Chapter 11

Introduction to Pointers part 3

Agenda

- The this pointer
- Pointers and functions
  - Returning pointers
  - Pointer parameters

We've seen this before

```
class Person
{
private:
     string _name;
public:
     // Constructor
     Person (const string &name)
     {
     _name = name;
     }
};
```

this is a special member variable of every object. It is a pointer to the object itself

```
class Person
{
private:
     string name;
public:
     // Constructor
     Person (const string &name)
     {
     this->name = name;
     }
};
```

This is explicitly referring to the Name property of the Person object.

Pointers & Functions

You can return a pointer from a function (*remember a pointer is just a data type*!)

```
int * alloc_array( int size )
{
    // Allocate array
    int* pintarray = new int[size];

    // Initialize array
    for (int i = 0; i < size; i++)
        pintarray[i] = 42;

    // return array pointer
    return pintarray;
}
```

Watch for homework!

You can call your function to create and initialize your array

```
int main( )
{
    int* array = alloc_array(10);

    for (int i = 0; i < 10; ++i)
        cout << array[i] << endl;

    delete[] array; //don't forget!
}
```

Be Careful! You should NEVER return a pointer to a local variable!

```
int* bad_alloc_int_array()
{
    int mylocalarray[100];
    int * pintarray = mylocalarray;
    return pintarray;
}
```

WHY IS THIS BAD?

Passing pointers to functions

- Again, pointers are just a data type
  - And they are a lot like arrays
- We can pass pointers to functions
  - And just as we did for arrays, we'll want to pass the size (if it points to an array of values)

Full example

- Write a program that
  1. Asks the user how many numbers they want to enter
  2. Reads those numbers into a dynamically-allocated array
  3. Counts the number of even numbers in the array

- You can get this from https://github.com/ptucker/PointersAndFunctions.git

Main

```
int main()
{
    int nums;

    prompt_count(&nums);
    // we've seen this one
    int* numbers = alloc_array(nums);
    populate_numbers(numbers, nums);
    cout << "you gave me " << countEvens(numbers, nums)
         << " evens.\n";

    delete[] numbers;
}
```

functions

```
void prompt_count(int* size) {
    cout << "how many numbers will you enter? ";
    cin >> *size;
}

void populate_numbers(int* numbers, int size) {
    for (int* curr = numbers; curr < numbers + size; curr++) {
        cout << "Enter number: ";
        cin >> *curr;
    }
}

int countEvens(int* numbers, int size) {
    int evens = 0;
    for (int*curr = numbers; curr < numbers + size; curr++) {
        if (*curr % 2 == 0)
            evens++;
    }

    return evens;
}
```

In-class Exercise

- Try out the previous code
  - Make sure you understand how it works
- Convert the countEvens function to use array notation (e.g. `numbers[i]`) rather than the pointer notation
- Add a function that counts the positive numbers in the array

Recall ways we pass arguments to functions in C++

- **PASS-BY-VALUE** parameters create a copy of the argument
- **PASS-BY-REFERENCE** parameters actually reference the argument memory

---

- **POINTER** parameters pass a pointer to the argument

Review: Pass-by-**Value** Example

```
void Swap(int First, int Second)
{
        int Temp = First;
        First = Second;
        Second = Temp;
}

#include <iostream.h>
void main()
{
        int x = 6, y = 7;
        Swap( x, y );
        cout << "X=" << x << " Y="<<y;
}
```

	Memory
x	6
y	7

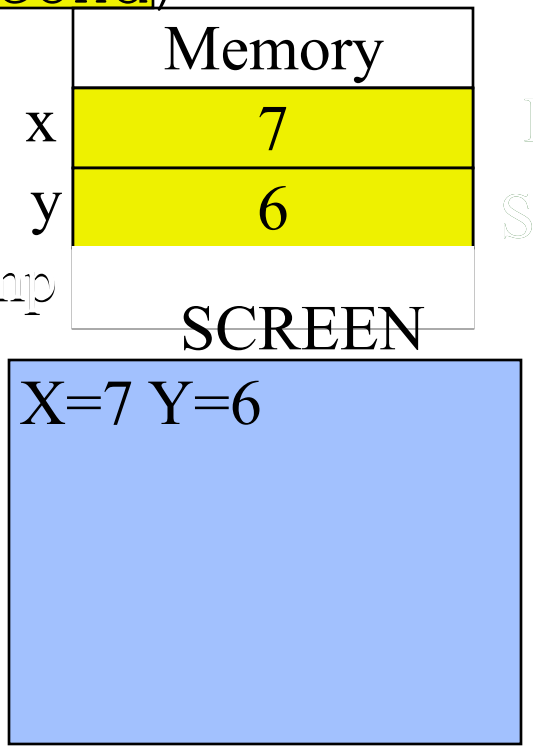
SCREEN

X=6 Y=7

Review: Pass-by-**Reference** Example

```
void Swap(int& First, int& Second)
{
        int Temp = First;
        First = Second;
        Second = Temp;
}

#include <iostream.h>
void main()
{
        int x = 6, y = 7;
        Swap( x, y );
        cout << "X=" << x << " Y=" << y;
}
```



Pass-by-Pointer Example

```
void Swap(int *pFirst, int *pSecond)
{
    int Temp = *pFirst;
    *pFirst = *pSecond;
    *pSecond = Temp;
}
```

	Memory	
x	7	10
y	6	14

```
#include <iostream.h>
void main()
{
    int x = 6, y = 7;
    Swap( &x, &y );
    cout << "X=" << x << " Y="<<y;
}
```

SCREEN

X=7 Y=6

Pointer Parameters: Three Rules

1) Add asterisk * with the parameter in the prototype and header

```
void prompt_count( int *size );
```

2) Use asterisk * in the function body to dereference the pointer

```
cin >> *size;
```

3) Supply address as argument in the function call

```
prompt_count( &size );
```