CS 368 Announcements
Wednesday, October 16, 2013

Record your attendance on the sign-in sheet.

Homework hw – graded

Program p2

- due Wednesday, October 23rd by 10:00 pm
- note: will add in p3 destructor, copy constructor, operator=

Last Time

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

Today

- wrap up intro to defining classes
- compiling multiple files
- makefiles

Next Time

- cont. Ch. 4
- constructor & member initialization
- `const` member functions
- copy constructor
- copy assignment (operator=)
- destructor

Recall 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 member functions
    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

```
#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

```
#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;
}
```

Compiling Multiple Files

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
```

Makefile

```
main: testIntList.o IntList.o
     g++ testIntList.o IntList.o


testIntList.o: testIntList.cpp IntList.h
     g++ -c testIntList.cpp


IntList.o: IntList.cpp IntList.h
     g++ -c IntList.cpp


clean:
     rm *.o
```

Using the Makefile

```
king06(1)% ls
IntList.cpp  IntList.h   Makefile  testIntList.cpp

king06(2)% make
g++ -c testIntList.cpp
g++ -c IntList.cpp
g++ testIntList.o IntList.o

king06(3)% ls
a.out       IntList.h    Makefile  testIntList.o
IntList.cpp  IntList.o   testIntList.cpp

king06(4)% rm IntList.o
rm: remove regular file `IntList.o'? y

king06(5)% make
g++ -c IntList.cpp
g++ testIntList.o IntList.o

king06(6)% make clean
rm *.o

king06(7)% ls
a.out  IntList.cpp  IntList.h  Makefile  testIntList.cpp

king06(8)% make
g++ -c testIntList.cpp
g++ -c IntList.cpp
g++ testIntList.o IntList.o

king06(9)% make IntList.o
make: `IntList.o' is up to date.
```