Memberwise Assignment & Pointers

Chapters 7 and 10

Example

```
Time cTest1(9, 25, 32);
Time cTest2;


cTest2 = cTest1;


cTest2.printStandard();
```

Pointers

- Pointers are one of the most powerful features of C++
- Pointers give programmers more control over the computer’s memory
- A pointer is the memory address of a variable
- A pointer is one of the most difficult and important concepts in C/C++

Variable Addresses

- A variable’s address is the address of the first byte allocated to that variable
- Why the first byte?
- How can we find out the size of data types on a machine?

Pointer Declarations (10.2)

- The memory address of a variable can be stored in another variable called a pointer
- Pointers are declared using the `*` operator
- The following declares a pointer to an integer
  - `int *pLength;`
- In the following statement, **`length`** is an integer and **`pLength`** is a pointer to an integer
  - `int *pLength, length;`

Pointer Declarations (10.2)

- How would you create two pointers to doubles?
- Note:
    - Using our coding standards, we will use the convention that all pointer variables start with a small p (eg. pCount, pX)

Address Operator (10.1)

- How do we assign to a pointer the address of a variable?
- Use the address operator (**`&`**)
- **`&`** returns the memory address of it’s operand
- Example:
    - **`pLength = &length;`**
- Where have we used **`&`** before?

Address Operator

- Operand of the address operator must be an *lvalue*
- An *lvalue* is something to which a value can be assigned
- Address operator cannot be applied to constants
  - `int *pX;`
  - `int const NUM = 98;`
  - `pX = &NUM;                // ERROR`
  - `pX = &8;                  // ERROR`

Pointer Operations (10.2)

```
int x, *pX;
x = 8;    // set x to a value of 8
pX = &x; // set the pointer variable to point
         // to the address of x

cout << "x is: " << x << endl;
cout << "Size of x is: " << sizeof(x) << endl;
cout << "Address of x is: " << pX << endl;
cout << "Address of x is: " << &x << endl;
```

Indirection Operator

- How can we use the pointer variable to modify the value in the variable?
  - i.e. how to use **`pX`** to change the value of **`x`**
- *Answer:* use the indirection operator (**`*`**)
- The **`*`** operator dereferences the pointer
  - You are actually working with whatever the pointer is pointing to
- Using the example on the previous slide
  - **`cout << "pX is pointing to: " << *pX << endl;`**

Indirection Operator

- Using **`*`** as we did in the previous example is called dereferencing the pointer
- Using our example, how can we dereference **`pX`** so that it changes the value of **`x`** from 8 to 10?
- How can we change the value of **`x`** to a value entered by the user using the indirection operator?

Common Pointer Mistakes

- What is wrong with the following?

```
int x, *pX;
x = 8;


*pX = 2;
pX = 9;
*x = 4;
```

Pointers and Functions (10.7)

- What are the two ways of passing arguments into functions?
- Write two functions **`square1`** and **`square2`** that will calculate the square of an integer.
  - **`square1`** should accept the argument passed by value,
  - **`square2`** should accept the argument passed by reference.

Pointers and Functions (10.7)

- There is a third way of passing arguments into functions
- It’s called
  - passing by reference without using reference arguments
  - Or passing by reference using pointers
- The address of the argument is passed instead of the argument itself

Passing by reference (10.7)

```
void square3(int *pNum)
{
  *pNum *= *pNum;
}
```

- What would a function call to the above function look like?

Function Call (10.7)

```
int val = 5;

square3(&val);

cout << val << endl;
```