Using Functions in C++

Instructor: Andy Abreu
Two Functions

• $\text{sqrt}( 4 );$
  – Square root function finds the square root for you
  – It is defined in the cmath library, \#include<cmath>

• \text{rand}();
  – Random function generates random value for you
  – It is defined in the cstdlib library,
    \#include <cstdlib>
sqrt() function

• sqrt function takes in a number, and returns the square root

• sqrt function is defined as
  – double sqrt( double )
  – sqrt function takes an input argument of type double
  – sqrt function returns an value that is of type double
Rand() function

- rand() function doesn’t need any input and it returns a int
- rand function is defined as
  - int rand()
  - rand function does not take any arguments
  - rand function returns a value that is of type int
Create our own functions

• Creating a function is much like declaring a variable, it has two parts...
  – Prototype
    • This gives the compiler a preview of what your function would look like
    • This usually goes after ‘using namespace std;’ and before `int main()`
  – Definition
    • This defines the actions the function should take
    • This usually goes after the main() function
Model of function prototype

```
return_type function_name( parameter_list );
```

- **return_type**
  - What the function will return

- **function_name**
  - Name of the function

- **parameter_list**
  - List of data type of parameter(s)
Model for function definition

• `return_type function_name( parameter_list )`
  
  ```
  //code goes in here
  ```
  
• `parameter_list`
  – This parameter list will include the type and the name of the variable
Example of function, reading input

- Prototype / Header:
  ```cpp
  int getNumber();
  ```

- Definition:
  ```cpp
  int getNumber() //matches above prototype/header
  {
      int num;
      cout << "Enter a number: ";
      cin >> num;
      return num;
  }
  ```
Using the function

• Calling the function:
  
  ```
  int main()
  {
      int n = getNumber();
      return 0;
  }
  ```

  /* Note the return type of the function matches the variable in which the value will be stored. */
Function’s Return Value

• Function often serve very specific purposes. In our example it was to read in a value from the user.
• This function \texttt{getNumber} need to be able to communicate this newly obtained value back to the calling function.

• It does so with a return statement.
Important Note

• This return statement is for transferring information from the sub function back to the calling function.

• The act of returning a value is done so through the keyword return. Returning a value is NOT the same as cout information to screen.

• Next example demonstrates a function that outputs to the screen, however does not return a value.
Example – output function

- Prototype / Header:
  ```
  void printNumber( int );
  ```

- Definition:
  ```
  void printNumber( int num ) //matches above
  {
    cout << num << endl;
  }
  ```

- Note the function type is void, nothing is being returned
Example of calling function

• Calling the function:

```java
int main()
{
    // gets a number from the user
    int n = getNumber();
    // prints the number to screen
    printNumber( n );
    return 0;
}
```
Why use functions

• Organizational reason
  – Sometimes we have a lot to do in our program
  – Functions offers a way to break a part a large program into smaller sub programs.
  • Think of a paragraph of text that is very long, if you lost your position, it is hard to find it again.
Why use functions

• Logical reason
  – A task might be performed repeatedly throughout different parts of the program
  – Instead of copying and pasting the same code into multiple places, we can replace that with a function.
    • If we need to make modifications it is much harder to change it in multiple places
    • Much easier to change it in just that one function
Designing of functions

• There are many different views on what is consider a well design function.
• There are even arguments on why functions should be used at all, poorly designed functions will use up a lot of system resources, when the function is called.
Designing a function – Guide Line

- Each function should do one thing, achieve one task.

- Functions should be short, not more than X number of lines long
  - X being a number that the designer sees fit and it also depends on what the function needs to accomplish.
  - Think of it as writing a paragraph, as soon as you complete presenting the idea then you are done.