CS 368 Announcements
Wednesday, March 6, 2013

Program p1 graded

Homework hw
- due Friday, March 8 by 10 pm
- can use late days

Program p2
- due Monday, March 18
- 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 from last lecture
  - `const` member functions
- multi-file compilation - makefiles
- constructor
- member initialization
- start copy constructor

Next Time
- finish Ch. 4
- copy constructor
- copy assignment (operator=)
- destructor
- start Ch. 5 operator overloading

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
```

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
```

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.
```

Constructors

A constructor is called:

- when an object is declared:

- when an object is dynamically allocated:

In Java: data members are initialized automatically before a constructor executed

In C++: before body of constructor is executed, each data member (field) that is an object is initialized by calling the field's no-arg constructor

Member Initialization Lists

In IntList.cpp

```
IntList::IntList() :
    items(new int[SIZE]), numItems(0), arraySize(SIZE)
{
}


IntList::IntList(int size) :
    items(new int[size]), numItems(0), arraySize(size)
{
}
```

Copy Constructor

Called automatically when an object is:

- passed by value to a function

- returned (by value) as a function result

- declared with an initialization from an existing object of the same class

Copy Constructor (continued)

In IntList.cpp

```
IntList::IntList(const IntList &L) :
    items(new int[L.arraySize]), numItems(L.numItems),
    arraySize(L.arraySize)
{
    for (int k = 0; k < numItems; k++)
        items[k] = L.items[k];
}
```