CISC181 Introduction to Computer Science

Dr. McCoy

Lecture 23
November 19, 2009

Another Look at Classes - const

- We have declared constants before – they are objects of a type that cannot be modified.
- Recall – the compiler complains if you try to modify an object that is declared to be a constant.
- With classes, must assure the compiler that a constant is not being modified – so need to declare some member functions to take const objects

7.2 `const` (Constant) Objects and `const` Member Functions

- Principle of least privilege
  - Only allow modification of necessary objects
- Keyword `const`
  - Specify object not modifiable
  - Compiler error if attempt to modify `const` object
  - Example

```
const Time noon( 12, 0, 0 );
```

    - Declares `const` object `noon` of class `Time`
    - Initializes to 12

7.2 `const` (Constant) Objects and `const` Member Functions

- `const` member functions
  - Member functions for `const` objects must also be `const`
    - Cannot modify object
  - Specify `const` in both prototype and definition
    - Prototype
      - After parameter list
    - Definition
      - Before beginning left brace

7.2 `const` (Constant) Objects and `const` Member Functions

- Constructors and destructors
  - Cannot be `const`
  - Must be able to modify objects
    - Constructor
      - Initializes objects
    - Destructor
      - Performs termination housekeeping

Example D&D Exercise 7.7

Create a date class with the following capabilities:

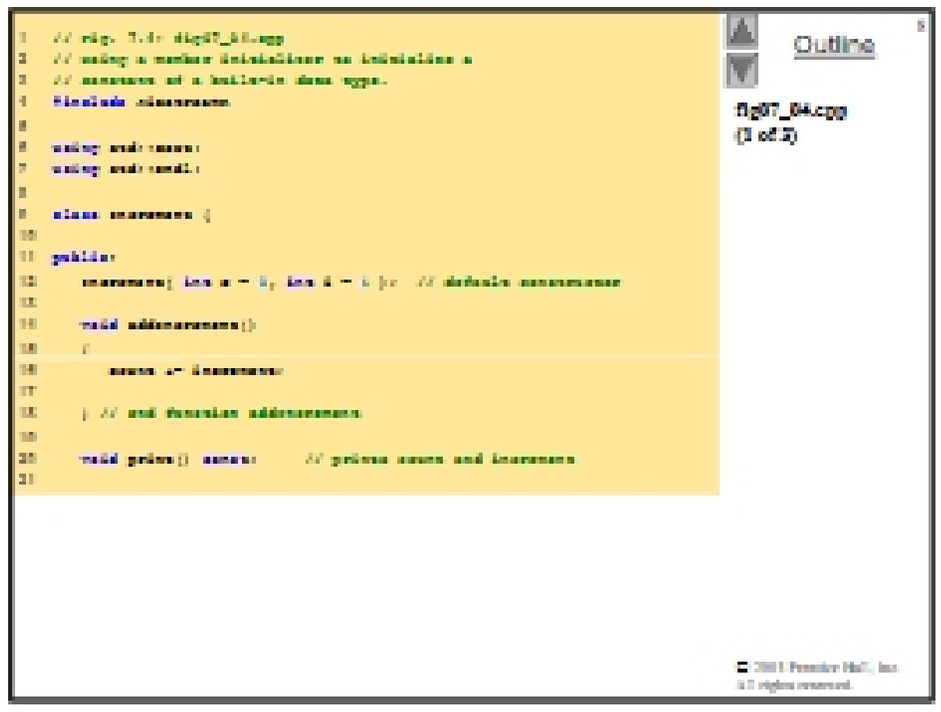
1. Output the date in multiple formats such as: ddd yyyy; mm/dd/yy, June 14, 1992
2. Use overloaded constructors to create Date objects initialized with dates of the formats in part (1).

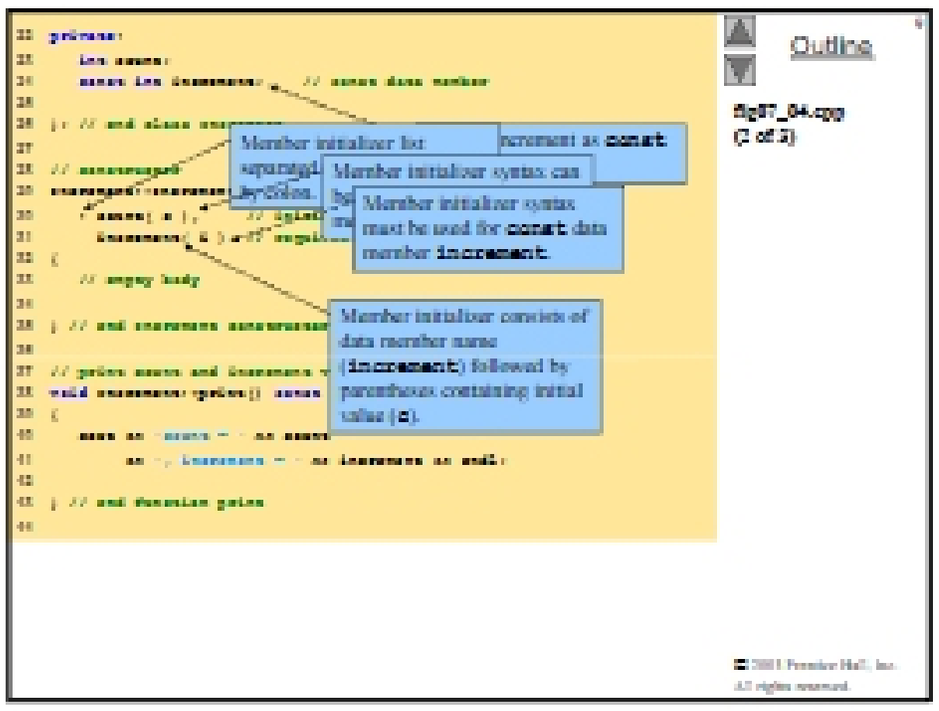
- ~/Class/cisc181/examples
- Driver: 22-date.cc; Header: date.h;implementation: date.cc

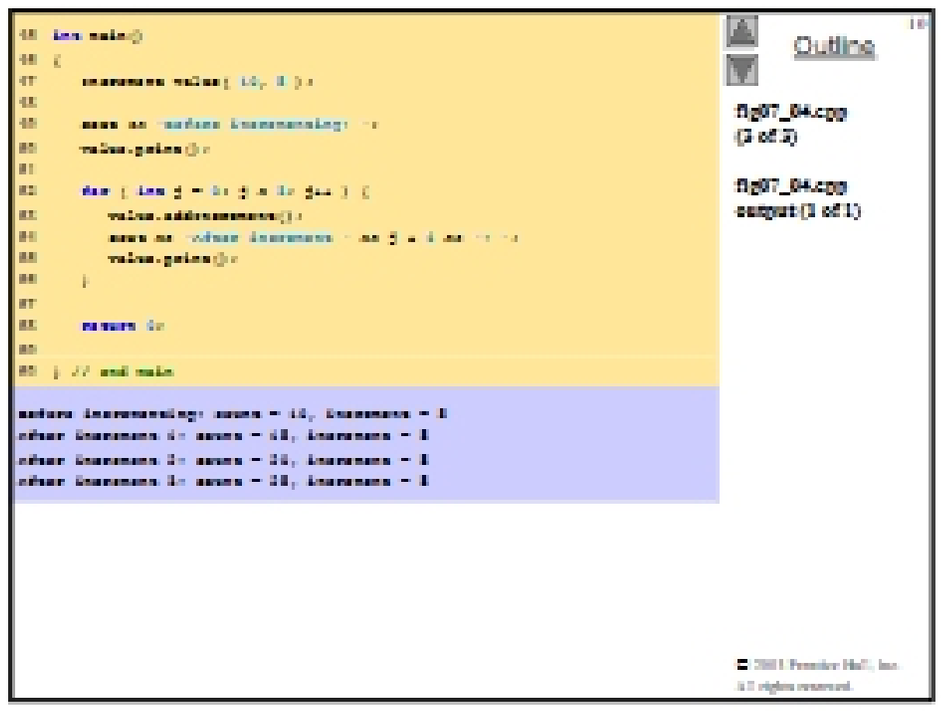
- This particular solutions does not use const objects – but you can see that the print functions allow the date variable to be const (see next example in slides)

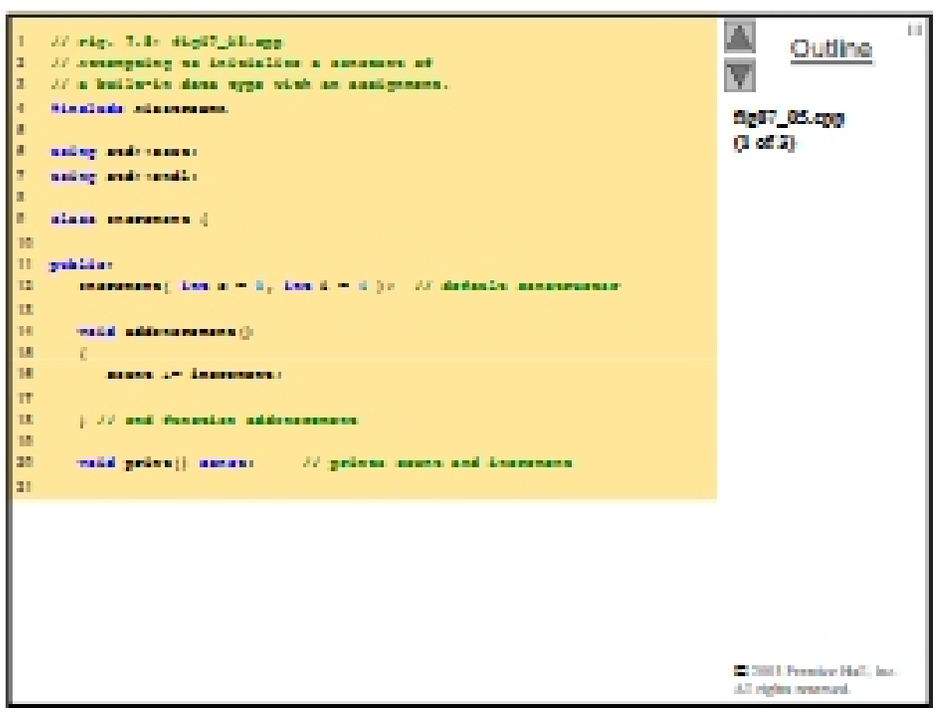
7.2 `const` (Constant) Objects and `const` Member Functions

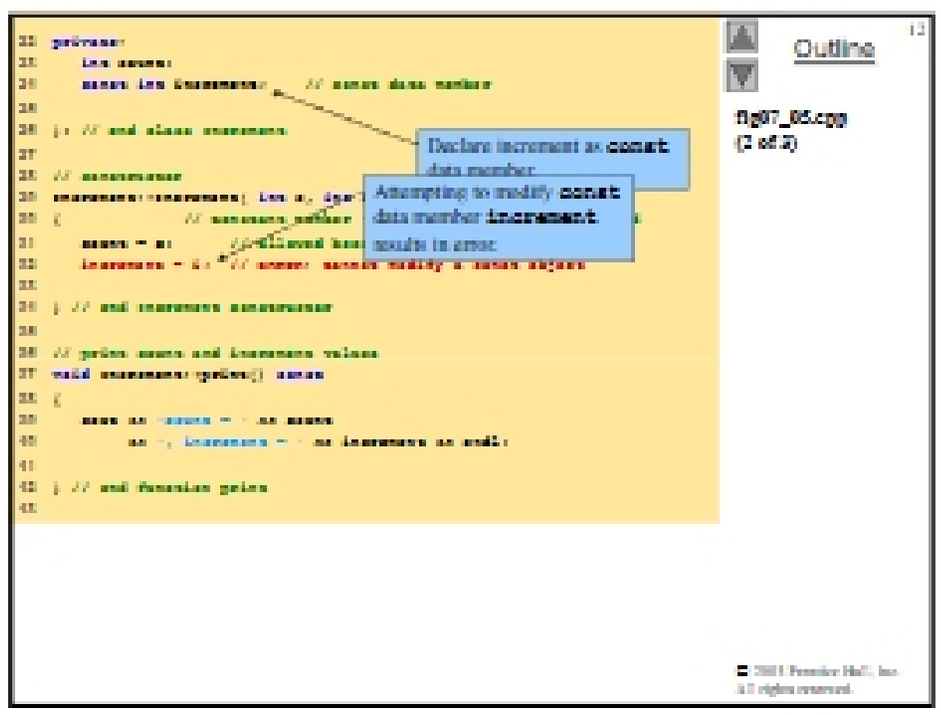
- Member initializer syntax
  - Initializing with member initializer syntax
    - Can be used for
      - All data members
    - Must be used for
      - **`const`** data members
      - Data members that are references











Outline

fig07_05.cpp
(2 of 2)

fig07_05.cpp
output (3 of 3)

Not using member initializer syntax to initialize **const** data member **increment** results in error.

Attempting to modify **const** data member **increment** results in error.

7.3 Composition: Objects as Members of Classes

- **Composition**
  - Class has objects of other classes as members
- **Construction of objects**
  - Member objects constructed in order declared
    - Not in order of constructor's member initializer list
    - Constructed before enclosing class objects (host objects)

Outline

date1.h (1 of 1)

Note no constructor with parameter of type **Date**. Recall compiler provides default copy constructor.

Outline

date1.cpp (1 of 2)

Outline

date1.cpp (2 of 2)

No arguments; each member function contains implicit handle to object on which it operates.

Output to show timing of destructors.

Outline

date1.cpp (3 of 2)