User-Defined Functions: Passing Data

- Passing by Value
- Passing by Reference

Passing Data by Value

- Pass by value: when an argument is passed to a function, its value is copied into the parameter.
- Changes to the parameter in the function do not affect the value of the argument

User-Defined Functions: Passing Data by Value (cont.)

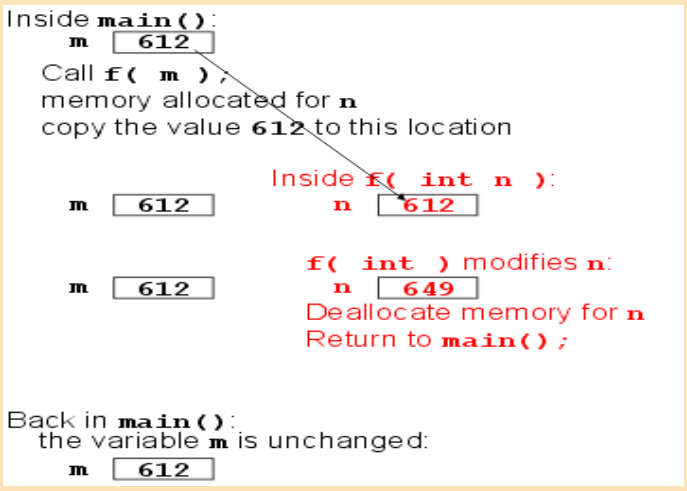
```
#include <iostream>
using namespace std;

void f( int n ) {
cout << "Inside f( int ), the value of the parameter is "
    << n << endl;
    n += 37;
cout << "Inside f( int ), the modified parameter is now "
    << n << endl;}

int main() {
    int m = 612;

    cout << "The integer m = " << m << endl;
    cout << "Calling f( m )..." << endl;
    f( m );
    cout << "The integer m = " << m << endl;
    return 0;
}
```

User-Defined Functions: Passing Data by Value (cont.)



Passing Information to Parameters by Value

- Example: `int val=5;`

```
evenOrOdd(val);
```

argument in calling function

parameter in `evenOrOdd` function

- `evenOrOdd` can change variable `num`, but it will have no effect on variable `val`

In-Class Exercise

- Do Lab 11, Exercise 1, No. 12 (pg. 152)

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 `}`

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)
{
  double result;
  result = num1 + num2;
  return result;
}
```

A Value-Returning Function

Return Type

```
int sum(int num1, int num2)
{
   double 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`

The `return` Statement - example

Program 6-11

```
1  // This program uses a function to perform division. If division
2  // by zero is detected, the function returns.
3  #include <iostream>
4  using namespace std;
5
6  // Function prototype.
7  void divide(double, double);
8
9  int main()
10 {
11    double num1, num2;
12
13    cout << "Enter two numbers and I will divide the first\n";
14    cout << "number by the second number: ";
15    cin >> num1 >> num2;
16    divide(num1, num2);
17    return 0;
18 }
```

(Program Continues)

The `return` Statement - example

Program 6-11(Continued)

Return to called function

```
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

Program 6-12

```
1   // This program uses a function that returns a value.
2   #include <iostream>
3   using namespace std;
4
5   // Function prototype
6   int sum(int, int);
7
8   int main()
9   {
10     int value1 = 20,    // The first value
11         value2 = 40,    // The second value
12         total;          // To hold the total
13
14     // Call the sum function, passing the contents of
15     // value1 and value2 as arguments. Assign the return
16     // value to the total variable.
17     total = sum(value1, value2);
18
19     // Display the sum of the values.
20     cout << "The sum of " << value1 << " and "
21          << value2 << " is " << total << endl;
22     return 0;
23  }
```

(Program Continues)

Returning Function - example

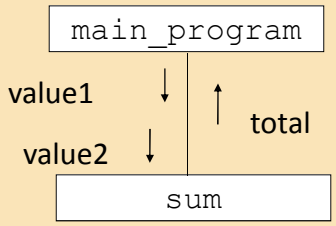
Program 6-12 (Continued)

```
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 }
```

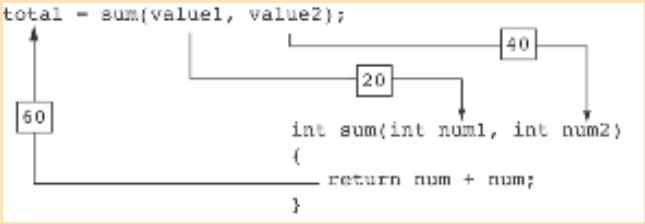
Program Output

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

Structure Chart



Returning a Value From a Function



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 Value From a Function

- The prototype and the definition must indicate the data type of return value (not `void`)
- Calling function should use return value:
  - assign it to a variable
  - send it to `cout`
  - use it in an expression

Returning a Boolean Value

- Function can return `true` or `false`
- Declare return type in function prototype and heading 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 6-14

```
1  // This program uses a function that returns true or false.
2  #include <iostream>
3  using namespace std;
4
5  // Function prototype
6  bool isEven(int);
7
8  int main()
9  {
10    int val;
11
12    // Get a number from the user.
13    cout << "Enter an integer and I will tell you ";
14    cout << "if it is even or odd: ";
15    cin >> val;
16
```

(Program Continues)

Program 6-14 *(continued)*

```
17      // Indicate whether it is even or odd.
18      if (isEven(val))
19         cout << val << " is even.\n";
20      else
21         cout << val << " is odd.\n";
22      return 0;
23   }
24
25   //*************************************************************
26   // Definition of function isEven. This function accepts an    *
27   // integer argument and tests it to be even or odd. The function  *
28   // returns true if the argument is even or false if the argument  *
29   // is odd. The return value is an bool.                       *
30   //*************************************************************
31
32   bool isEven(int number)
33   {
34      bool status;
35
36      if (number % 2)
37         status = false;    // number is odd if there's a remainder.
38      else
39         status = true;     // Otherwise, the number is even.
40      return status;
41   }
```

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.
```

In-Class Exercise

```
#include <iostream>
using namespace std;
void try1(int p);
int try3(int r);

int main()
{ int a=2;
 cout << a <<endl;
 try1(a);
 cout << a <<endl;
 int b=3;
 cout << b <<endl;
 int c=4;
 try3(c);
 cout << c <<endl;
 c=try3(c);
 cout << c <<endl;
 cout << try3(5) <<endl;
  return 0;}
```

```
void try1(int p)
{
 p++;
 cout << p <<endl;
}

int try3(int r)
{
   return r*r;
}
```

In-Class Exercise

- Do Lab 11, Exercise 2, No. 2 – Program 11.9 (pg. 159)
- Write a function prototype and header for a function named `distance`. The function should return a `double` and have a two `double` parameters: `rate` and `time`.
- Write a function prototype and header for a function named `days`. The function should return an integer and have three integer parameters: `years`, `months` and `weeks`.
- Examine the following function header, then write an example call to the function.

```
void showValue(int quantity)
```

In-Class Exercise

- The following statement calls a function named `half`. The `half` function returns a value that is half that of the argument. Write the function.

```
result = half(number);
```

- A program contains the following function:

```
int cube (int num)
{
    return num*num*num;
}
```

Write a statement that passes the value 4 to this function and assigns its return value to the variable `result`.

In-Class Exercise

- Write a C++ program to calculate a rectangle's area. The program consists of the following functions:
  - `getLength` – This function should ask the user to enter the rectangle's length, and then returns that value as a double.
  - `getWidth` – This function should ask the user to enter the rectangle's width, and then returns that value as a double.
  - `getArea` – This function should accept the rectangle's length and width as arguments and return the rectangle's area.
  - `displayData` – This function should accept the rectangle's length, width and area as arguments, and display them in an appropriate message on the screen.
  - `main` – This function consists of calls to the above functions.