Functions

CS 124 – Intro to Software Development

Macbeth – Lesson 3.2

Agenda

- Opening Prayer
- Spiritual Thought
- Q&A
  - Reminder about TestBed
  - Review Assignment
- Functions
- Looking Ahead

Spiritual Thought

<u>**Richard G. Scott**</u>

As spiritual knowledge unfolds, it must be *understood, valued, obeyed, remembered,* and *expanded.*

Reminder about TestBed

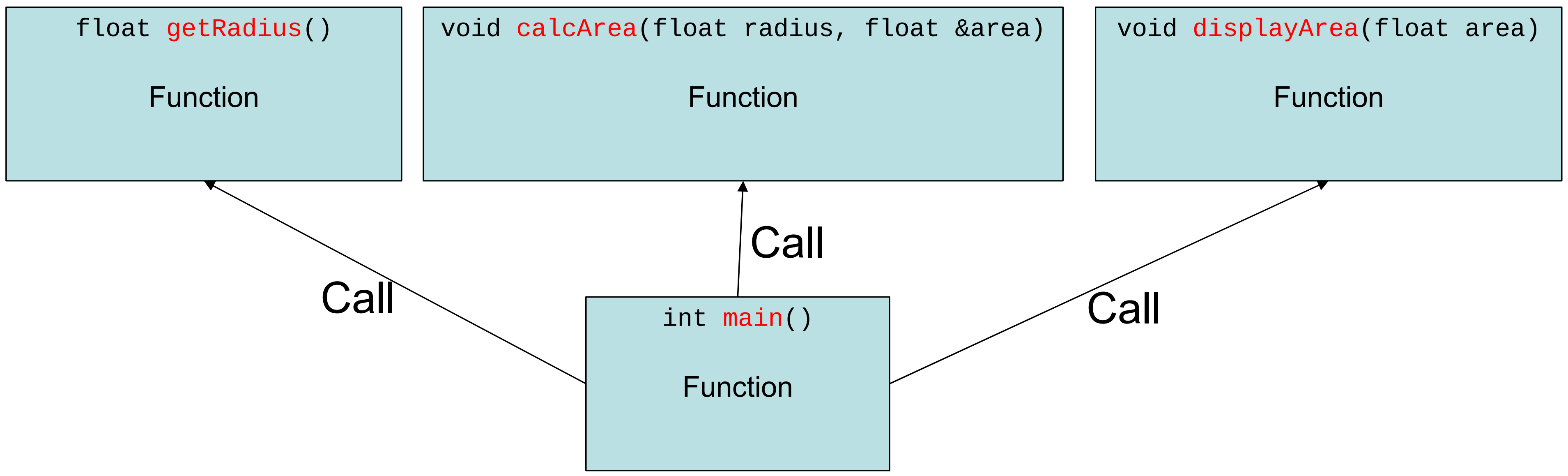
- Assignments are autograded. You get 6 points for turning it in before class and 4 points for turning it in by the weekly project due date.
- You must run TestBed before you submit your assignment.
- If TestBed fails, then you will get a 0 (its auto-graded).
- Its better to try again after you learn some more in class so you can be better prepared for the weekly project.

Review Last Assignment

- Review Last Assignment
  - What was the hardest part?
  - How did you solve it?

Functions

- Functions can simplify and organize your code.
- Think of each function as its own piece of software. When combined together in the `main` function, you can create more complex programs.



Function Declaration

- Every function is written as follows:

```
<return type> <function name> (<parameter list>)
{
      <what the function does>
      return <return value>;
}
```

- Example:

```
float convertPoundsToGrams(float pounds)
{
     float grams = 0.0;
     grams = 453.592 * pounds;
     return grams;
}
```

Functions with Return

- Functions with a `return` need to be used by the calling function.
- The value can be displayed with `cout` or set to variable using the equal sign.

```
float convertPoundsToGrams(float pounds)
{
   float grams = 0.0;
   grams = 453.592 * pounds;
   return grams;
}

int main()
{
   float weightGrams = 0.0;
   weightGrams = convertPoundsToGrams(100.0);
   cout << "100 lbs = " << weightGrams << " grams\n";
   return 0;
}
```

Float value is returned

Functions with No Return

- If the function has no return value, then use the keyword `void`.
- If there is no return value, then the `return` keyword is not needed

```
void printMoney(float money)
{
   cout.precision(2);
   cout.setf(ios::fixed);
   cout.setf(ios::showpoint);
   cout << "Money = $" << money << endl;
   // No return needed
}

int main()
{
   float cash = 213.25;
   printMoney(cash);   // No equal sign needed
   return 0;
}
```

Function Parameters

- The return value allows you to send back <u>one</u> result
- You can send <u>more</u> results by using function parameters (separated by commas)
- Each parameter is treated like a variable for the function with a data type and name

```
float calcVolume(float height, float width, float depth)
{
   float volume = height * width * depth;
   return volume;
}

int main()
{
   float boxVolume = 0.0;
   float boxHeight, boxWidth, boxDepth;
   cout << "Enter dimensions of the box (h w d): ";
   cin >> boxHeight >> boxWidth >> boxDepth;
   boxVolume = calcVolume(boxHeight, boxWidth, boxDepth);
   cout << "Volume of the box = " << boxVolume << endl;
   return 0;
}
```

Pass the 3 values to the function. The compiler takes care of the mapping.

Two Kinds of Parameters

	Pass by Value	**Pass by Reference**
What is passed to the function?	A copy of the value in the variable	A reference to the actual variable
What happens if the value changes in the function?	The variable in the calling function is <u>not</u> updated.	The variable in the calling function is updated.
How do I declare the parameter?	`void foo(int x, bool y, float z)`	`void foo(int &x, bool &y, float &z)`

Example

Pass by Value

```
void add(int x, int y, int result)
{
   result = x + y;
}

int main()
{
   int x = 3;
   int y = 4;
   int result = 0;
   add(x, y, result);
   cout << "Result = " << result;
}
```

`Result = 0`

Pass by Reference

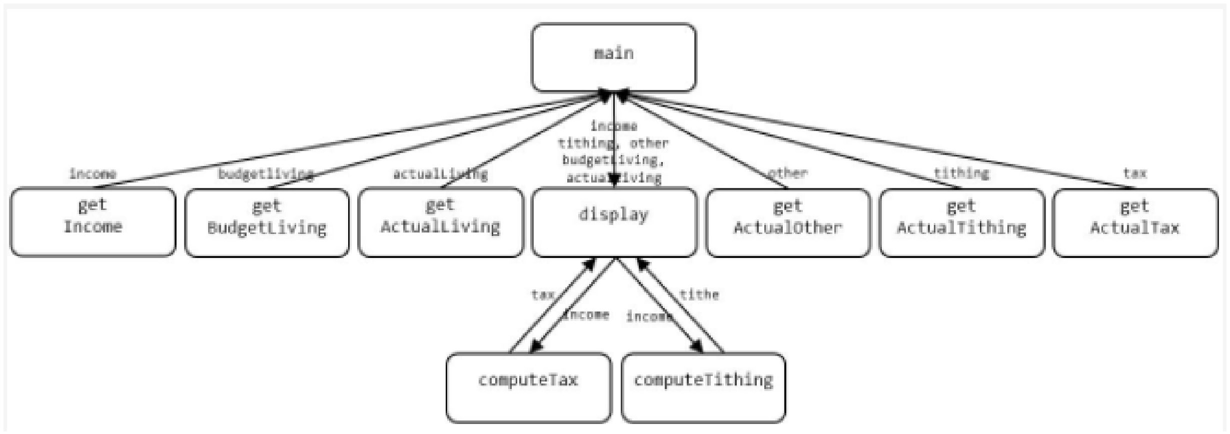
```
void add(int x, int y, int &result)
{
   result = x + y;
}

int main()
{
   int x = 3;
   int y = 4;
   int result = 0;
   add(x, y, result);
   cout << "Result = " << result;
}
```

`Result = 7`

BYU IDAHO

Project 03

- Make a copy of project 02.
- Create functions to match the structure chart shown in the project



- Read (and re-read) all the instructions and notes in the project.

Project 03

- Create the following functions:
  - getIncome – Use cout and cin to return the income (budget and actual)
  - getBudgetLiving – Use cout and cin to obtain and return the living budget
  - getActualLiving – Use cout and cin to obtain and return the living actual
  - getActualOther – Use cout and cin to obtain and return the other actual
  - getActualTithing – Use cout and cin to obtain and return the tithing actual
  - getActualTax – Use cout and cin to obtain and return the tax actual
  - computeTax – Takes the income as an input. For this project, return 0.0
  - computeTithing – Takes the income as an input. Returns 10% of the income.
  - display –
    - Takes the income, living budget, living actual, tax actual, tithing actual, and other actual as inputs.
    - Calls computeTax and computeTithing.
    - Calculates the actual difference (the assignment has the formula). Note that the budget difference is still 0.0 for this project.
    - Display the table
  - main – Modify it to call the "get" functions above and then call the display function passing the values returned from the "get" functions.

Looking Forward

- Before Class on Friday
    - 1.5 Prepare
        - Read Chapter 1.5 Boolean Expressions
        - Submit assign15
    - 03 Ponder – Start on your project that is due Saturday
        - You have everything you need to know to finish the project!

- Extra Practice (optional)
    - Write the code for the 3 functions `getRadius`, `calcArea`, and `displayArea` from slide 5. Make sure you use the parameters listed on the slide. Some are pass by value and some are pass by reference. In your main function, call these functions to test what you did.