# Passing Multiple Arguments

When calling a function and passing multiple arguments:

- the number of arguments in the call must match the prototype and definition
- the first argument will be used to initialize the first parameter, the second argument to initialize the second parameter, etc.

**Program 6-8**

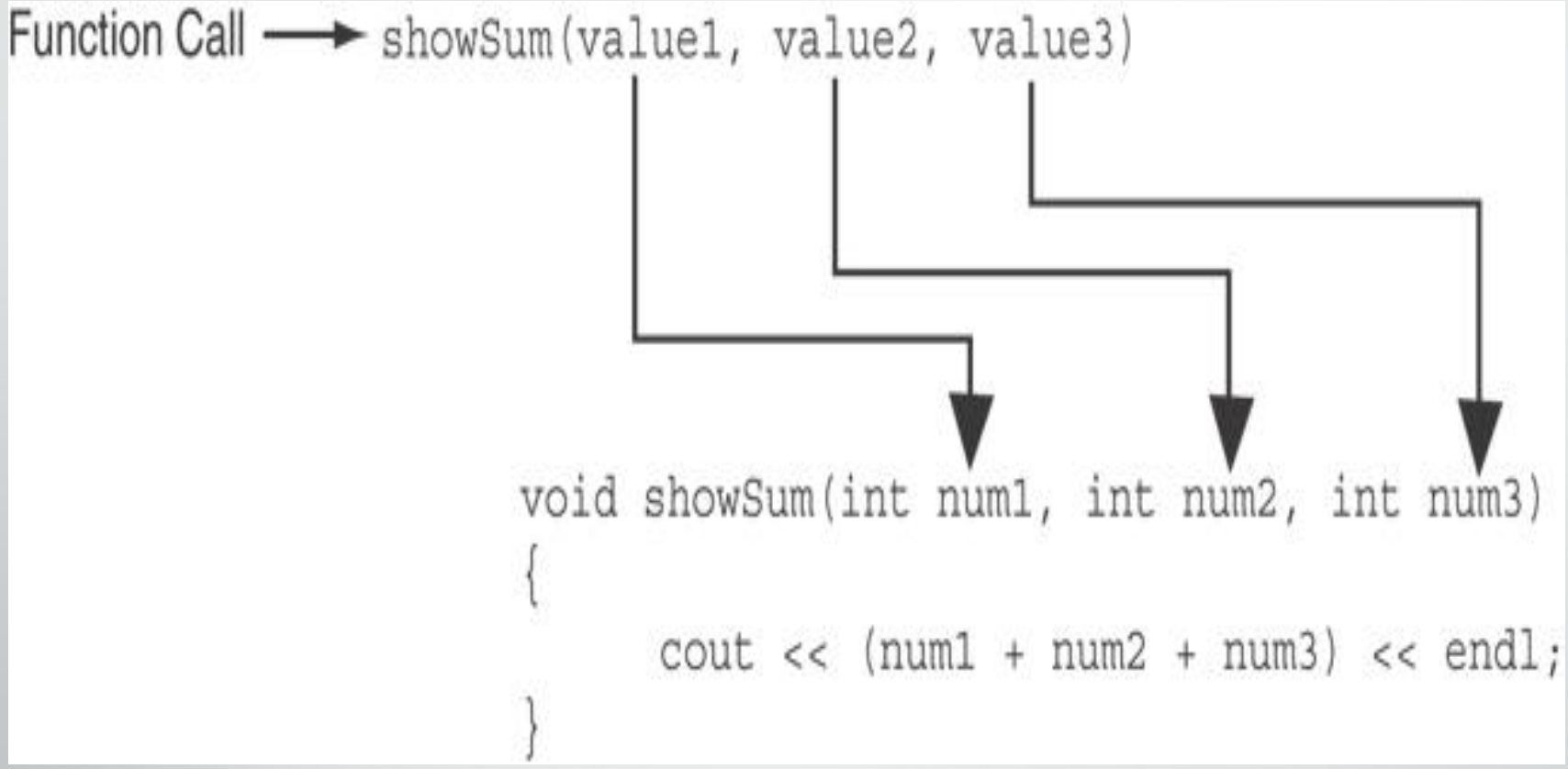
```
// This program demonstrates a function with three parameters.
#include <iostream>
using namespace std;

// Function Prototype
void showSum(int, int, int);

int main()
{
   int value1, value2, value3;

   // Get three integers.
   cout << "Enter three integers and I will display ";
   cout << "their sum: ";
   cin >> value1 >> value2 >> value3;

   // Call showSum passing three arguments.
   showSum(value1, value2, value3);
   return 0;
}

```

*(Program Continues)*

```
22 //*************************************************************
23 // Definition of function showSum.                            *
24 // It uses three integer parameters. Their sum is displayed. *
25 //*************************************************************
26
27 void showSum(int num1, int num2, int num3)
28 {
29    cout << (num1 + num2 + num3) << endl;
30 }
```

**Program Output with Example Input Shown in Bold**

```
Enter three integers and I will display their sum: 4 8 7 [Enter]
19
```




The function call in line 18 passes value1, value2, and value3 as a arguments to the function.

# The return Statement

- Used to end execution of a function
- Can be placed anywhere in a function
  - Statements that follow the `return` statement will not be executed
- Can be used to prevent abnormal termination of program
- In a `void` function without a `return` statement, the function ends at its last `}`

**Program 6-11**

```
// This program uses a function to perform division. If division
// by zero is detected, the function returns.
#include <iostream>
using namespace std;

// Function prototype.
void divide(double, double);

int main()
{
   double num1, num2;

   cout << "Enter two numbers and I will divide the first\n";
   cout << "number by the second number: ";
   cin >> num1 >> num2;
   divide(num1, num2);
   return 0;
}
```

*(Program Continues)*

```
20  //*************************************************************
21  // Definition of function divide.                            *
22  // Uses two parameters: arg1 and arg2. The function divides arg1*
23  // by arg2 and shows the result. If arg2 is zero, however, the  *
24  // function returns.                                         *
25  //*************************************************************
26
27  void divide(double arg1, double arg2)
28  {
29     if (arg2 == 0.0)
30     {
31        cout << "Sorry, I cannot divide by zero.\n";
32        return;
33     }
34     cout << "The quotient is " << (arg1 / arg2) << endl;
35  }
```

**Program Output with Example Input Shown in Bold**

```
Enter two numbers and I will divide the first
number by the second number: 12 0 [Enter]
Sorry, I cannot divide by zero.
```

# Returning a Value From a Function

- A function can return a value back to the statement that called the function.
- You've already seen the `pow` function, which returns a value:

```
double x;
x = pow(2.0, 10.0);
```

# Returning a Value From a Function

- In a value-returning function, the `return` statement can be used to return a value from function to the point of call. Example:

```
int sum(int num1, int num2)
{
  int result;
  result = num1 + num2;
  return result;
}
```

# A Value-Returning Function

Return Type

```
int sum(int num1, int num2)
{
    int result;
    result = num1 + num2;
    return result;
}
```

Value Being Returned

# A Value-Returning Function

```
int sum(int num1, int num2)
{
   return num1 + num2;
}
```

Functions can return the values of expressions, such as `num1 + num2`

**Program 6-12**

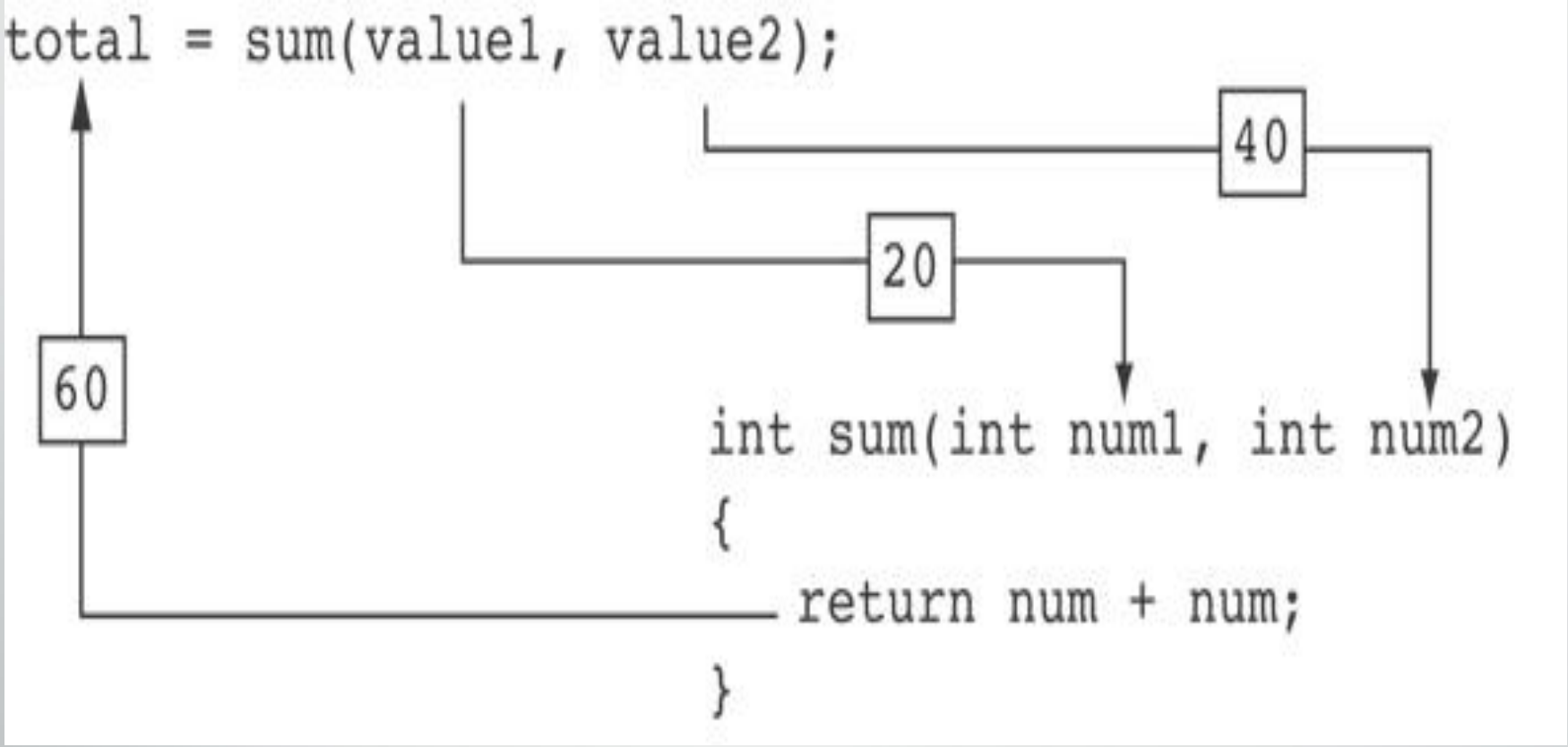
```
// This program uses a function that returns a value.
#include <iostream>
using namespace std;

// Function prototype
int sum(int, int);

int main()
{
   int value1 = 20,    // The first value
       value2 = 40,    // The second value
       total;          // To hold the total

   // Call the sum function, passing the contents of
   // value1 and value2 as arguments. Assign the return
   // value to the total variable.
   total = sum(value1, value2);

   // Display the sum of the values.
   cout << "The sum of " << value1 << " and "
        << value2 << " is " << total << endl;
   return 0;
}
```

*(Program Continues)*

```
24
25 //*****************************************************
26 // Definition of function sum. This function returns  *
27 // the sum of its two parameters.                      *
28 //*****************************************************
29
30 int sum(int num1, int num2)
31 {
32    return num1 + num2;
33 }
```

**Program Output**

```
The sum of 20 and 40 is 60
```




The statement in line 17 calls the sum function, passing `value1` and `value2` as arguments.
The return value is assigned to the `total` variable.

# Returning a Boolean Value

- Function can return `true` or `false`
- Declare return type in function prototype as `bool`
- Function body must contain `return` statement(s) that return `true` or `false`
- Calling function can use return value in a relational expression

**Program 6-15**

```
// This program uses a function that returns true or false.
#include <iostream>
using namespace std;

// Function prototype
bool isEven(int);

int main()
{
   int val;

   // Get a number from the user.
   cout << "Enter an integer and I will tell you ";
   cout << "if it is even or odd: ";
   cin >> val;

   // Indicate whether it is even or odd.
   if (isEven(val))
      cout << val << " is even.\n";
   else
      cout << val << " is odd.\n";
   return 0;
}

```

*(Program Continues)*

```
//*****************************************************************
// Definition of function isEven. This function accepts an        *
// integer argument and tests it to be even or odd. The function  *
// returns true if the argument is even or false if the argument  *
// is odd. The return value is a bool.                            *
//*****************************************************************

bool isEven(int number)
{
   bool status;

   if (number % 2 == 0)
      status = true;  // The number is even if there is no remainder.
   else
      status = false; // Otherwise, the number is odd.
   return status;
}
```

**Program Output with Example Input Shown in Bold**

```
Enter an integer and I will tell you if it is even or odd: 5 [Enter]
5 is odd.
```