Monday, February 24, 2003
MAT 102 - Data Structures

Function Overloading in C++
Example • Task: Find the maximum integer among two or three integers.

- We will need two different functions: one to compare two integers and one to compare three integers.
- In C++, we can use the same name for two functions or type of arguments, as long as the function arguments differ in some way (number or type of arguments).

Example
```c
int max(int a, int b) {
    if (a >= b)    return a;
    else    return b;
}

int max(int a, int b, int c) {
    if ((a >= b) && (a >= c))     return a;
    else    return max(b, c);
}

int main() {
    // use max functions here
}
```
Overloading Constructors • Now use the idea of overloading constructors in a C++ class.

You should always write a default constructor that takes no arguments.

You may want to write other constructors that take arguments.
class List {
public:
  List();
  List(int initSize, ListItemType initValue);
  List(const List &anotherList);
  ~List();
  // other member functions

private:
  int size;
  ListNode *head;
  // private member functions

private:
  // other member functions

friend List ~List();

friend List (const List &anotherList);

friend List (int initSize, ListItemType initValue);

friend List ();

public:
};

class List {

private:
  // constructors for the List class

};
constructor that creates a list of size initSize and sets all items to initValue

```cpp
for (int i = 1; i <= initSize; i++)
    head = NULL;
size = 0;
```

`LIST::LIST(1nt initSize, 1nt initValue)`
int main(){  List defaultList;  List secondList(5, 75);  defaultList.print();  secondList.print();}

The list is empty, nothing to print.

List size is 5

List item 1 is 75
List item 2 is 75
List item 3 is 75
List item 4 is 75
List item 5 is 75
In the Poly class, for example, a copy constructor is called (three times) when you type:

\( h = \text{addTwoPolys}(f, g) \)

• You define and initialize an object
• You return an object from a function
• You pass an object by value to a function

A class's copy constructor is called when

The Copy Constructor
Declaring and Initializing

The copy constructor is also called for "g" in the following:

```cpp
Poly g = f;   // calls copy constructor
```

in the following:

```cpp
f.input();
```

```cpp
Poly f;
```

The copy constructor is also called for "f".
A shallow copy is not enough for classes that have been (or will be) dynamically allocated.

A shallow copy is not enough for classes that have been (or will be) dynamically allocated.

This is called a "shallow copy". A shallow copy is fine for basic data types and for arrays.

This is called a "shallow copy". A shallow copy is fine for basic data types and for arrays.

The compiler generates a copy constructor for you, but it only copies the member variables of the class.

The compiler generates a copy constructor for you, but it only copies the member variables of the class.

Compiler Generated Copy
Deep Copy

Separate copy of the object.
A deep copy copies more than just the member variables; it makes a complete, separate copy of the object.

Deep Copy
{ 

newPtr->next = NULL; 

{ 

newPtr->item = origPtr->item; 
newPtr = newPtr->next; 
assert (newPtr->next == NULL); 
newPtr->next = new ListNode; 
} 

origPtr = origPtr->next; 
origPtr = NULL; 
for (ListNode* origPtr = anotherList.head->next; 
ListNode* newPtr = head; 
copy the rest of the list //
head->item = anotherList.head->item; 
assert (head->next == NULL); 
head = new ListNode; 
} 
else 
head = NULL; 
if (anotherList.head == NULL) 
(size = anotherList.size; 
} 
this List (const List &anotherList) 
copy constructor, ensures a deep copy //
Schedule for the next few classes

- Fri, Mar 7, Exam #1, Pass out Project #3
- Review for Exam
- Wed, Mar 5, meet in classroom, Project #2 Due
- Mon, Mar 3, meet in lab, work on Project #2
- C++ strings and I/O using files
- Fri, Feb 28, meet in classroom, lecture on
- Wed, Feb 26, meet in lab, work on Project #2
- Fri, Mar 7, Exam #1, Pass out Project #3
- Review for Exam
- Wed, Mar 5, meet in classroom, Project #2 Due
- Mon, Mar 3, meet in lab, work on Project #2
- C++ strings and I/O using files
- Fri, Feb 28, meet in classroom, lecture on
- Wed, Feb 26, meet in lab, work on Project #2