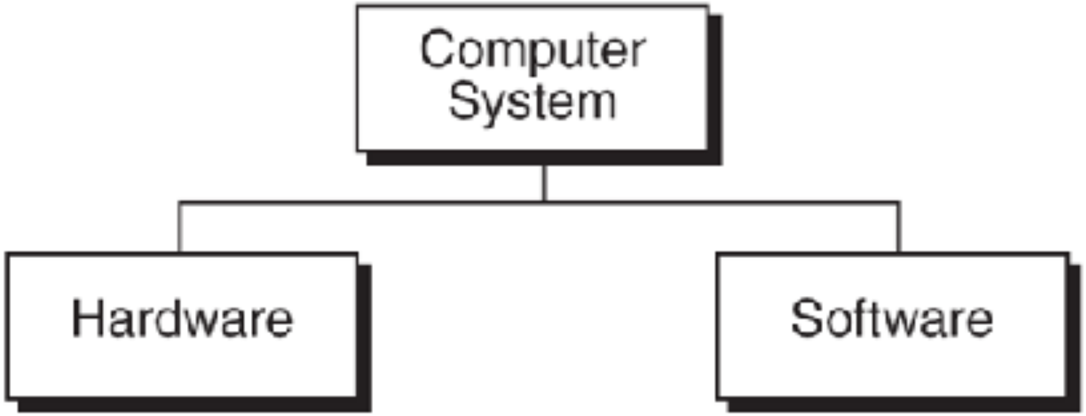


FIGURE 1-1 A Computer System

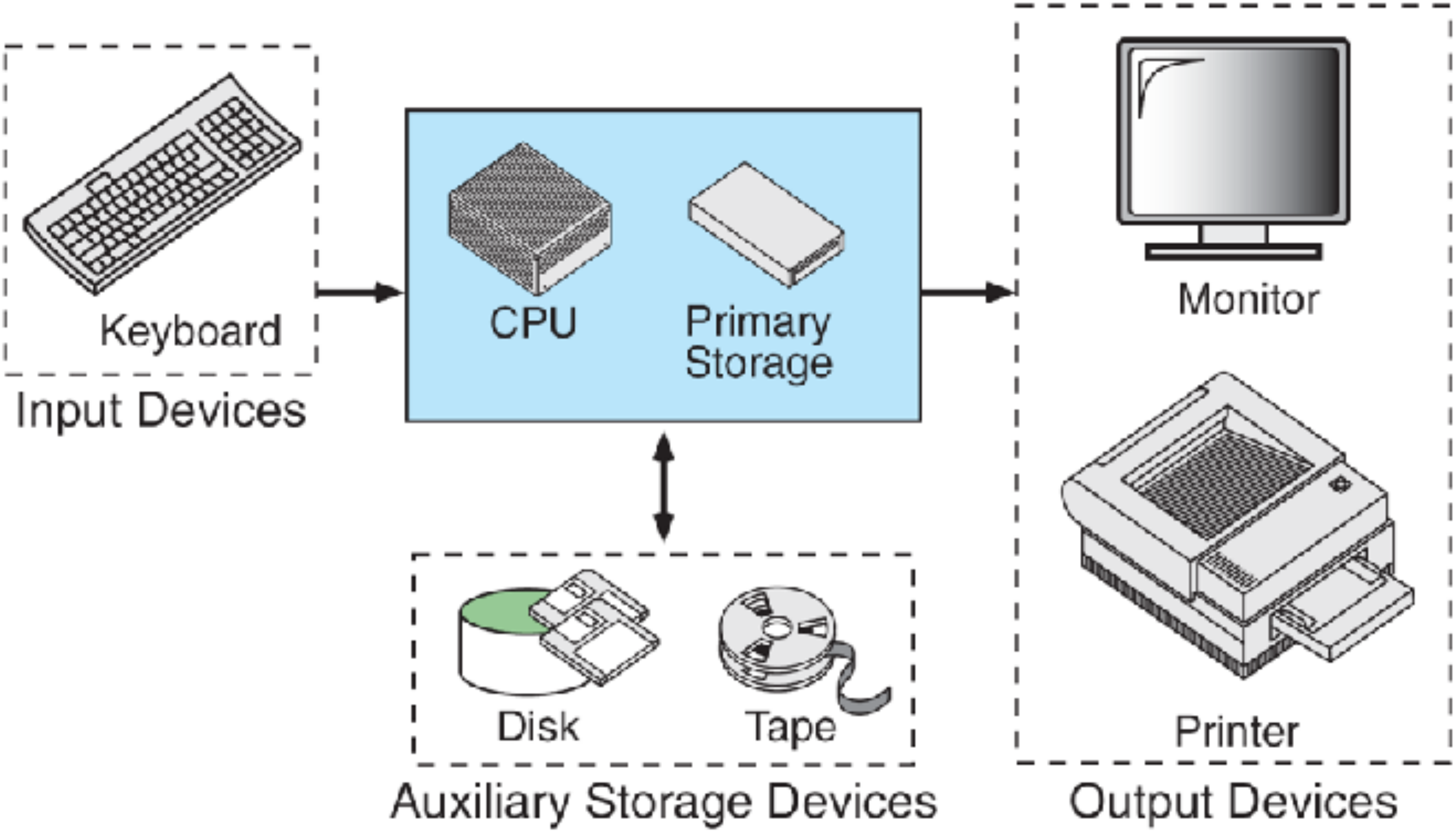


FIGURE 1-2 Basic Hardware Components

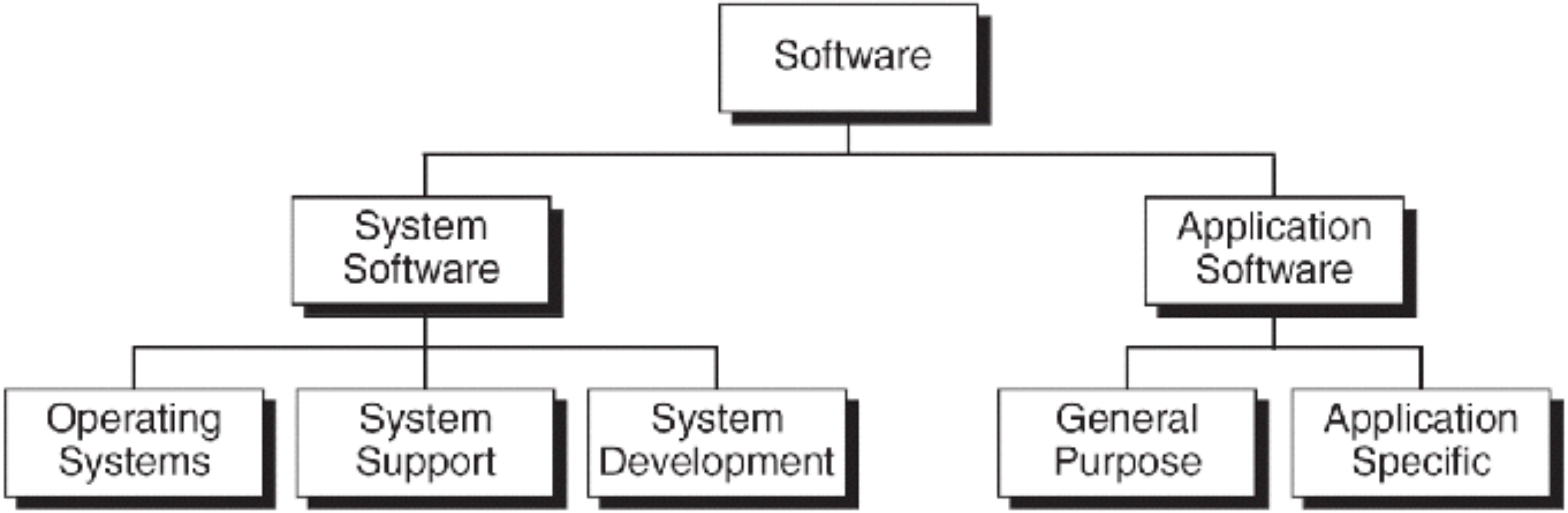


FIGURE 1-3 Types of Software

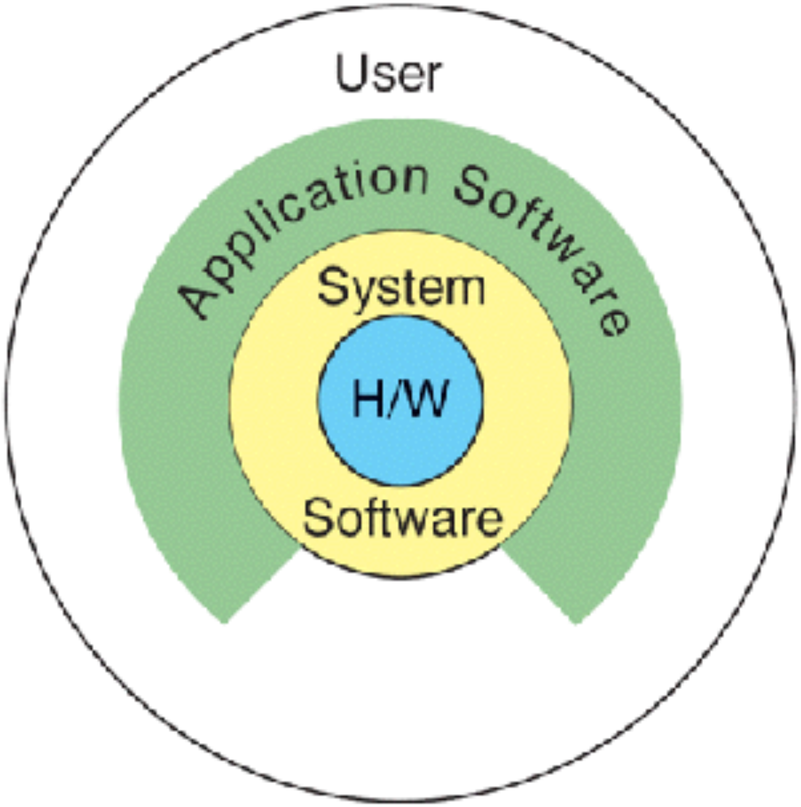


FIGURE 1-4 Relationship between system and application software

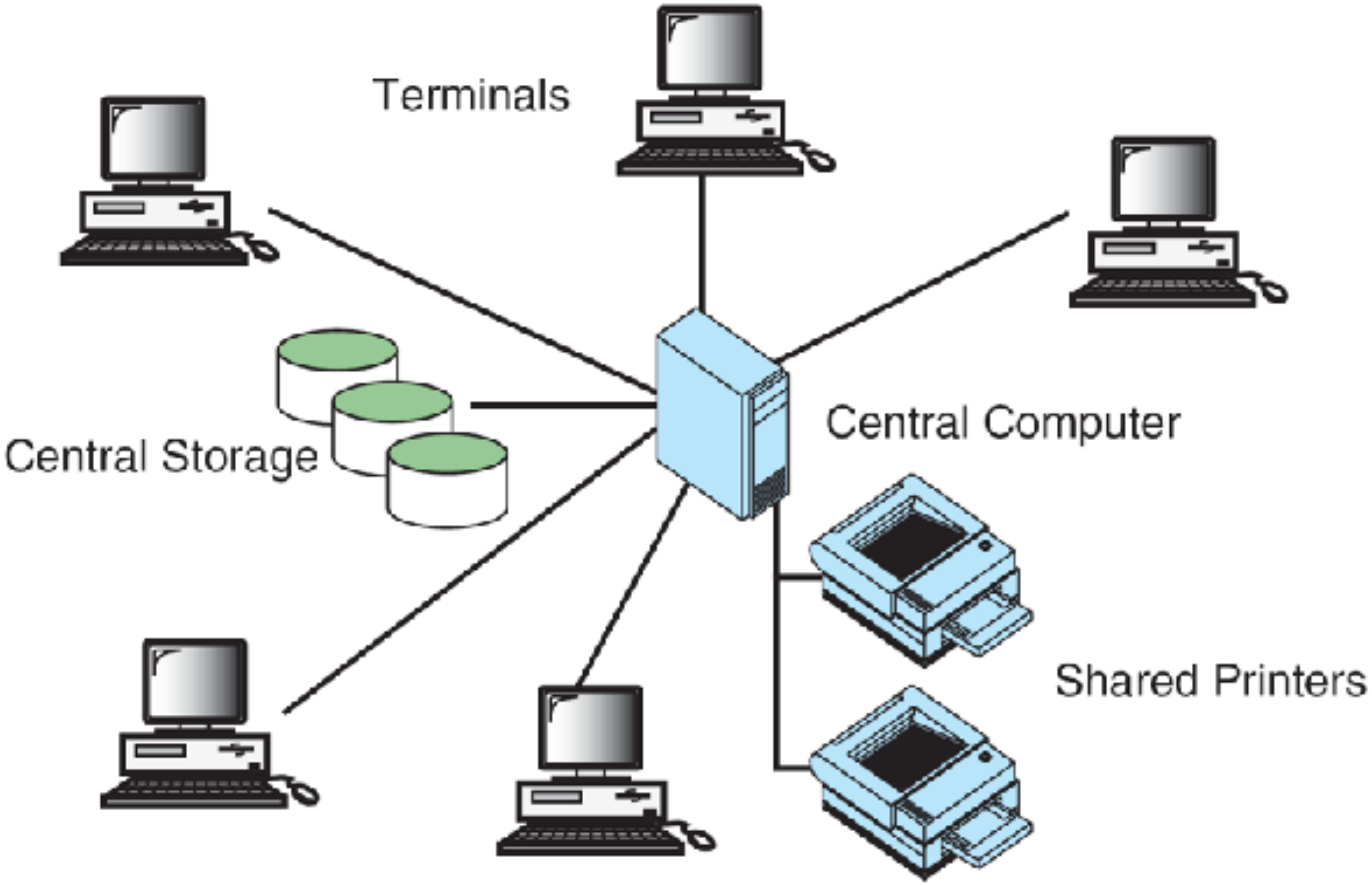


FIGURE 1-6 Time-sharing Environment

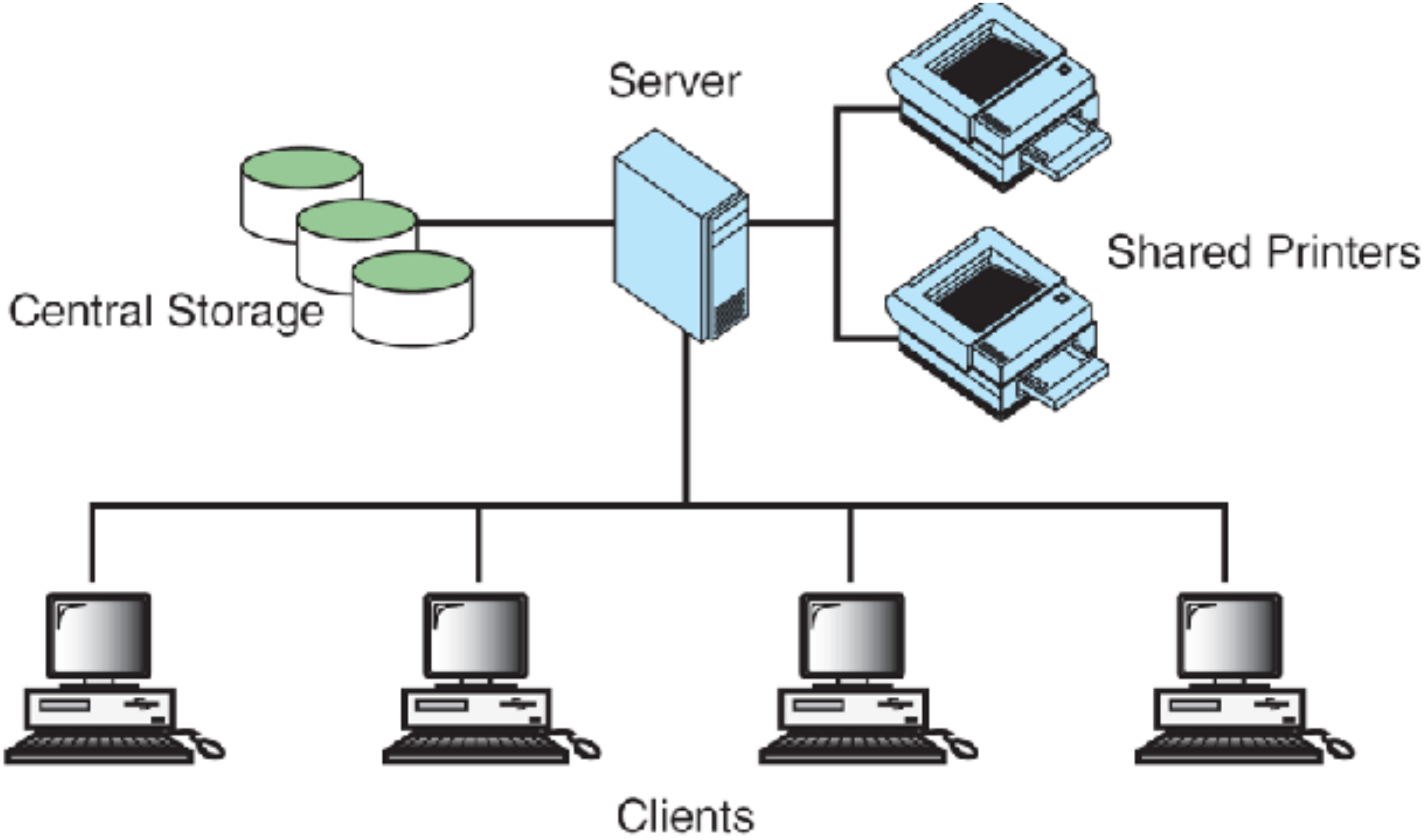


FIGURE 1-7 The Client/Server Environment

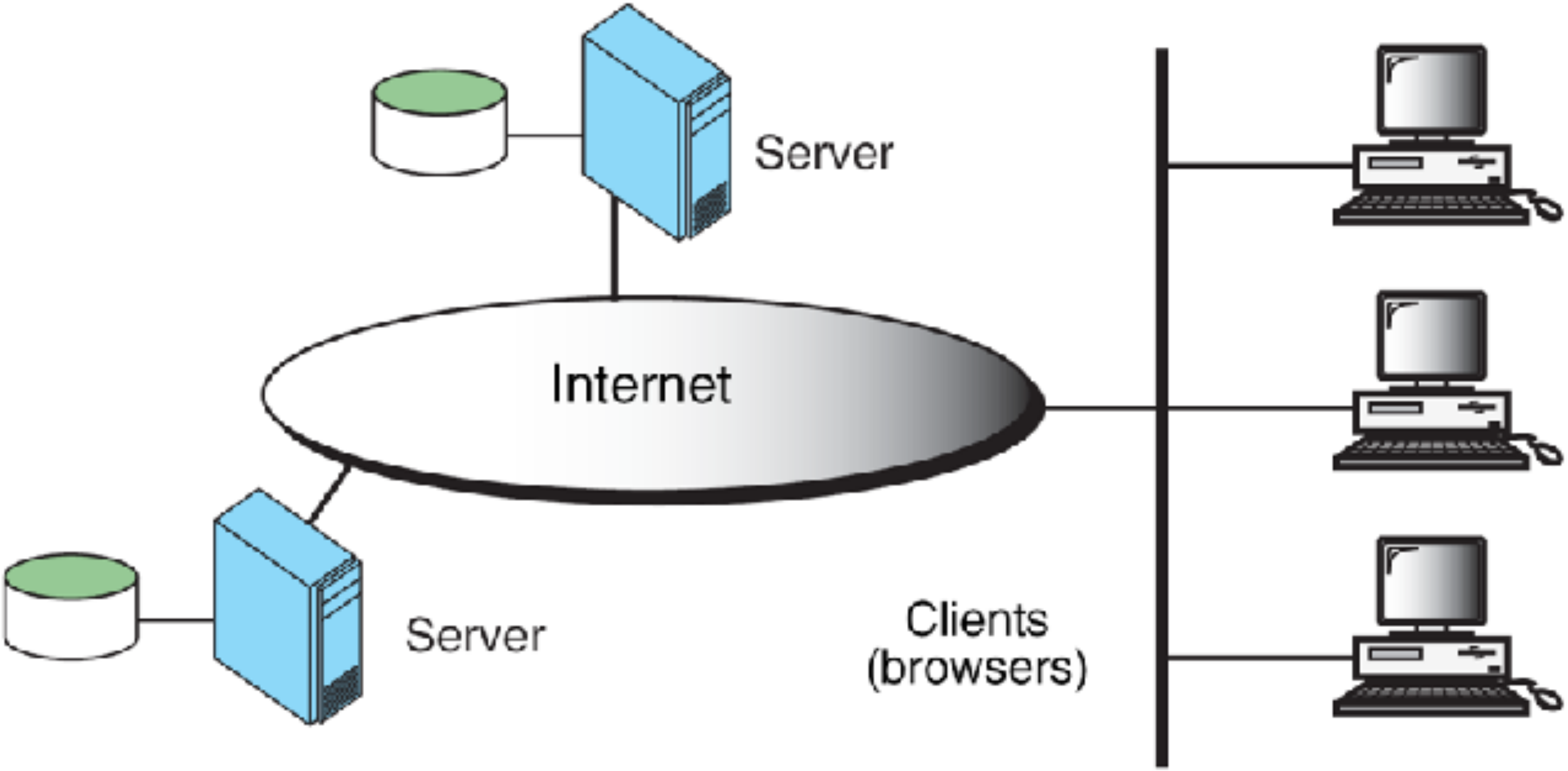


FIGURE 1-8 Distributed Computing

1-3 Computer Languages

To write a program for a computer, we must use a computer language. Over the years computer languages have evolved from machine language to natural languages.

Topics discussed in this section:

Machine Languages
Symbolic Languages
High-Level Languages

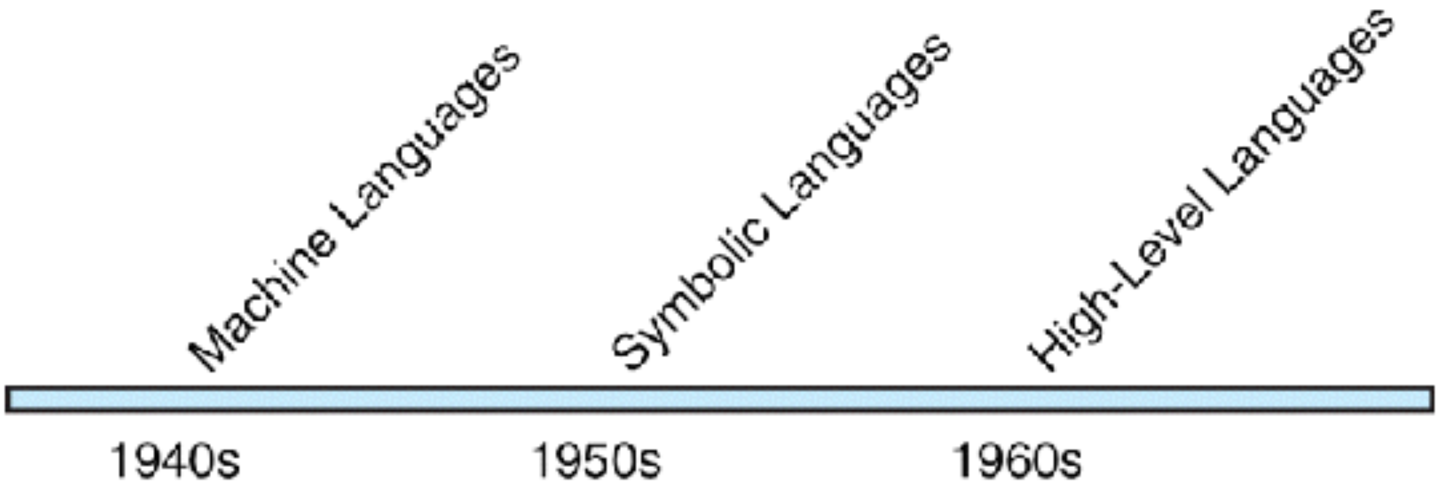
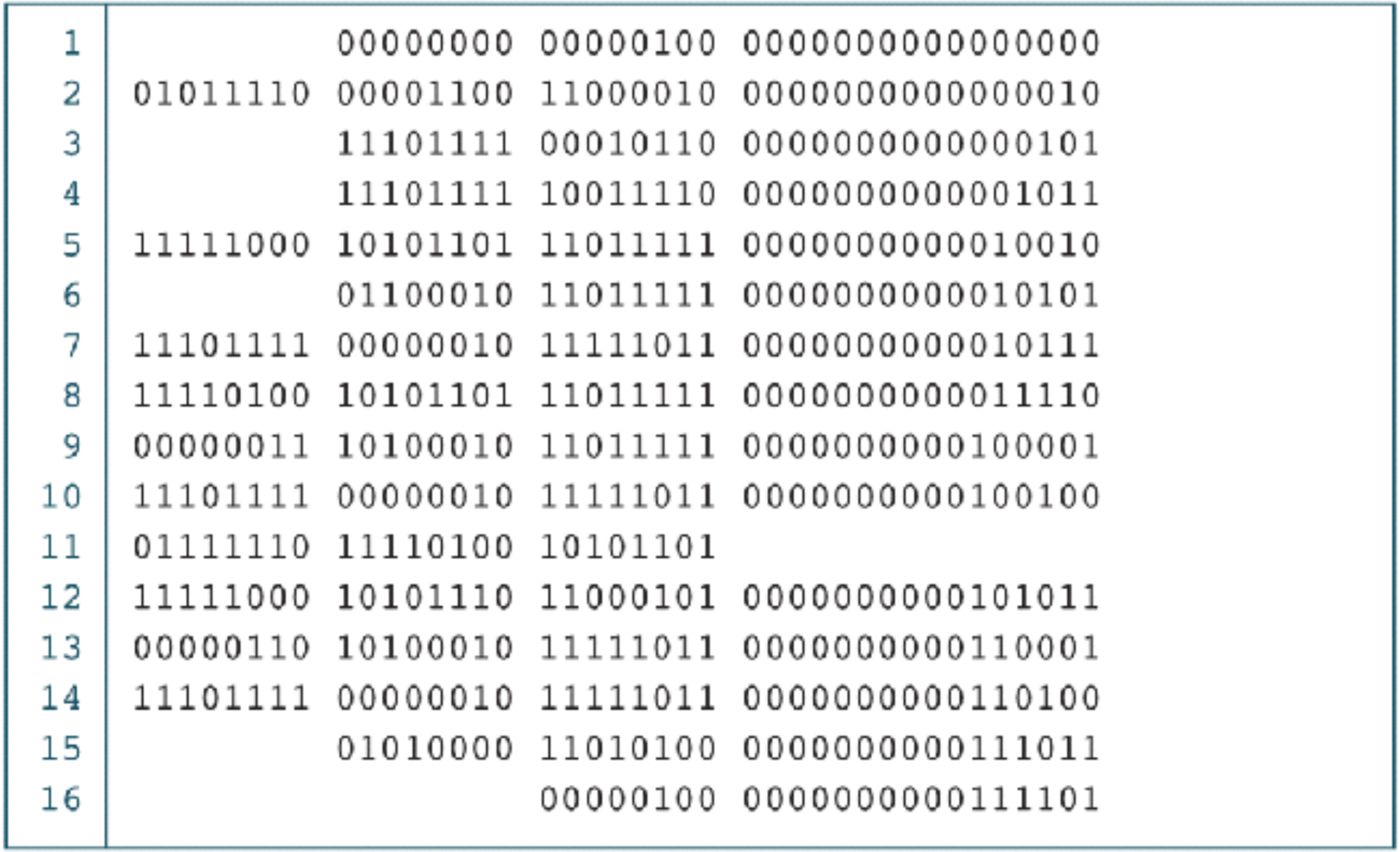


FIGURE 1-9 Computer Language Evolution

PROGRAM 1-1 The Multiplication Program in Machine Language

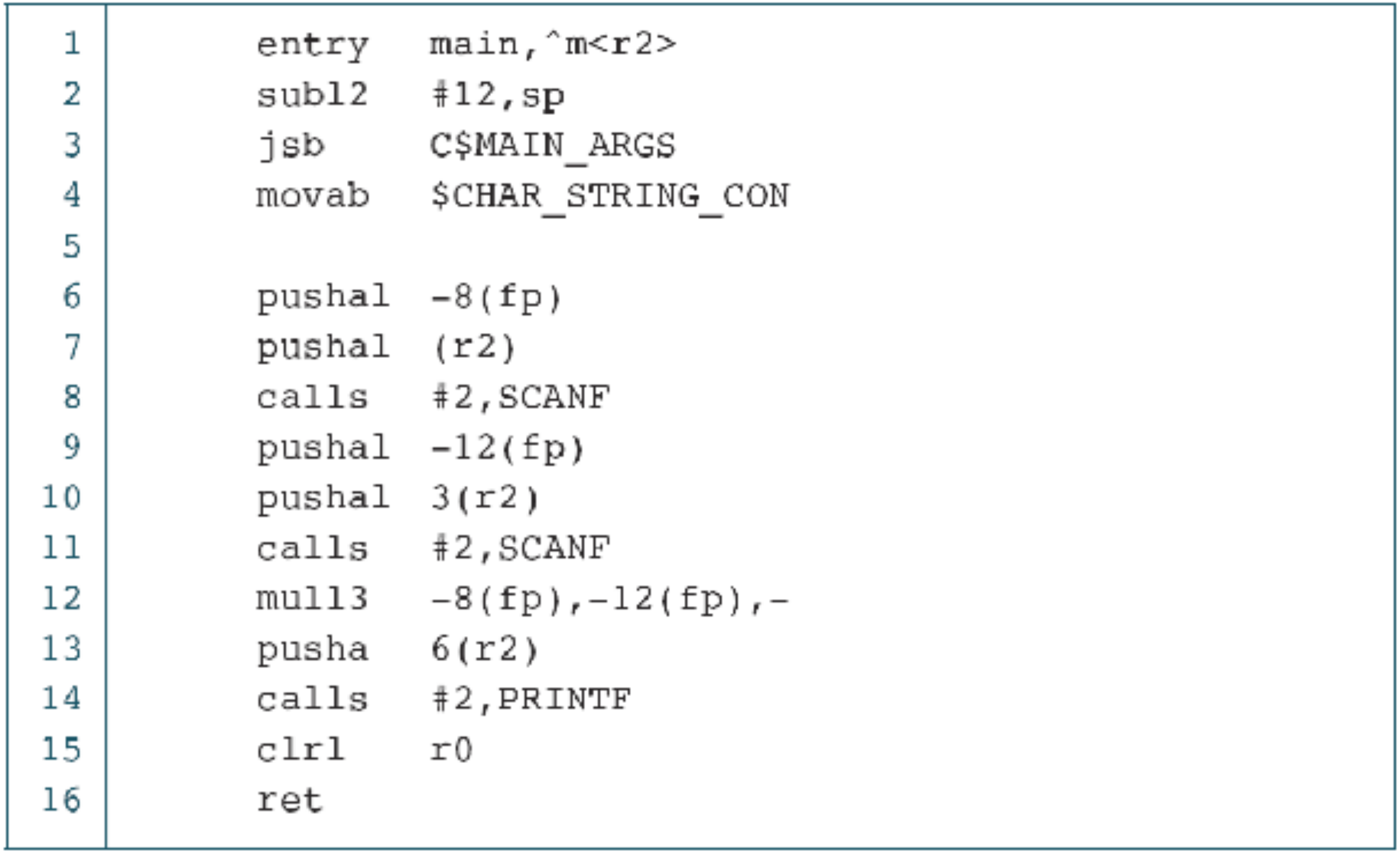


```
         00000000 00000100 0000000000000000
01011110 00001100 11000010 0000000000000010
         11101111 00010110 0000000000000101
         11101111 10011110 0000000000001011
11111000 10101101 11011111 0000000000010010
         01100010 11011111 0000000000010101
11101111 00000010 11111011 0000000000010111
11110100 10101101 11011111 0000000000011110
00000011 10100010 11011111 0000000000100001
11101111 00000010 11111011 0000000000100100
01111110 11110100 10101101
11111000 10101110 11000101 0000000000101011
00000110 10100010 11111011 0000000000110001
11101111 00000010 11111011 0000000000110100
         01010000 11010100 0000000000111011
                  00000100 0000000000111101
```

Note

The only language understood by computer hardware is machine language.

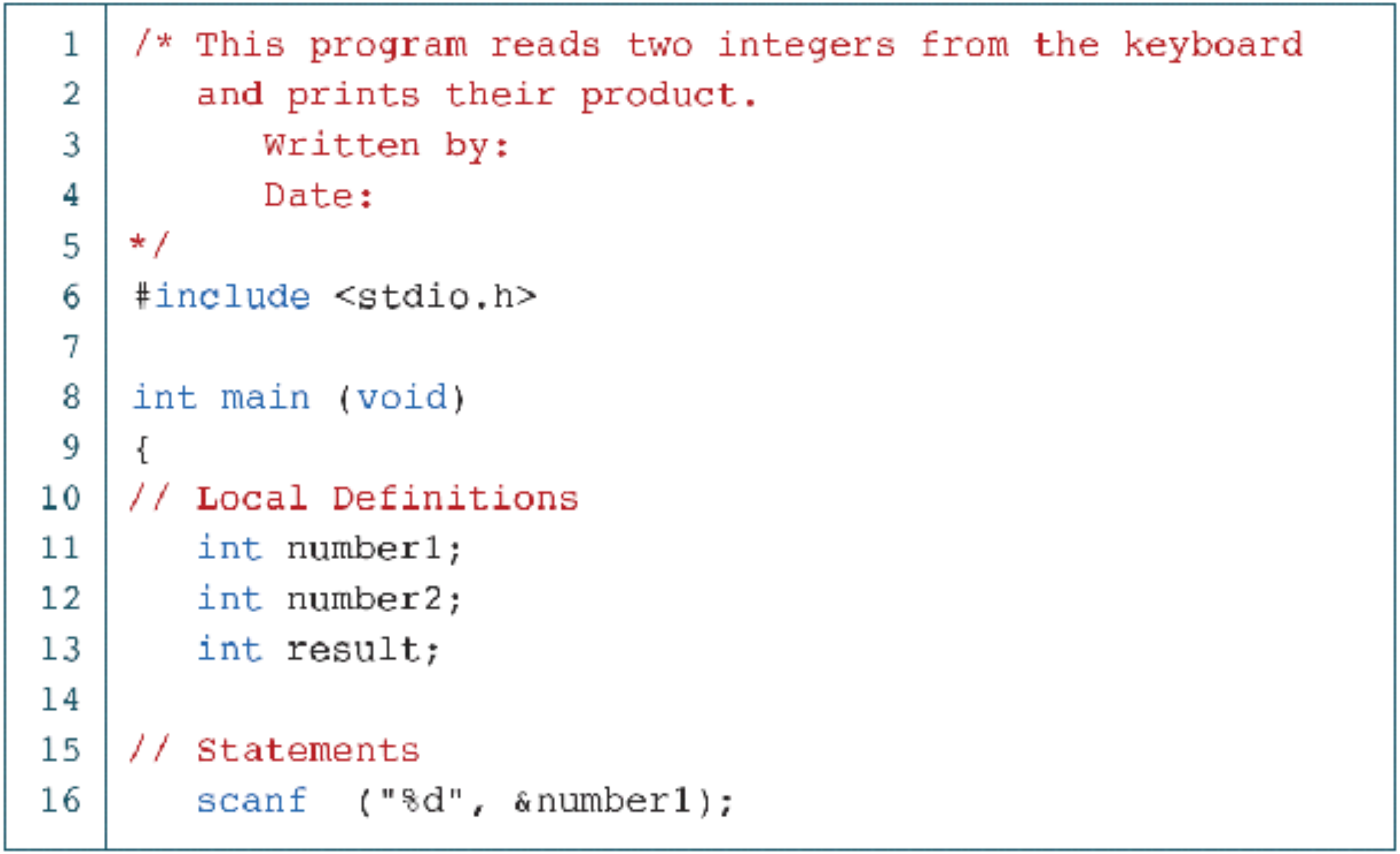
PROGRAM 1-2 The Multiplication Program in Symbolic Language



```
        entry   main,^m<r2>
        subl2   #12,sp
        jsb     C$MAIN_ARGS
        movab   $CHAR_STRING_CON

        pushal  -8(fp)
        pushal  (r2)
        calls   #2,SCANF
        pushal  -12(fp)
        pushal  3(r2)
        calls   #2,SCANF
        mull3   -8(fp),-12(fp),-
        pusha   6(r2)
        calls   #2,PRINTF
        clrl    r0
        ret
```

PROGRAM 1-3 The Multiplication Program in C



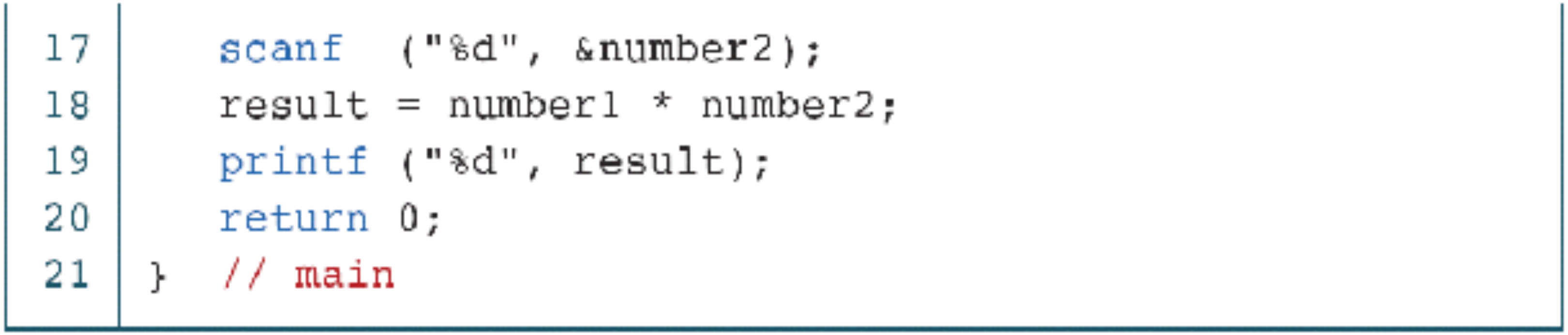
```c
/* This program reads two integers from the keyboard
   and prints their product.
      Written by:
      Date:
*/
#include <stdio.h>

int main (void)
{
// Local Definitions
   int number1;
   int number2;
   int result;

// Statements
   scanf  ("%d", &number1);
```

continued

PROGRAM 1-3 The Multiplication Program in C (continued)



```
   scanf  ("%d", &number2);
   result = number1 * number2;
   printf ("%d", result);
   return 0;
}  // main
```

1-4 Creating and Running Programs

In this section, we explain the procedure for turning a program written in C into machine language. The process is presented in a straightforward, linear fashion, but you should recognize that these steps are repeated many times during development to correct errors and make improvements to the code.

Topics discussed in this section:

Writing and Editing Programs
Compiling Programs
Linking Programs
Executing Programs

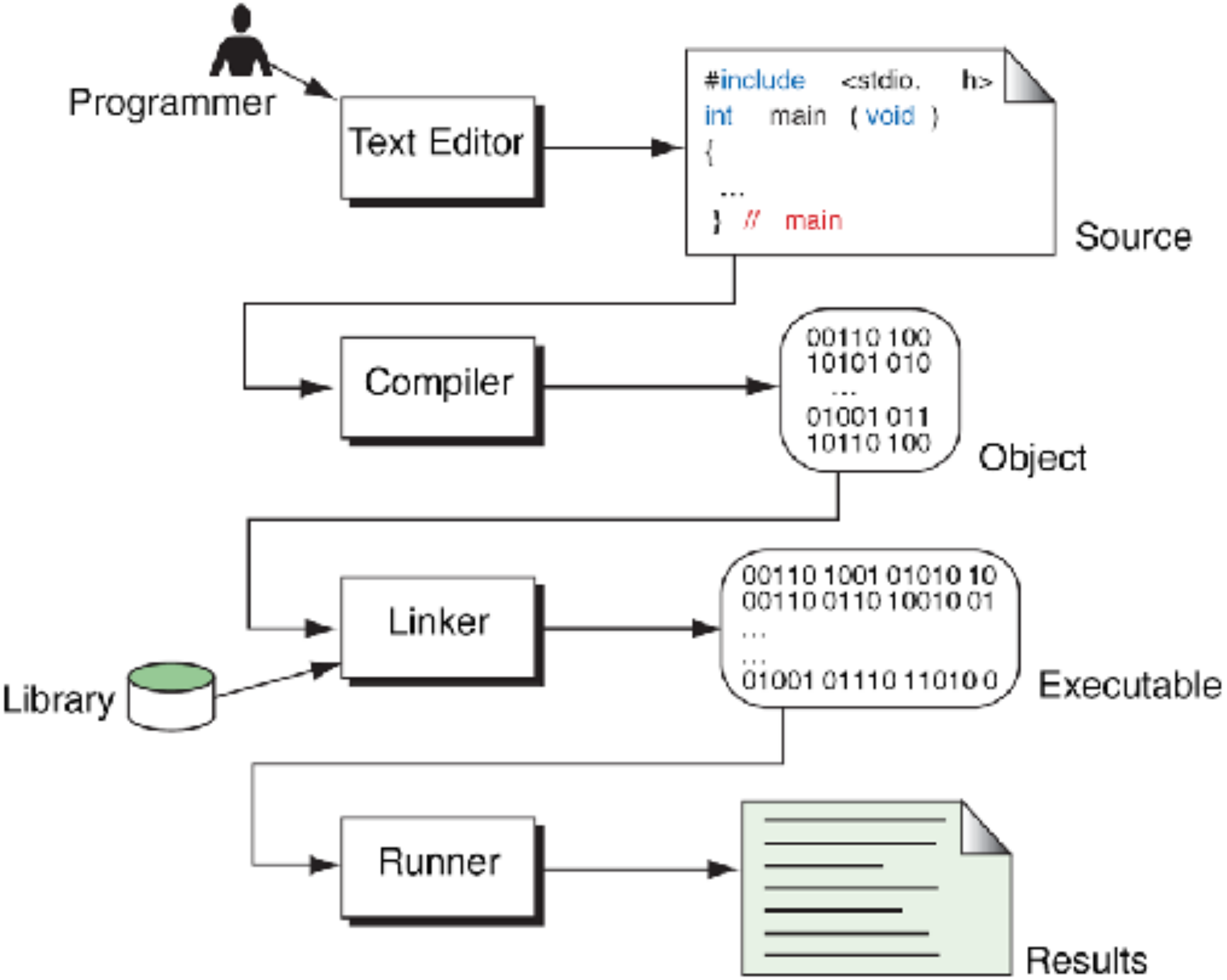


FIGURE 1-10 Building a C Program

1-5 System Development

We've now seen the steps that are necessary to build a program. In this section, we discuss how we go about developing a program. This critical process determines the overall quality and success of our program. If we carefully design each program using good structured development techniques, our programs will be efficient, error-free, and easy to maintain.

Topics discussed in this section:

System Development Life Cycle
Program Development

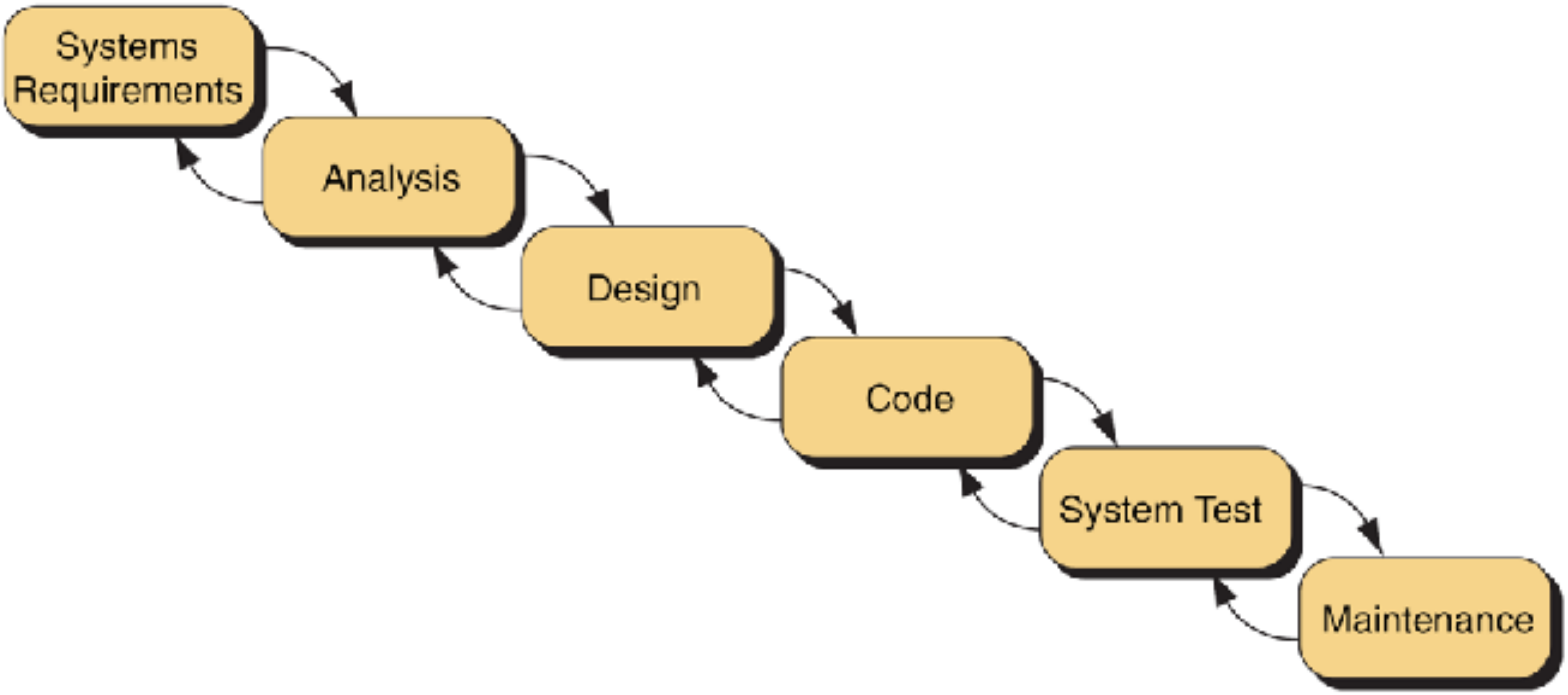


FIGURE 1-11 Waterfall Model

Note

Except for the most simple program, one set of test data will not completely validate a program.