Functions

Chapter 6, page 303
Functions

• “A collection of statements that perform a specific task”, p 303

  ○ And can be accessed at any point in the code through a function call and optionally produce a value

```cpp
#include <cmath>

x = pow(2.0, 3);
x = pow(4.0, 0.32);
```
Functions

• Functions are a way of building *modules* in your program

• Encapsulate some calculation

• Less repetitive code

• Example:

```cpp
x = pow(4.0, 2.2);
cout << x << endl;
```
double max (double v1, double v2) {
  double maxValue;
  if (v1 > v2) {
    maxValue = v1;
  }
  else {
    maxValue = v2;
  }
  return maxValue;
}
Calling a function

```cpp
int main()
{
    double value1, value2;

    // must match data types & parameters
    value1 = max(4.2, 2.4);
    value2 = max(value1, 2.4);

    cout << value1 << " " << value2;

    return 0;
}
```
**void** Functions

- Not all functions need to produce a value

```c
void printDayOfWeek (int day)
{
    if ( SUNDAY == day )
    {
        cout << " Sunday ";
    }
    else if (MONDAY == day )
    {
        cout << " Monday ";
    }
    // no return value!
}
```
#include <iostream>
using namespace std;

// place the function at the top of the file
void printSquares (int value, int value2) {
    cout << value * value << " ";
    cout << value2 * value2 << endl;
    return;
}

int main() {
    int x = 3, y = 2;
    printSquares(x, y);
    printSquares(y, x);
    return 0;
}
Practice

• Write a function that will calculate the average of three integers and print the result to the screen.

• What parameters do you need?

• What should the return type be?

• Write some C++ statements to call this function to determine the average of three integers given by the user.
Practice

• Write a function to calculate the factorial of a given integer.

• Remember: \( N! = n \times (n-1) \cdots 2 \times 1 \)

• Write some C++ statements to use the function to print 4! to the screen
Commenting a function declaration

/**
 * Function:    max
 * Description: finds the maximum value of two doubles
 * Parameters:  value1 – a double, first of the pair
 *               value2 – a double, second of the pair
 * Returned:    the maximum of two values given
 */

double max (double value1, double value2)
Compiling Functions

- The function declaration *must* be placed above the function's first use in the file.

```cpp
double max (double v1, double v2) // declaration
{
    . . .
    return maxValue;
}
```

```cpp
int main()
{
    double value1 = 4.2;
    cout << max(value1, 2.4); // use
    return 0;
}
```

The compiler needs to check to ensure that the function is being called with the correct data types.
Compiling Functions, part 2

• Or, the a function prototype must be given before the function is used

```cpp
double max (double v1, double v2);

int main()
{
    double value1 = 4.2;
    cout << max(value1, 2.4);
    return 0;
}
```

```cpp
double max (double v1, double v2)
{
    ...
    return maxValue;
}
```
Practice

- Write a function that flips the case of a letter. When an upper case letter is given, return the lower case version. When a lower case letter is given, return the upper case version.

- If a punctuation or numeric character is given, just return that character.

- What parameters do you need?

- What should the return type be?
Passing Arguments

• Arguments are passed into functions

• Parameters are evaluated in the order given

• A *copy* of the argument is made in the parameter

• If a parameter is changed in the function, is that reflected in main?
What will happen?

```c
void swap (int value, int value2)
{
    int tmp = value;
    value = value2;
    value2 = tmp;
    cout << value << " " << value2 << endl;
    return;
}

int main()
{
    int x = 9, y = 10;
    swap(x, y);
    cout << x << " --- " << y << endl;
    return 0;
}
```
Passing Arguments

• Pass by value
  ○ arguments are **copied** into the parameter list
  ○ changes made in the function will **not** be reflected in main

• Pass by reference
  ○ changes made in the function are reflected in the main
Example

```cpp
void swap(int &, int &);
int main(void)
{
    int i, j;
    cin >> i >> j;
    swap(i,j);
    cout << i << j;
    return 0;
}

void swap(int & num1, int & num2)
{
    int temp;
    temp = num1;
    num1 = num2;
    num2 = temp;
    return;
}
```
void changeIt(int, int&, int&);

int main()
{
    int i, j, k, l;
    i = 2;
    j = 3;
    k = 4;
    l = 5;
    changeIt(i, j, k);
    cout << i << j << k << endl;
    changeIt(k, l, i);
    cout << i << k << l << endl;
}

void changeIt(int j, int&, int& l)
{
    i++;
    j += 2;
    l += i;
}
Rules for Parameter Lists

- Same number of arguments as parameters
- Arguments & parameters are matched by position
- Arguments & parameters must have the same type
- The names of the arguments and parameters may be the same or different
- For reference parameters only, the parameter must be a single, simple variable
Example

- Given the following function prototype:

  ```c
  void checkIt(float &, float &, int, int, char &);
  ```

- And declarations in main:

  ```c
  float x, y;
  int m;
  char next;
  ```

  Which are legal?

  ```c
  checkIt(x, y, m+3, 10, next);
  checkIt(m, x, 30, 10, 'c');
  checkIt(x, y, m, 10);
  checkIt(35.0, y, m, 12, next);
  checkIt(x, y, m, m, c);
  ```
Program

• Write a function to compute the median and average of three integers, and return the two values.

• An example function call would look like:
  ◦ medianAndAverage(4, 5, 6, median, average);
bool return values

```cpp
bool isEven (int value)
{
    return (value % 2) == 0;
}

int main()
{
    int x = 9, y = 10;
    if( isEven(x) )
    {
        cout << "EVEN: " << x << endl;
    }
    if( isEven(y) )
    {
        cout << "EVEN: " << y << endl;
    }
    return 0;
}
```
Practice

• Write a function to calculate the area of a rectangle. This function should produce a value and return it to the calling function.

• Write another function to calculate the area of a circle.
  - what data type should each function return?
  - what parameters should each function accept?
Practice

- Build a small program that asks the user for either a rectangle or circle and displays the area of the selection shape. Use the functions we just defined.

- Continue asking for input until the user types something other than ‘r’ or ‘c’.