Functions for All Subtasks

CS 211

Adam Sweeney

September 19, 2017

Wichita State University, EECS

Introduction

- Functions don't always have to return by values
- There was a reason I kept making sure that the term 'pass by value' was defined

Agenda

- Chapter 5 (5.1 - 5.2)
- `void` functions
- Pass by Reference

`void` Functions

`void` **Functions**

- So far, all functions have returned a value
- This is not a requirement of a function
- It is possible for functions to do things, but not need to return a value
- These are `void` functions

An Example

```
void showResults(double degFahrenheit, double degCelsius)
{
    using namespace std;

    cout.setf(ios::fixed);
    cout.setf(ios::showpoint);
    cout.precision(1);

    cout << degFahrenheit << " degrees Fahrenheit is equivalent to\n"
         << degCelsius << " degrees Celsius.\n";

    return;
}
```

Notes on the Example

- It can be seen that the function, as written, does not do a calculation
- Whether it did or not does not change the fact that the intention of the function was not to return the converted value, but to simply output some text
- The return statement simply ends with a semicolon, remember, it does not return anything
- EVERY time we declare or define a function, the data type is required; since we are not returning anything, we put `void`

How to Use `void` Functions

```
int main()
{
    double fahr, cels;
    .
    .
    showResults(fahr, cels);
    .
    .
    return 0;
}
```

How to Use `void` Functions (cont.)

- `void` functions do not return a value
- Since the function doesn't give anything back, we don't have to worry about putting output anywhere
- Again, because there is no output
- So, we can simply call the function
- Don't forget the semicolon!

`void` & `return`

- Functions that return a value are required to have a `return` statement
- It's how the function gets its output to you
- `void` functions are not required to have a `return` statement
- If you do use one, it is simply `return;`
- `return` simply serves as a means of exiting the function
- Comes in handy if your `void` function branches and needs to terminate early in certain situations

Functions Calling Functions

- This is not a new concept
- We write our programs within the main **function**
- So naturally, it follows that a function we write can also call a function within its body
- `https://repl.it/LRVD/3`

Pass by Reference

Pass by Reference

- Pass by Value creates a copy of the variable
- Pass by Reference does not
- When you pass arguments by reference, you are giving the function your actual variable
- `https://repl.it/LR1B/2`

How’s That Now?

- Recall our simple diagram of computer memory
- These variables exist in memory, and every byte of memory has an address
- When we pass by reference, we actually give the function the address of our variable, and not a copy of the value
- Note the & used in the `changeReference()` function
- This is called the ’address of’ operator, or reference operator
- This operator is what passes our arguments by reference, and not by value

Mix'n'Match

- You are free to mix and match pass by reference and pass by value parameters in your function definitions