Example 2.1
Representation of integer constants on a 16-bit computer.

The program in Fig.2.9 illustrates the use of integer constants on a 16-bit machine. The output in figure 2.3 shows that the integer values larger than 32767 are not properly stored on a 16-bit machine. However, when they are qualified as long integer (by appending L), the values are correctly stored.

INTEGER NUMBERS ON 16-BIT MACHINE

Program

```
     main()
     {
         printf("Integer values\n\n");
         printf("%d %d %d\n", 32767,32767+1,32767+10);
         printf("\n");
         printf("Long integer values\n\n");
         printf("%ld %ld %ld\n", 32767L,32767L+1L,32767L+10L);
      }
```

Ou

tput

```
         Integer values

         32767 -32768 -32759

         Long integer values

         32767 32768 32777
```

Fig. 2.3 Representation of integer constants

Example 2.2
Program in Figure 2.8 shows typical declarations, assignments and values stored in various types of variables.

The variables **x** and **p** have been declared as floating-point variables. Note that the way the value of 1.234567890000 that we assigned to **x** is displayed under different output formats. The value of x is displayed as 1.234567880630 under %.12lf format, while the actual value assigned is 1.234567890000. This is because the variable **x** has been declared as a **float** that can store values only upto six decimal places.

The variable **m** that has been declared as **int** is not able to store the value 54321 correctly. Instead, it contains some garbage. Since this program was run on a 16-bit machine, the maximum value that an **int** variable can store is only 32767. However, the variable **k** (declared as **unsigned**) has stored the value 54321 correctly. Similarly, the **long int** variable **n** has stored the value 1234567890 correctly.

The value 9.87654321 assigned to **y** declared as double has been stored correctly but the value is printed as 9.876543 under %lf format. Note that unless specified otherwise, the **printf** function will always display a **float** or **double** value to six decimal places. We will discuss later the output formats for displaying numbers.

EXAMPLES OF ASSIGNMENTS

Program

```
   main()
    {
    /*..........DECLARATIONS............................*/

        float      x, p ;
        double     y, q ;
        unsigned   k ;

    /*..........DECLARATIONS AND ASSIGNMENTS............*/

        int        m = 54321 ;
        long int   n = 1234567890 ;

    /*..........ASSIGNMENTS.............................*/

        x = 1.234567890000 ;
        y = 9.87654321 ;
        k = 54321 ;
        p = q = 1.0 ;

    /*..........PRINTING................................*/

        printf("m = %d\n", m) ;
        printf("n = %ld\n", n) ;
        printf("x = %.12lf\n", x) ;
        printf("x = %f\n", x) ;
        printf("y = %.12lf\n",y) ;
        printf("y = %lf\n", y) ;
        printf("k = %u  p = %f  q = %.12lf\n", k, p, q) ;
    }
```

Output

```
    m = -11215
    n = 1234567890
    x = 1.234567880630
    x = 1.234568
    y = 9.876543210000
```

```
 y = 9.876543
 k = 54321  p = 1.000000  q = 1.000000000000
```

Fig. 2.8 Examples of assignments

Example 2.3

The program in Fig.2.9 illustrates the use of **scanf** funtion.

The first executable statement in the program is a **printf,** requesting the user to enter an integer number. This is known as "prompt message" and appears on the screen like

Enter an integer number

As soon as the user types in an integer number, the computer proceeds to compare the value with 100. If the value typed in is less than 100, then a message

Your number is smaller than 100

is printed on the screen. Otherwise, the message

Your number contains more than two digits

is printed. Outputs of the program run for two different inputs are also shown in Fig.2.9.

INTERACTIVE COMPUTING USING **scanf** FUNCTION

Program

```
main()
{
    int  number;

    printf("Enter an integer number\n");
    scanf ("%d", &number);

    if ( number < 100 )
      printf("Your number is smaller than 100\n\n");
    else
      printf("Your number contains more than two digits\n");
}
```

Output

```
Enter an integer number
54
```

```
 Your number is smaller than 100

 Enter an integer number
 108
 Your number contains more than two digits
```

*Fig.2.9 Use of **scanf** function*

Example 2.4

Sample Program 3 discussed in Chapter 1 can be converted into a more flexible interactive program using **scanf** as shown in Fig.2.10.

In this case, computer requests the user to input the values of the amount to be invested, interest rate and period of investment by printing a prompt message

Input amount, interest rate, and period

and then waits for input values. As soon as we finish entering

INTERACTIVE INVESTMENT PROGRAM

Program

```
main()
{
    int    year, period ;
    float  amount, inrate, value ;

    printf("Input amount, interest rate, and period\n\n") ;
    scanf ("%f %f %d", &amount, &inrate, &period) ;
    printf("\n") ;
    year = 1 ;

    while( year <= period )
    {
          value = amount + inrate * amount ;
          printf("%2d  Rs %8.2f\n", year, value) ;
          amount = value ;
          year = year + 1 ;
    }
}
```

Output

```
Input amount, interest rate, and period

10000  0.14  5

 1  Rs 11400.00
 2  Rs 12996.00
 3  Rs 14815.44
 4  Rs 16889.60
 5  Rs 19254.15

Input amount, interest rate, and period

20000  0.12  7

 1  Rs 22400.00
 2  Rs 25088.00
 3  Rs 28098.56
 4  Rs 31470.39
 5  Rs 35246.84
 6  Rs 39476.46
 7  Rs 44213.63
```

Fig.2.10 Interactive investment program