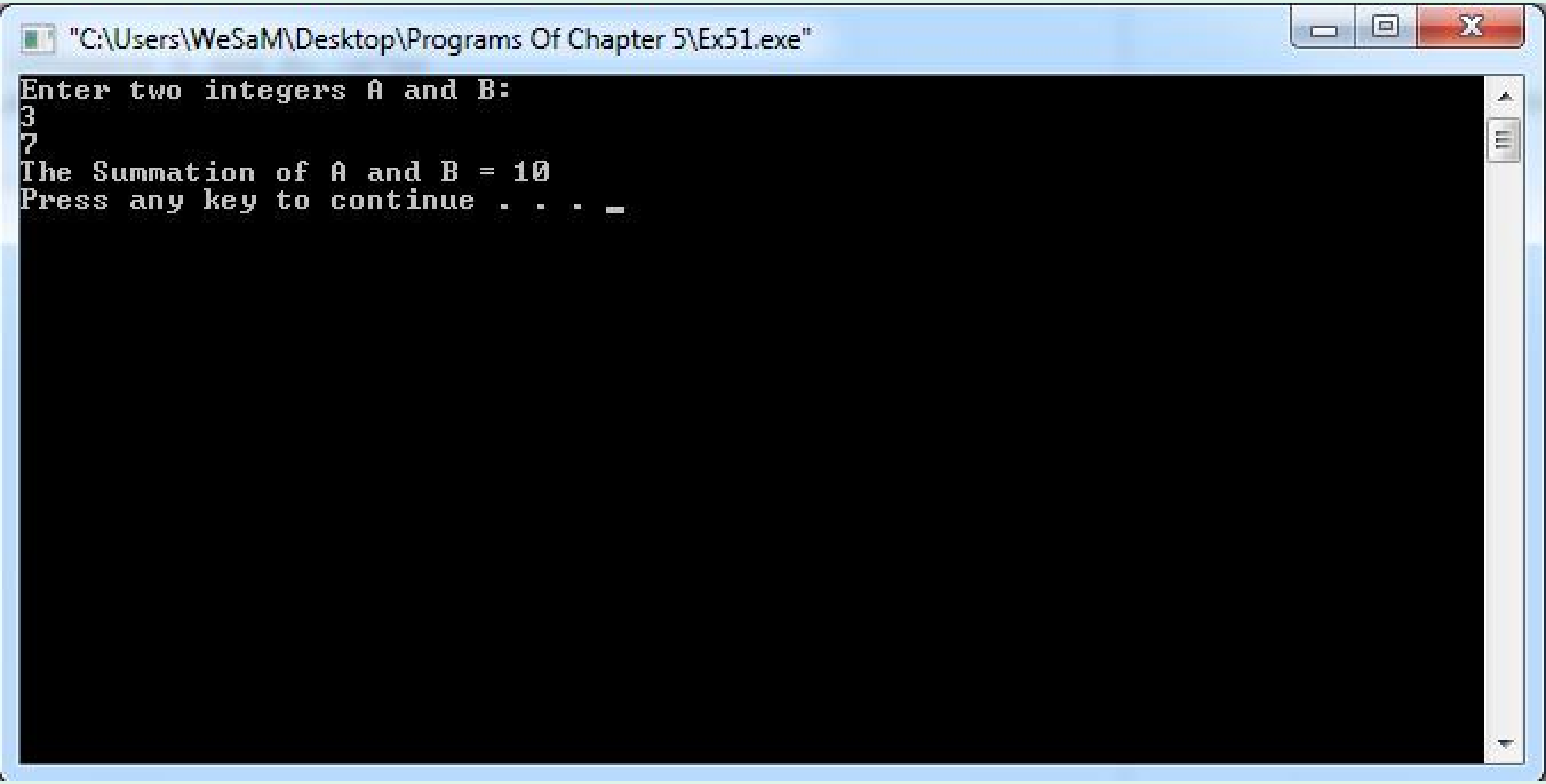
- **scanf ( )**: function that used to read data from the standard input device; the keyboard.
- The syntax for the **scanf ( )** function is

  scanf ("conversion string", &addr1, &addr2, …);

- **"conversion string"** contains specifiers: e.g., **%d** or **%f**
- **&addr1, &addr2,** … are memory addresses: e.g., **&x** or **&y**
- The values that the program read from the standard input device will be stored at these addresses.
- ***It is an error if the number of addresses not equal to the number of conversion strings.***

5.1 scanf ( ) function

```
/* Example51 : Using scanf ( ) to add A and B*/
# include <stdio.h>                /* the header file for the printf () function */
main ( )
{
int A,B,C;
printf (“Enter two integers A and B: \n”);
scanf (“%d  %d”,&A,&B);     /* all variables whose values are to be input
                             must be preceded by an & */
C=A+B;
printf ("The Summation of A and B = %d \n",C);
return 0;
}
```


"C:\Users\WeSaM\Desktop\Programs Of Chapter 5\Ex51.exe"

```
Enter two integers A and B:
3
7
The Summation of A and B = 10
Press any key to continue . . . _
```

- Be aware of that; **scanf ( )** function doesn't actually start reading the input until the Enter key is pressed.

5.2 printf ( ) function

- **printf ( )** function is used to print out messages on the screen.
- The syntax for **printf ( )** function is:

printf (“ ”);

e.g., **printf (“**Enter two integers A and B: \n**”);**

printf (“The Summation of A and B = %d \n**”,** A+B**);**

printf (“First Number %d and Second Number %d \n**”,** A,B**);**

- The format specifiers and the expressions are matched in order from left to right.
- The following are some of the format specifiers that can be used in **scanf ( )** and **printf ( )**:

%c	character format specifier
%d	integer format specifier
%f	floating-point format specifier
%e or %E	scientific notation format specifier
%s	string format specifier

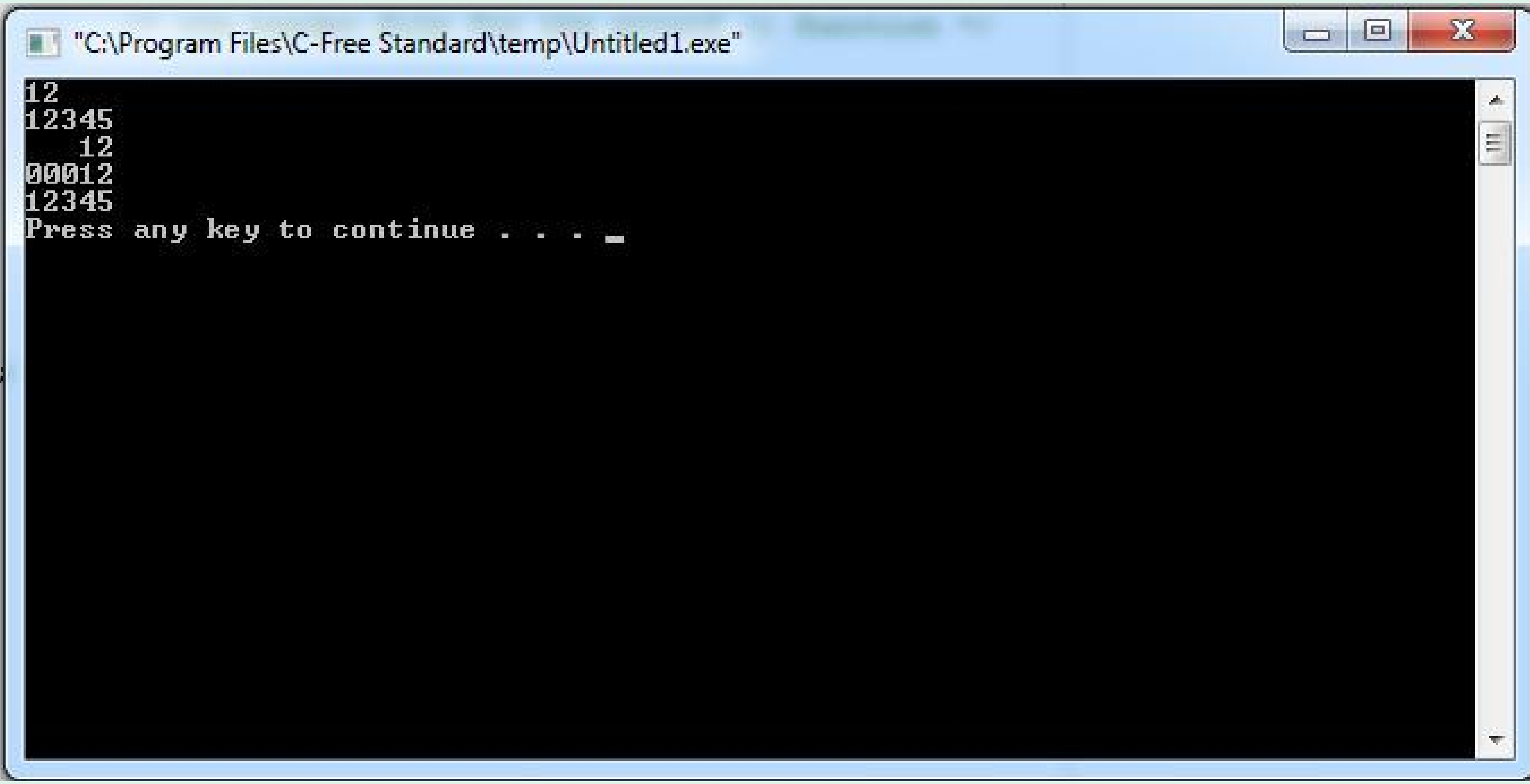
5.3 Adding the Minimum Field Width

- A integer is added between the percent sign (%) and the letter (d) in a format specifier to specify the minimum field width and ensures that the output reaches the minimum width.
- For example, in %10f, 10 is a minimum field width specifier that ensures that the output is at least 10 character spacers wide.

5.3 Adding the Minimum Field Width

The following program shows the using the Minimum Field Width specifier

```
/* Example9 : Using Minimum Field Width specifier */
# include <stdio.h>            /* the header file for the printf () function */
main ( )
{
int x, y;
x = 12;
y = 12345;
printf (“%d \n”, x);
printf (“%d \n”, y);
printf (“%5d \n”, x);          /* a minimum field width,5, is speciified*/
printf (“%05d \n”, x);         /* a minimum field width is 5 and zeros are
                               used to pad the spaces*/
printf (“%2d \n”, y);
return 0;
}
```



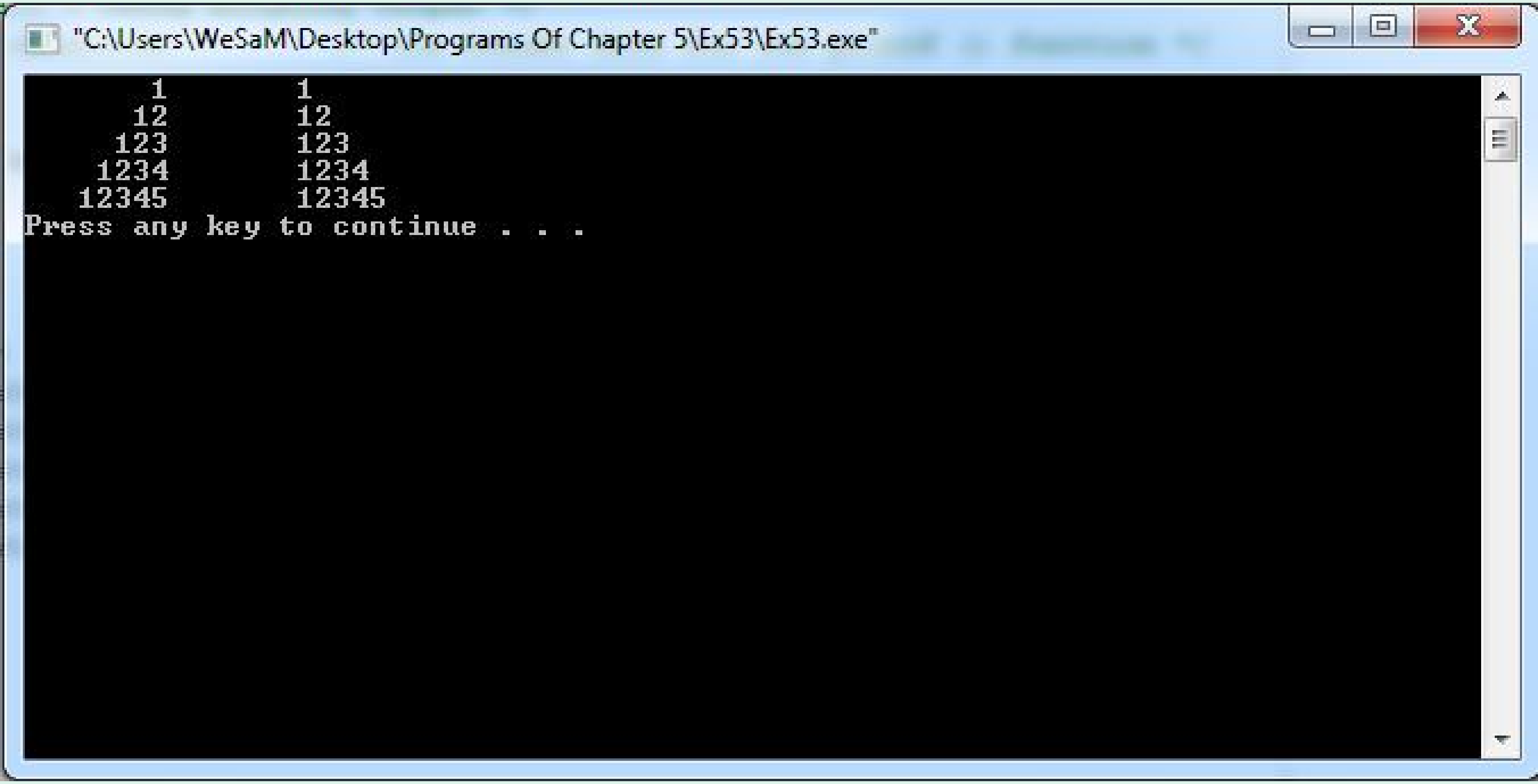
```
12
12345
   12
00012
12345
Press any key to continue . . . _
```

5.4 Aligning Output

- By default, all output is placed on the right edge of the field, as long as the field width is longer than the width of the output.
- You can change this and force output to be left-justified.
- To do so, you need to prefix the minimum field specifier with the minus sign (-).
- For example, %-12d specifies the minimum field width as 12, and justifies the output from the left edge of the field.

5.4 Aligning Output

```
/* The following program shows the using of Aligning Output */
/* Example53 : Using Aligning Output */
# include <stdio.h>                /* the header file for the printf () function */
main ( )
{
int x, y, z, m, n;
x = 1;
y = 12;
z = 123;
m = 1234;
n = 12345;
printf (“ %8d      %-8d\n”, x, x);
printf (“ %8d      %-8d\n”, y, y);
printf (“ %8d      %-8d\n”, z, z);
printf (“ %8d      %-8d\n”, m, m);
printf (“ %8d      %-8d\n”, n, n);
return 0;
}
```


"C:\Users\WeSaM\Desktop\Programs Of Chapter 5\Ex53\Ex53.exe"

```
       1        1
      12        12
     123        123
    1234        1234
   12345        12345
Press any key to continue . . .
```

5.5 The Precision Specifier

- You can put a period (.) and an integer right after the minimum field width specifier.
- The combination of the period and the integer make up a precision specifier.
- The precision specifier is another important specifier you can use to determine the number of decimal places for floating-point numbers, or to specify the maximum field width for integers or strings.
- For instance, with %10.3f, the minimum field width length is specified as 10 character long, and the number of decimal places is set to 3.
- Remember, the default number of decimal places is 6.

5.5 The Precision Specifier

- The following program shows the using of precision specifier

```
/* Example54 : Using precision specifier */
# include <stdio.h>
main ( )
{
int x;
double y;
x = 123;
y = 123.456789;
printf (“ Default integer format : %d \n”, x);
printf (“ With precision specifier : %2.8d\n”, x);
printf (“Default float format : %f \n”, y);
printf (“With precision specifier : %-10.2f\n” , y);
return 0;
}
```