LECTURE 33: THE BIG THREE

Today

- Reading
  - Weiss Chapter 4.6
- Objectives
  - This week's assignment
  - Call-by-value vs. call-by-reference
  - The Big Three

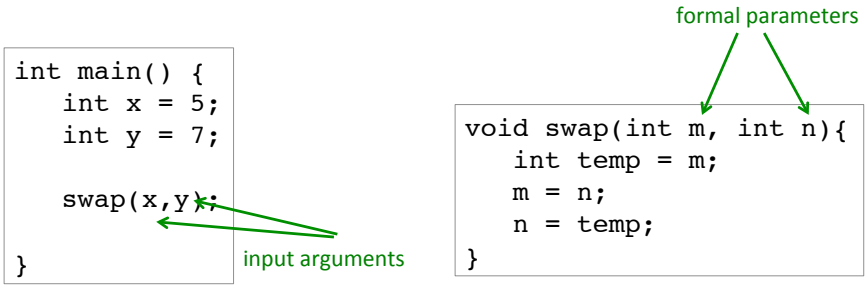
This Week's Assignment

- Create a Twenty-Questions Animal Game
- Split into two parts
  - Part 1 is due *Tuesday* April 21st
  - Part 2 is due *Tuesday* 28th
- Part 1 is writing the I/O functions
- Part 2 is implementing a BinaryTree class and implementing the main function that controls the game

Call-by-value

- Java and C++ use call-by-value when passing parameters
- Call-by-value: the input arguments are copied into the formal parameters

Call-by-value



x and y aren't swapped and lots of time spent copying actual objects

Call-by-value

- Change the formal parameters to now be pointers

```
int main() {
   int x = 5;
   int y = 7;

   swap(&x,&y);

}
```

```
void swap(int *m, int *n){
   int temp = *m;
   *m = *n;
   *n = temp;
}
```

x and y are now swapped but requires changing syntax!

References in C++

A reference is a constant pointer that is automatically dereferenced

- See ptr_exs.cpp
- To declare a reference, use & symbol

```
int x = 5;
int &int_ref = x;  // cannot change!
```

Call-by-reference

- C++ also allows call-by-reference

```
int main() {
   int x = 5;
   int y = 7;

   swap(x,y);

}
```

```
void swap(int &m, int &n){
   int temp = m;
   m = n;
   n = temp;
}
```

The best solution!

Binary Search Example

```
int binarySearch(int val,vector<int> arr,int lo, int hi) {
      if( lo > hi) { return -1; }
      int mid = (lo+hi)/2;
      if(arr[mid] == val) {
            return mid;
      }
      else if(val < arr[mid]) {
            return binarySearch(val,arr,lo,mid-1);
      }
      else {
            return binarySearch(val, arr, mid, hi);

      }
}
```

This is inefficient. Why?

Binary Search Example

new function prototype

```
int binarySearch(int val, const vector<int>& arr,int lo, int hi);
```

- The `&` operator means no copying of input arguments
- `const` means this function will not change (mutate) this input parameter
- Only const methods can be called on `arr`

References in C++

- Benefits
  - Get the low-memory overhead of using a pointer
  - Without the need to use the dereference operator

The Big Three

- Destructor, copy constructor, `operator=`
- Default implementations of these methods are provided
- *Rule-of-thumb: If you need to overwrite one of these, overwrite them all*

Destructor

- Called when the object goes out of scope or when delete is called on an object
- Releases all resources
  - memory, files, streams

Copy Constructor

- Constructs a new object from an existing object

```
IntCell copy = existing;
IntCell copy(existing);
```

- Behind the scenes
  - an input parameter to a call-by-value function
  - an object returned by value
- Copy constructor is not called

```
IntCell copy;
copy = existing;
```

operator=

- Assignment for two *already constructed* objects
- Example usage,

```
IntCell first(3);
IntCell scnd;
scnd = first;
```

The Big Three for IntCell class

```
class IntCell {
public:
	// destructor
	~IntCell() {
		// does nothing
	}

	// copy constructor
	IntCell(const IntCell & rhs) : value(rhs.value) {
		//does nothing
	}

	// operator= on next slide
private:
	int value;
};
```

The Big Three for IntCell class

```
// assignment operator
IntCell & operator=(const IntCell& rhs) {
   if(this != &rhs){ // alias test
      value = rhs.value;
   }
   return *this;   // *this is the actual object
}
```

The Big Three for IntCell class

- The `this` keyword has the same functionality as in Java
  - `this` is a *pointer* to the current object
- The alias test is extremely important
  - disallows x = x
- Function returns a reference to object
  - allows for a = b = c

The Big Three with Pointers

- Class contains a pointer to heap-allocated memory
  - Default destructor does not call delete
  - Default copy constructor and operation= functions only copy the value of the pointer (not the data pointed to)
    - Called a shallow-copy
  - A deep-copy is when an entirely separate copy is made
    - Often times, this is what we really want!

The Big Three for IntCell class

- Change the IntCell class to hold a pointer to an integer
- See `intcell_default.h` and `intcell_bigthree.h`
  - One uses the default big three
  - The other (`intcell_bigthree.h`) correctly overwrites the big three
- Miscellaneous final comments
  - To disallow default copy constructor and operator=, move to private section
  - The `this` keyword has type `Class* const` (can't modify)

Access Modifiers in C++

- In Java: public, protected, private, and “default”
- In C++: public, private, and friend
- Friends are allowed access to all private members (variables and functions) of the class