Functions

Functions calling other functions

- Write a complete C++ program that allows the user the ability to enter the numerator and denominator of a fraction. Print the fraction and the reduced fraction.
- The C++ driver for this problem is on the next slide.
- You are to write each of the function definitions for each of the function prototypes.
- You will have functions calling other functions. Ummm!!!!!

Reduced Fraction Driver

```
int minimum (int, int);
int getPositiveInt ();
int greatestCommonDivisor (int, int);
void printFraction (int, int);
void printFractionReduced (int, int);

int main()
{
  int numerator, denominator;

  numerator = getPositiveInt ();
  denominator = getPositiveInt ();

  printFraction (numerator, denominator);
  cout << " reduced is ";
  printFractionReduced (numerator, denominator);
  cout << endl << endl;

  return EXIT_SUCCESS;
}
```

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

```
void ValTest(int parm1, int parm2)
{
    parm1 = 33;
    parm2 = 44;
}

void RefTest(int &parm1, int &parm2)
{
    parm1 = 77;
    parm2 = 88;
}

int main()
{
    int val1 = 0, val2 = 0, val3 = 0, val4 = 0;

    ValTest(val1, val2);
    cout << "val1 = " << val1 << ", val2 = " << val2 << endl;

    RefTest(val3, val4);
    cout << "val3 = " << val3 << ", val4 = " << val4 << endl;

    return EXIT_SUCCESS;
}
```

Example

```
void swap(int &num1, int &num2);
int main()
{
  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;
}
```

What is the output?

```
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&
  i, 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:

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

- And declarations in main:

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

Which are legal?

```
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:
  - o **`medianAndAverage(4, 5, 6, median, average);`**

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.
  - o what data type should each function return?
  - o 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’.