Program 3-24

```
// This program asks for the lengths of the two sides of a
// right triangle. The length of the hypotenuse is then
// calculated and displayed.
#include <iostream>
#include <iomanip>     // For setprecision
#include <cmath>       // For the sqrt and pow functions
using namespace std;

int main()
{
    double a, b, c;

    cout << "Enter the length of side a: ";
    cin >> a;
    cout << "Enter the length of side b: ";
    cin >> b;
    c = sqrt(pow(a, 2.0) + pow(b, 2.0));
    cout << "The length of the hypotenuse is ";
    cout << setprecision(2) << c << endl;
    return 0;
}
```

Program Output with Example Input Shown in Bold

```
Enter the length of side a: 5.0 [Enter]
Enter the length of side b: 12.0 [Enter]
The length of the hypotenuse is 13
```

The following statement, taken from Program 3-24, calculates the square root of the sum of the squares of the triangle's two sides:

```
c = sqrt(pow(a, 2.0) + pow(b, 2.0));
```

Notice that the following mathematical expression is used as the `sqrt` function's argument:

```
pow(a, 2.0) + pow(b, 2.0)
```

This expression calls the `pow` function twice: once to calculate the square of `a` and again to calculate the square of `b`. These two squares are then added together, and the sum is sent to the `sqrt` function.

Random Numbers

Random numbers are useful for lots of different programming tasks. The following are just a few examples:

- Random numbers are commonly used in games. For example, computer games that let the player roll dice use random numbers to represent the values of the dice. Programs that show cards being drawn from a shuffled deck use random numbers to represent the face values of the cards.

- Random numbers are useful in simulation programs. In some simulations, the computer must randomly decide how a person, animal, insect, or other living being will behave. Formulas can be constructed in which a random number is used to determine various actions and events that take place in the program.
- Random numbers are useful in statistical programs that must randomly select data for analysis.
- Random numbers are commonly used in computer security to encrypt sensitive data.

The C++ library has a function, `rand()`, that you can use to generate random numbers. (The `rand()` function requires the `cstdlib` header file.) The number returned from the function is an `int`. Here is an example of its usage:

```
y = rand();
```

After this statement executes, the variable `y` will contain a random number. In actuality, the numbers produced by `rand()`are pseudorandom. The function uses an algorithm that produces the same sequence of numbers each time the program is repeated on the same system. For example, suppose the following statements are executed.

```
cout << rand() << endl;
cout << rand() << endl;
cout << rand() << endl;
```

The three numbers displayed will appear to be random, but each time the program runs, the same three values will be generated. In order to randomize the results of `rand()`, the `srand()` function must be used. `srand()` accepts an `unsigned int` argument, which acts as a seed value for the algorithm. By specifying different seed values, `rand()` will generate different sequences of random numbers.

A common practice for getting unique seed values is to call the `time` function, which is part of the standard library. The `time` function returns the number of seconds that have elapsed since midnight, January 1, 1970. The `time` function requires the `ctime` header file, and you pass 0 as an argument to the function. Program 3-25 demonstrates. The program should generate three different random numbers each time it is executed.

Program 3-25

```
// This program demonstrates random numbers.
#include <iostream>
#include <cstdlib>    // For rand and srand
#include <ctime>      // For the time function
using namespace std;

int main()
{
    // Get the system time.
    unsigned seed = time(0);

    // Seed the random number generator.
    srand(seed);

```

(program continues)

Program 3-25 *(continued)*

```
    // Display three random numbers.
    cout << rand() << endl;
    cout << rand() << endl;
    cout << rand() << endl;
    return 0;
}
```

Program Output

```
23861
20884
21941
```

If you wish to limit the range of the random number, use the following formula:

```
y = (rand() % (maxValue - minValue + 1)) + minValue;
```

In the formula, *`minValue`* is the lowest number in the range, and *`maxValue`* is the highest number in the range. For example, the following code assigns a random number in the range of 1 through 100 to the variable `y`:

```
const int MIN_VALUE = 1;
const int MAX_VALUE = 100;
y = (rand() % (MAX_VALUE - MIN_VALUE + 1)) + MIN_VALUE;
```

As another example, the following code assigns a random number in the range of 100 through 200 to the variable `y`:

```
const int MIN_VALUE = 100;
const int MAX_VALUE = 200;
y = (rand() % (MAX_VALUE - MIN_VALUE + 1)) + MIN_VALUE;
```

The following "In the Spotlight" section demonstrates how to use random numbers to simulate rolling dice.

In the Spotlight:

Using Random Numbers

Dr. Kimura teaches an introductory statistics class and has asked you to write a program that he can use in class to simulate the rolling of dice. The program should randomly generate two numbers in the range of 1 through 6 and display them. Program 3-26 shows the program, with three examples of program output.

Program 3-26

```
// This program simulates rolling dice.
#include <iostream>
#include <cstdlib>    // For rand and srand
#include <ctime>      // For the time function
using namespace std;

```