Call by Reference Arguments

Lecture 11

Outline

1. Function Review
2. Call by Reference
3. Comparison with Call by Value
4. Gotchas

Function Review (1)

Write a function named **`get_integer`** that reads a single integer from the user and returns it

Answer

```
int get_integer()
{
   int input;
   cout << "Enter an integer: ";
   cin >> input;
   return input;
}
```

Function Review (2)

Write a function named **`maximum`** that takes two integer arguments and returns the larger of the two values

Answer

```
int maximum(int a, int b)
{
   if ( a >= b )
   {
      return a;
   }
   return b;
}
```

Function Review (3)

- Functions can only return a single value (or no value for `void` functions)
- Values are copied to the function arguments, and any changes made are not permanent after the function finishes
- This is known as *call by value*
  - The function call uses the value of the arguments

Function Review (4)

```
#include <iostream>
using namespace std;

int do_something(int a);

int main()
{
    int x, result;
    x = 10;
    cout << "before do_something, x=" << x << endl;
    result = do_something( x );
    cout << "after do_something, x=" << x << endl;
    cout << "but result=" << result << endl;
    return 0;
}

int do_something(int a)
{
    a--;
    cout << "in do_something, a=" << a << endl;
    return a;
}
```

Call by Reference

- Sometimes you want the argument to a function to be the actual variable in the function call

- In other words, if you change the variable in the function, you want the change to be permanent after the function finishes
  - Known as *call by reference*

- To set an argument to be call by reference, put an **&** after the type in the argument list
  - For example: `int do_something(int& x);`

Example

```
#include <iostream>
using namespace std;

int do_something(int& a);

int main()
{
    int x, result;
    x = 10;
    cout << "before do_something, x=" << x << endl;
    result = do_something( x );
    cout << "after do_something, x=" << x << endl;
    cout << "but result=" << result << endl;
    return 0;
}

int do_something(int& a)
{
    a--;
    cout << "in do_something, a=" << a << endl;
    return a;
}
```

Call by Reference vs. Call by Value

- Call by reference arguments are used when you need to <u>change</u> the value of a variable in the function
- Call by reference arguments can also be used to get around the limit of returning only a single value from a function
- If neither of these is needed, then the argument should be call by value (no **&**)

Exercise

Modify the **get_integer** function so that it uses a call by reference argument to store the input value

```
int get_integer()
{
   int input;
   cout << "Enter an integer: ";
   cin >> input;
   return input;
}
```

Answer

```
void get_integer(int& input)
{
   cout << "Enter an integer: ";
   cin >> input;
}
```

Mixed Arguments

- Each argument in a function can be either call by value or call by reference
- You have to decide for each argument if it should be call by reference or not
- If the variable needs to be changed by the function then make it call by reference
- Otherwise, leave it call by value

Example

```
#include <iostream>
#include <string>
using namespace std;

void get_double(string message, double& input);
const double PI = 3.14159;
void circle_calc(double radius, double& area, double& circumference);

int main()
{
     double r, a, c;
     get_double("Enter the radius: ", r);
     circle_calc(r, a, c);
     cout << "Area is " << a << " and circumference is " << c << endl;
     return 0;
}

void get_double(string message, double& input)
{
     cout << message;
     cin >> input;
}

void circle_calc(double radius, double& area, double& circumference)
{
     area = PI * radius * radius;
     circumference = PI * 2 * radius;
}
```

Gotchas

- Each call by reference argument in the function call must be a single variable, otherwise you will get build errors
  - Examples:

```
void get_integer(int& input)
…
get_integer( 10 );
get_integer( x*x );
```

- Always double check that the function declaration and definition match for each argument

Wrap Up

- Call by reference arguments can be used when you want to modify the value of a variable in a function
- They are also useful as a way to get around the one return value limit
- You should only use them when appropriate
- Always double check that the declaration and definition match
  - Call by reference requires that the argument is a single variable (not a constant or result of an expression)