Lab # 12: Functions - II

EC-102 – Computer Systems and Programming

Usman Ayub Sheikh

School of Mechanical and Manufacturing Engineering (SMME),
National University of Sciences and Technology (NUST)

November 5, 2015

Outline

1. Returning Values from Functions
   - The `return` Statement
   - Eliminating Unnecessary Variables
2. Returning Structure Variables
3. Exercises
   - Exercise 1
   - Exercise 2
   - Exercise 3

Returning Values from Functions

- When a function completes its execution, it can return a single value to the calling program
- Usually this value consists of an answer to the problem the function has solved

Returning Values from Functions

```
// demonstrates return values, converts pounds to kg
#include <iostream>
using namespace std;

float lbstokgs(float); // declaration

int main()
{
    float lbs, kgs;
    cout << "Enter your weight in pounds: "; cin >> lbs;
    kgs = lbstokgs(lbs);
    cout << "Your weight in kilograms is: " << kgs << endl;
    return 0;
}

float lbstokgs(float pounds)
{
    float kilograms = 0.453592 * pounds;
    return kilograms;
}
```

Returning Values from Functions

- When a function returns a value, the data type of this value must be specified
- In the declaration `float lbstokgs(float);`, the first `float` represents the return type
- When a function returns a value, the call to the function `lbstokgs(lbs)` is considered to take on the value returned by the function

The `return` Statement

- While many arguments may be sent to a function, only one argument may be returned from it
- Always include a function's return type in the function declaration. If it does not return anything, use the keyword `void` to indicate this

Eliminating Unnecessary Variables

```
// eliminates unnecessary variables
#include <iostream>
using namespace std;

float lbstokgs(float); // declaration

int main()
{
    float lbs;
    cout << "Enter your weight in lbs: "; cin >> lbs;
    cout << "Your weight in kgs is: " << lbstokgs(lbs) << endl;
    return 0;
}

float lbstokgs(float pounds)
{
    return 0.453592 * pounds;
}
```

Returning Structure Variables

```cpp
// demonstrates returning a structure
#include <iostream>
using namespace std;

struct Distance
{
    int feet;
    float inches;
};

Distance addengl(Distance, Distance);
void engldisp(Distance);

int main()
{
    Distance d1, d2, d3;
    cout << "\nEnter feet: "; cin >> d1.feet;
    cout << "Enter inches: "; cin >> d1.inches;
    cout << "\nEnter feet: "; cin >> d2.feet;
```

Returning Structure Variables

```
    cout << "Enter inches: "; cin >> d2.inches;
    d3 = addengl(d1, d2);
    cout << endl;

    cout << "Sum of ";
    engldisp(d1); cout << " and ";
    engldisp(d2); cout << " is: ";
    engldisp(d3); cout << endl;
    return 0;
}
```

Returning Structure Variables

```
Distance addengl(Distance dd1, Distance dd2)
{
    Distance dd3;
    dd3.inches = dd1.inches + dd2.inches; //add the inches
    dd3.feet = 0;
    if(dd3.inches >= 12.0)
    {
        dd3.inches -= 12.0;
        dd3.feet++;
    }
    dd3.feet += dd1.feet + dd2.feet;
    return dd3;
}
```

Returning Structure Variables

```cpp
void engldisp(Distance dd)
{
    cout << dd.feet << "\'-" << dd.inches << "\"";
}
```

Exercise 1

- Write a function that
  - takes two `Distance` values as arguments, and
  - returns the larger one.
- Include a `main()` program that
  - accepts two `Distance` values from the user,
  - compares them, and
  - displays the larger.

Exercise 2

- Write a function called `hms_to_secs()` that
  - takes three `int` values – for hours, minutes, and seconds – as arguments, and
  - returns the equivalent time in seconds (type `long`).
- Create a program that
  - exercises this function by repeatedly obtaining a time value in hours, minutes, and seconds from the user (format 12:59:59),
  - calling the function, and
  - displaying the value of seconds it returns.

Exercise 3

- Create a structure called `Time`. Its three members, all of type `int`, should be called `hours`, `minutes`, and `seconds`.
- Create two functions,
  - One of them, `time_to_secs()`, should take as its only argument a structure of type time, and return the equivalent in seconds (type `long`)
  - The other one, `secs_to_time()` should take as its only argument a time in seconds (type `long`), and return a structure of type `time`
- Write a program that
  - exercises these functions by obtaining two time values from the user in hh:mm:ss format, and
  - printing out the sum of them in hh:mm:ss format