Lecture 7

Chapter 3.2
Topics

- Function declaration/prototyping
- Value return
- Formal parameters
- Function definition
- Function header
- Function body
- Return statement
- Alternate function declarations
- Functions that return a Boolean
- Void functions
- Functions with no arguments
- Precondition and postcondition
Function declaration/prototyping

• Used to describe the function
  – Example
    • double areaOfRectangle(double sideA, double sideB);
    • The **double** at the beginning specifies the return type to be a double.
    • areaOfRectangle is the name of the function.
    • sideA and sideB are formal parameters passed into the function, both being doubles.
  – Function declaration is written above where the function is being used. (Above main())
Function definition

• Is where the actual function is written.
  – Example of function definition.
    double areaOfRectangle(double sideA, double sideB)
    {
      return (sideA * sideB);
    }
  – Function header is the first line describing the return type, name, and parameters.
  – Function body in this case is only one line but it is whatever is perform between the braces
  – A function always ends with a return statement (void doesn’t have to)
Alternate function declarations

• double areaOfRectangle(double sideA, double sideB);
  – or
• double areaOfRectangle(double, double);
• double totalCost(int numOfItems, float costOfItem);
  – or
• double totalCost(int, float);
Functions that return a Boolean

• Can be used to directly control an if-else expression.

• Instead of
  
  if(((rate >= 10) && (rate < 20)) || (rate == 0))

• It could read this
  
  if(appropriate(rate))

    ...

    bool appropriate(int rate)
Void functions

• A function that does not return a value.
  – Example
    • printTPSReport(0);

• These do not need a “return” command but will sometimes have them as the following.
  – return;
Functions with no arguments

- These functions are legal and usually perform a very specific action.
  - Example
    ```
    void initializeScreen()
    {
      cout << endl;
    }
    ```
    - And is used like so
      ```
      initializeScreen();
      ```
Precondition and postcondition

• Useful for describing what a functions does.
  – Precondition – states what is assumed to be true when the function is called. The function cannot be assumed to work correctly unless the precondition holds.
  – Postcondition – Describes the effects of the function call; what will be true after the function is executed. For a function the has a return value the postcondition describe the value returned.