CS 162
Intro to Computer Science II

Lecture 14
Constructors (cont.)
Shallow vs. Deep copy
Midterm Review
2/14/23
Today’s Topics:

- Constructors (cont.)
- Destructor
- Has-a relationship
- Midterm review

- Shallow vs. Deep copy
- Begin Big three
Introducing Constructor

• Constructor – a specially defined function
• Automatically called when the object is created
• Sets up (initializes) the object with appropriate values
  • Member variable values
  • Allocating memory for member variables
  • *Opening a file to read from or write to
• If a constructor is not provided by the programmer, one will be automatically generated (implicitly) but will not initialize any values
More details on Constructors

- **Must** have the same name as the class
- Not allowed to return anything
- May have parameters
  - If no parameters provided, referred to as **default constructor**
  - If parameters are provided, referred to as **non-default constructor** (a.k.a. **parameterized constructor**).
  - It can be defined in a couple ways:
    - Option 1: Use assignment statements
      ```
      Point::Point (){  
          this->x = -1;  
          this->y = -1;  
      }
      ```
    - Option 2: Use initialization list
      ```
      Point::Point (int a, int b){  
          this->x = a;  
          this->y = b;  
      }
      ```
  - If using const member variable, it has to be initialized in constructor(s) using initialization list
    - E.g. ```
        Point::Point():z(5){}  //where z is defined as a const int
      ```
More details on Constructors

• Each class may have **at most one** default constructor, and **any number** of non-default ones

• If you define any non-default constructors for a class, a default one is **likely needed**

• If constructors are explicitly defined for a class, the compiler will not generate one for you
  - Typical compile time error: a class has non-default constructors, but no default one. Create objects using default constructor → NoNo!!!

• Can’t be called using the dot operator

• Can be called after the object is created

```
next_point = Point (3,3);
```
Destructor

• Special function which is called automatically when the object is destroyed
  • Happens when a statically allocated object goes out of scope or when a dynamically allocated object is freed with delete

• Think of this as the “opposite” of the constructor

• Generally used to clean up dynamic memory usage, file I/O handles, database connections, etc.

• To create a destructor, declare a public class function with no return type, with the same name as the class, preceded by a tilde (~):
  • E.g. ~Point();

• Demo...
Passing Objects

• Can be passed the same way as any other variable
• Traditionally pass by reference
  • Generally more efficient
  • Pass by value makes two copies → requires the copy constructor at least once
  • Pass by reference only uses the one variable, no copies
  • Can be problematic since changes to references persist
Class Composition

• Class Composition – a fundamental concept in OOP
  • Describes a class that “has” one or more objects of other classes.
• Allows to model a “has-a” relationship between objects.

• i.e. In assignment 3, Manager “has a” Airport, and an Airport “has a” Flight
Today’s Topics:

• Constructors (cont.)
• Destructor
• Has-a relationship
• Midterm review
• Shallow vs. Deep copy
• Begin Big three
Exam details

• When: Fri 2/16 12:00 – 12:50 pm
• Where: LINC 200
• Question Types: T/F, multiple choices

• Close everything, No calculators, scratch paper will be provided upon requests
• Bring pen/pencils, eraser, and your photo ID
Topics covered:

• Functions
• Pointers
• Stack vs. Heap
• 1D/2D static vs. dynamic array
• Structs
• File separation
• Compilation and Makefile
• File I/O
• Object Oriented Programming
• structs vs. classes

• Classes
• Accessors, Mutators
• Default vs. non-default constructors
• This keyword
• Const vs. static
• Access specifiers
Study Guide

• Take practice midterm and time yourself
• Review lecture slides
• Review worksheet 1-6
• Review lab 1-6
• Review assignment 1-3
Today’s Topics:

• Constructors (cont.)
• Destructor
• Has-a relationship
• Midterm review

• Shallow vs. Deep copy
• Begin Big three
Shallow Copy vs. Deep Copy

• Shallow:
  • A.k.a.: member-wise copy
  • Copy the contents of member variables from one object to another
  • Default behavior when objects are copied or assigned

```
Objetct1
  var1
  var2
  var3

Objetct2
  var1
  var2
  var3
```
Shallow Copy vs. Deep Copy

• Shallow:
  • What if the object has dynamic memory allocated?

  - This could be problematic as if we make any changes to the array in object 1, object 2 will be affected as well...
Shallow Copy vs. Deep Copy

• Deep:
  • Copy what each member variable is pointing to so that you get a separate but identical copy
  • Has to be programmer-specified
Assignment Operator (=) Overload

• Predefined assignment operator returns a reference
  • Allows us to chain assignments together: \( a = b = c \)
    • First set \( b = c \) and return a reference to \( b \). Then set \( a = b \)
    • Need to make sure the assignment operator returns something of the same type as its left hand side

• Overloading assignment operator
  • Must be a member of the class
Copy Constructor

• Constructor that has one parameter that is of the same type as the class
  • Has to accept reference as parameter (normally $\texttt{const}$)
  • Allows for distinct copies, changes to one does not impact the other
  • Called automatically in three cases:
    • When a class object is being declared and initialized by another object of same type
    • Whenever an argument of the class type is “plugged in” for a call by value parameter
    • When a function returns a value of the class type
Destructor

• Delete the object
• Will be automatically created if one is not supplied
  • Will not handle dynamic memory
  • ~Class_name(); //no return type, no parameters, only one allowed
• Called when the object goes out of scope
  • When the function ends
  • When the program ends
  • A block containing local variables ends
  • A delete operator is called
The Big Three

• If you implement either a **Destructor**, a **Copy Constructor**, or an **Overloaded Assignment Operator**, you should ensure that all 3 are defined

• If you needed one, you probably need all of them

• This rule of thumb goes by several names:
  • The Big Three
  • The Rule of Three
  • The Law of The Big Three

• *C++11 has an expanded version: The Big 5
  • We won’t cover this yet
### Big Three Activity

<table>
<thead>
<tr>
<th>Function</th>
<th>Prototype</th>
<th>Job</th>
<th>When is it called</th>
<th>Default Behavior if not defined?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constructor</td>
<td>ClassName();</td>
<td>Build the object</td>
<td>Default is called when object is declared with no parameters and no “=“ sign. Nondefault is called if parameters are given</td>
<td>The compiler will provide a default one. It will initialize all variables with garbage values, will not set up pointers</td>
</tr>
<tr>
<td></td>
<td>ClassName(w/ params)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Copy Constructor</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Assignment Operator Overload</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Destructor</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
To-do’s before Monday’s lecture:

• Assignment 3, you should be able to implement
  • Read from the text file and populate your array and objects
  • View all info
  • Check flight info
  • Add a new flight
  • Print stats