Pointers

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
- How are variables declared in C++?

Pointer Declarations

- 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 *pX;`
- In the following statement, **x** is an integer and **pX** is a pointer to an integer
  - `int *pX, x;`

Pointer Declarations

- How would you create two pointers to doubles?
- Notes:
  - When naming pointer variables, the book uses the convention of having ptr at the end of the name (eg. countPtr, xPtr)
  - Using our coding standards, we will use the convention that all pointer variables start with a small p (eg. pCount, pX)
- Coding standards are available at http://zeus.cs.pacificu.edu/shereen/CodingStandards.html

Address Operator

- 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:
  - `pX = &x;`
- 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 const NUM = 98;`
  - `pX = &NUM;  // ERROR`
  - `pX = &8;  // ERROR`

Pointer Operations

```
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 << "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 returns a synonym to whatever the pointer variable 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

- 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

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

Passing by reference

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

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

Function Call

```
intval = 5;
square3( &intval );
cout << intval << endl;
```


Summary

- Today I introduced
  - The concept of pointer variables
  - The address operator
  - The indirection operator
- We have covered:
  - P. 320 - 329