Programming Language (630203)
Fall 2010/2011 – Lecture Notes # 14

User-Defined Functions I

Objectives of the Lecture

- **User-Defined Functions**
- **Value-Returning Function**
- **return statement**

User-Defined Functions

- **Value-returning functions**: have a return type
  - Return a value of a specific data type using the **return** statement
- **Void functions**: do not have a return type
  - *Do not* use a return statement to return a value
- To use these functions you must:
  - Include the appropriate header file in your program using the include statement.
  - Know the following items (**4 properties**):
    - **Name** of the function
    - Number of **parameters**, if any
    - **Data type** of each parameter
    - **Data type of the value returned**: called the type of the function
- Because the value returned by a value-returning function is unique, must:
  - **Save** the value for further calculation (in an assignment statement)
  - **Use** the value in some calculation (as a parameter in a function call)
  - **Print** the value (in an output statement)
- A value-returning function is used in an **assignment** or in an **output statement**
- One more **property** is associated with functions:
    - The **code** required to accomplish the task

```
int abs(int number)
{
      if (number < 0)
            number = -number;

      return number;
}
```

- **Heading**: first four properties above
  - **Example**: `int abs(int number)`
- **Formal Parameter**: variable declared in the heading
  - **Example**: `number`
- **Actual Parameter**: variable or expression listed in a call to a function
  - **Example**: `x = pow(u, v)`

Value-Returning Function

- **Syntax**:

```
functionType functionName(formal parameter list)
{
    statements
}
```

- **functionType** is also called the data type or **return type**
- **Syntax**: **Formal Parameter List**

```
dataType identifier, dataType identifier, ...
```

- **Function Call**

```
functionName(actual parameter list)
```

- **Syntax**: **Actual Parameter List**

```
expression or variable, expression or variable, ...
```

- Formal parameter list can be **empty**:

```
functionType functionName()
```

- A **call** to a value-returning function with an empty formal parameter list is:

```
functionName()
```

return Statement

- Once a value-returning function computes the value, the function returns this value via the return statement
  - It passes this value outside the function via the return statement
- **Syntax**: **return Statement**
- The return statement has the following syntax:

```
return expr;
```

- In C++, **return** is a reserved word
- When a return statement executes
  - Function immediately terminates
  - Control goes back to the **caller**

- When a return statement executes in the function main, the program terminates

```
double larger(double x, double y)
{
    double max;

    if (x >= y)
        max = x;
    else
        max = y;

    return max;
}
```

You can also write this function as follows:

```
double larger(double x, double y)
{
    if (x >= y)
        return x;
    else
        return y;
}
```

```
double larger(double x, double y)
{
    if (x >= y)
        return x;

    return y;
}
```

NOTE

1. In the definition of the function `larger`, `x` and `y` are formal parameters.
2. The `return` statement can appear anywhere in the function. Recall that once a `return` statement executes, all subsequent statements are skipped. Thus, it's a good idea to return the value as soon as it is computed.