CS 368 Announcements
Wednesday, October 9, 2013

Record your attendance on the sign-in sheet.

Homework hw – due Wednesday, October 9th by 10:00 pm

Program p1 graded

Program p2 – due Wednesday, October 23rd by 10:00 pm

Last Time
- **`typedef`**
- C++ memory model
- pointers and parameter passing
- pointers and return values

Today
- wrap up Ch. 3 from last lecture
- start Ch. 4
- `.h` and `.cpp` files
- intro to defining classes
- multi-file compilation

Next Time
- continue Ch. 4
- makefiles
- Unix utilities: gdb and valgrind
- constructor
- member initialization
- destructor
- copy constructor
- copy assignment

Handling Multiple Files

Separating interface from implementation:

- Java:

- C++:

Header file contains:

Source file contains:

IntList.h

```
#ifndef INTLIST_H
#define INTLIST_H

class IntList {

  public:
    // Constructors
    IntList();
    IntList(int size);

    IntList(const IntList &L);              // copy constructor
    ~IntList();                             // destructor
    IntList & operator=(const IntList &L);  // assignment

    // Other public methods
    void addToEnd(int k); // adds value k to end of list
    void print() const;   // prints out list (using cout)


  private:
    static const int SIZE = 10;  // default init array size
    int *items;      // dynamically allocated array of ints
    int numItems;    // number of items currently in list
    int arraySize;   // current size of the array
};

#endif
```

IntList.cpp

```cpp
#include <iostream>
#include "IntList.h"

using namespace std;

IntList::IntList()...
IntList::IntList(int size)...
IntList::IntList(const IntList &L)...
IntList::~IntList()...
IntList::IntList & operator=(const IntList &L)...

// addToEnd - adds item k to the end of the list
// If the array is full, array is first doubled in size.
void IntList::addToEnd(int k) {

    // check if enough room, if not, double the array
    if (numItems == arraySize) {
        int *newItems = new int[arraySize*2];
        for (int i = 0; i < numItems; i++)
            newItems[i] = items[i];
        delete [] items;
        items = newItems;
        arraySize *= 2;
    }

    // add the item
    items[numItems] = k;
    numItems++;
}

// print - prints out the list to the console (using cout)
void IntList::print() const {
    for (int i = 0; i < numItems; i++)
        cout << items[i] << " ";
    cout << endl;
}
```

Handling Multiple Files (cont.)

preprocessor command = any line beginning with #

```
#define

#include
```

conditional compilation

- What if, with all your nested `#include`s, you end up having a definition show up more than once?

testIntList.cpp

```cpp
#include <iostream>
#include "IntList.h"

using namespace std;

void copyAgain(IntList L) {
    IntList Lnew = L;
    Lnew.print();
    L.print();
}

int main() {
    IntList L1;
    IntList L2(25);
    IntList *p, *q;
    p = new IntList;
    q = new IntList(25);
    for (int i = 0; i < 20; i++)
        L1.addToEnd(i+1);
    L1.print();
    L2 = L1;
    L2.addToEnd(-1);
    IntList L3(L1);
    L1.print();
    L2.print();
    L3.print();
    copyAgain(L3);
//    delete p;
//    delete q;
    return 0;
}
```

Dealing with multiple files in your project

To compile all at once (only useful if you have just a couple files):

```
g++ ExampleObj.cpp testExampleObj.cpp -o runIt
```

OR

Can compile separately, creating object files (ending in `.o`):

```
g++ -c ExampleObj.cpp
g++ -c testExampleObj.cpp
```

and then link:

```
g++ ExampleObj.o testExampleObj.o -o runIt
```