Chapter 9
Objects and Classes

FORDHAM

CISC 5004 - Programming C++
Sam Kamens
Fall, 2022

Objects: Introduction

- Physical world
  - Group lower level objects (fabric, buttons, thread) into higher level objects (clothes)
- Programs
  - Group lower level items into higher level groupings (objects)
- **Object**
  - A grouping of data (variables) and operations that can be performed on that dats (functions)
  - Program objects example

Abstraction/Information Hiding

- **Abstraction**
  - User interacts with an item at a high level, with lower level details hidden
    - aka **Information Hiding** or **Encapsulation**
    - Example: Oven
- **Abstract Data Type** (ADT)
  - A data type whose creation and update and constrained to specific well-defined operations
  - A **class** can be used to implement an ADT

Public Member Functions

- Public Member Functions
  - Indicate all operations a class user can perform on the object
  - User doesn't know (or need to know) how the class is implemented
  - Only need to understand how the public functions behave
  - Example: Restaurant Class

Using a Class

- Programmer creates an object of the class by declaring a variable
  - `Restaurant favLunchPlace;`
- “.” operator (**member access operator**) is used to invoke a function on an object
  - `favLunchPlace.setRating(4);`
  - Note: we have seen this operator many times before
  - Using the restaurant class

Class example: string

- C++ string type is a class
  - Some of the public methods
  - What other public methods should there be?

Defining a class

- Private data members
  - Variables that can be accessed by functions within the class (member functions)
  - Class users can not access them
  - Appear in a `private:` section of the class definition
  - Example: Restaurant class private data members
- Public member functions
  - **Declare** each function in the `public:` section
  - **Define** each function (often in a different file or part of a file)
  - Example
- Scope Resolution Operator (`::`)
  - We have seen this (std::string)

Examples

- Complete Restaurant class
- RunnerInfo Class

Inline Member Functions

- Include function definition inside class definition
  - Example
  - Why?
    - Easier to read
    - Allows the compiler to optimize
  - OK to reference private variables before they are defined
  - Style: may use one-line definitions - example

Mutators and accessors

- Mutator
  - May modify a class's data members
  - aka "setter"
- Accessors
  - Accesses data members but does modify them
  - aka "getter"
  - Usually defined as **const**
    - Causes a compiler error if the function attempts to modify a data member
- Restaurant Mutators and Accessors

Private Helper Functions

- Non-public function used by public functions to carry out tasks
  - Example

Initialization

- Always initialize variables when declared
- Initialize class data members in the class definition (example)

Constructors

- Constructor
  - Special class member function called automatically when a variable of that class is declared
  - Can initialize data members
    - Potentially perform other work as well
    - Be careful: Error handling with constructors is complicated
  - May or may not have arguments
    - Without arguments ⇒ **default constructor**
  - Restaurant Example

Vector of Objects

- Can use a vector of objects
- Class can have vectors for private data
  - Reviews Class
- Can use classes within classes
  - Restaurant class with Reviews

File structure for classes

- Usually two files per class
  - ClassName.h
    - Class definition, including data members and function declarations (including inline definitions)
  - ClassName.cpp
    - Definitions of other member functions
  - Restaurant Example
  - Sometimes multiple related classes are grouped into fewer files, but the above is a good general practice

Class Design

- How do you choose what classes to create?
- Programmer thinks about what “things” or “concepts” to represent
  - Example
- Class ideas are converted to code
  - Careful not to include unnecessary header (.h) files
  - Only include the ones your code uses directly

Unit Testing for Classes

- Testbench should thoroughly test all public member functions
- Features of a good testbench
  - Automatic checks, failures are printed
  - Independent test cases (assume tests could run in any order)
  - Code coverage (ideally 100%)
  - Test edge cases and invalid inputs

Regression Testing

- Retest a class every time it changes
  - If previously passing tests fail, then the class as regressed
- Update testbench along with the class itself
- Testing is complicated
  - Some people are Test Engineers (or QA Engineers) who test for a living
- Sometimes the tests themselves have bugs
  - When tests report a failure, check the test too
  - Buggy test example

Constructor Overloading

- Class can offer multiple constructors (**overload**) with different parameter types
  - The constructor matching the arguments will be called.
  - Example
- If a constructor is explicitly defined, the compiler will not create a default constructor
  - Be sure to define one yourself, unless you explicitly do not want to support it
- Constructor parameters can be optional/default
  - If all parameters are optional, this can serve as default constructor as well
  - Example

Constructor Initializer Lists

- Alternate approach for initializing data members in a constructor
  - Example
  - Can make initialization more efficient (as here)
  - Also helpful with derived classes (more later)

The ‘this’ parameter

- Object’s method is called using `object.Function(...)`
- Compiler convert this to `Function(object, . . .)`
- Within a member function, the `object` can be referred to as `this`
  - It is a pointer, so it is accessed using `this->`
  - Allows for disambiguating variable names (see example)
  - Very useful in copy methods

Operator Overloading

- Can define (or redefine) built-in operators like `'+'`, `'*'`, `'[]'`
  - Example: Add time objects
  - Implementation
- Can overload multiple times with different operands
  - Example

Overloading Comparison Operators

- Create an `operator==` function
  - Programmer decides what makes two objects equal
  - Example
- Overloading the < operator
- Implement other relational operators in terms of `operator==` and `operator<`

```
bool operator>(const Review& lhs, const Review& rhs){ return rhs < lhs;}
bool operator<=(const Review& lhs, const Review& rhs){ return !(lhs > rhs);}
bool operator>=(const Review& lhs, const Review& rhs){ return !(lhs < rhs);}
```

Sorting a vector of objects

- `sort()`
  - Part of C++ Standard Template Library
  - Can sort vectors of arbitrary objects if the object has a less-than operator (`operator<`)
- To use `sort()`
  - Add `#include <algorithm>`
  - Overload the < operator
  - Call `sort(vector.begin(), vector.end());`
  - Example

Vector Abstract Data Type (ADT)

- Standard Template Library (STL) defines classes for common Abstract Data Types (ADTs)
- A vector is an ADT that is:
  - Ordered
  - Accessible by index
- Vector ADT functions
  - push_back() example
  - insert() and erase() example
  - insert in sorted order

Namespaces

- Defines a region (or scope) to avoid name conflicts
  - e.g. two definitions of `Seat` (auditorium, airplanes)
  - Resolving the conflict
- We have used this already with the `std` namespace
  - Namespace for standard library classes
  - Use scope resolution operator (`std::cout`), or
  - Add `using namespace` directive
    - Use this carefully

Static data members and functions

- `static` indicates a variable that is allocated only once for the whole program
  - Its value is maintained throughout
- In a class, a **static data member** is a member of the class, rather than of each class object
  - Independent of any one class object
  - Can be accessed without a class object at all
  - Declared inside the class definition, but must be defined outside
  - Example: Using static data member to create object ID's

Static member functions

- Class function independent of class objects
- Used to access and mutate private static data members
- No `this` pointer, so can only access static data members
- Example

Examples

- Salary calculation with classes
- Domain Name Availability
- Winning Team
- Artwork label