Lecture 9 – Classes

CST238 – Intro to Data Structures
YoungJoon Byun
ITCD

1

Lecture Objectives

- After completion of this lecture, you will be able to

2

Chapter 4: More about OOP and ADTs Classes

- 4.1 Procedural vs. Object-Oriented Programming
- 4.2 Classes
- **4.3 Example: A First Version of a User-Defined Time Class**
- 4.4 Class Constructors
- 4.5 Other Class Operators

3

A Class Library

- Class declaration is placed in a header file
  - The file has `.h` extension
  - It typically contains data items and prototypes
- Implementation file
  - The file has the same prefix as the header file.
  - But it has `.cpp` extension
- A program that uses the class library is called a client program (or a driver program).

Time.h – Interface for `Time` Class

```
// Figure 4.2 of text
#include <iostream>

class Time
{
public:
   void set(unsigned hours, unsigned minutes, char am_pm);
   void display(ostream & out) const;

private:
   unsigned myHours;
   unsigned myMinutes;
   char myAMorPM;        // 'A' or 'P'
   unsigned myMilTime;   // military time equivalent
};
```

Time.cpp – Implementation of `Time` Class (1 of 2)

```
#include <iostream>    // Figure 4.3 of text
using namespace std;
#include "Time.h"
// Prototype of utility function
int toMilitary (unsigned hours, unsigned minutes, char am_pm);

void Time::set(unsigned hours, unsigned minutes, char am_pm) {
  if (hours >= 1 && hours <= 12 && minutes >= 0 && minutes <= 59 &&
      (am_pm == 'A' || am_pm == 'P')) {
    myHours = hours;
    myMinutes = minutes;
    myAMorPM = am_pm;
    myMilTime = toMilitary(hours, minutes, am_pm);
  }
  else
    cerr << "*** Can't set time with these values ***\n";
}
```

Time.cpp – Implementation of `Time` Class (2 of 2)

```
void Time::display(ostream & out) const {
   out << myHours << ':'
         << (myMinutes < 10 ? "0" : "") << myMinutes
         << ' ' << myAMorPM << ".M.  ("
         << myMilTime << " mil. time)";
}

int toMilitary(unsigned hours, unsigned minutes, char am_pm)
{
   if (hours == 12)
        hours = 0;
   return hours * 100 + minutes + (am_pm == 'P' ? 1200 : 0);
}
```

driver.cpp – Test Driver for `Time` Class

```
// Figure 4.4
#include <iostream>
using namespace std;
#include "Time.h"
int main() {
   Time mealTime;
   mealTime.set(5, 30, 'P');
   cout << "We'll be eating at ";
   mealTime.display(cout);
   cout << endl;
   cout << "\nNow trying to set time with illegal hours \n";
   mealTime.set(13, 0, 'A');
}
```

C++ Program Files (Modularity)

- A. Class declarations placed in a header file.
  - e.g. Time.h
  - We call it an **interface file**.
- B. Implementation file.
  - e.g. Time.cpp
- C. Driver file.
  - e.g., driver.cpp that has main() function
  - It is called a client program which uses the library class.

Translating a Library

10

Chapter 4: More about OOP and ADTs Classes

- 4.1 Procedural vs. Object-Oriented Programming
- 4.2 Classes
- 4.3 Example: A First Version of a User-Defined Time Class
- **4.4 Class Constructors**
- 4.5 Other Class Operators

11

Constructor

- A special function member to initialize data members in a class.
  - It has the same name as the class name without any return value.
- Syntax: In class declaration, add

```
class ClassName
{
public:
     ClassName();                    // constructor
     ClassName(parameter_list);  // constructor
    …
private:
   …
};
```

12

Example: Time.h

```
class Time
{
   public:
      Time();
      Time(unsigned initHours,
             unsigned initMinutes,
             char initAMPM);
      …
}
```

13

Implementation of a Constructor

```
ClassName::ClassName ()
  : member_initializer_list
  {
     // body of constructor definition
  }

ClassName::ClassName (parameter_list)
  : member_initializer_list
  {
     // body of constructor definition
  }
```

14

Example: Time.cpp

```
Time::Time()
{
   myHours = 12;
   myMinutes = 0;
   myAMorPM = 'A‘;
   myMilTime = 0;

}
// Or it can be defined as
Time::Time()
  : myHours(12), myMinutes(0), myAMorPM('A'),
   myMilTime(0)
{
}
```

15

Example: Time.cpp

```
Time::Time(unsigned initHours, unsigned initMinutes, char initAMPM)
{
  // Check class invariant
  if (initHours >= 1 && initHours <= 12 &&
      initMinutes >= 0 && initMinutes <= 59 &&
      (initAMPM == 'A' || initAMPM == 'P')) {
        myHours = initHours;
        myMinutes = initMinutes;
        myAMorPM = initAMPM;
        myMilTime = toMilitary(initHours, initMinutes, initAMPM);
  }
  else
     cerr << "*** Invalid initial values ***\n";
}
```

16

Example of Constructor Usage in driver.cpp

17

Chapter 4: More about OOP and ADTs Classes

- 4.1 Procedural vs. Object-Oriented Programming
- 4.2 Classes
- 4.3 Example: A First Version of a User-Defined Time Class
- 4.4 Class Constructors
- **4.5 Other Class Operators**

18

Other Typical Methods : Accessors and Mutators

- Accessors (get functions) and Mutators (set functions) are put in public area to deal with data members in private part.

19

Accessors (or get functions)

- Example: getHours()
  1. In the class declaration:
     unsigned getHours() const;
  2. In the class implementation:
     Time::getHours() const { return myHours; }
  3. In the driver, instead of
     cout << mealTime.myHour; // error!

     cout << mealTime. getHours();

20

Mutators (or set functions)

- Example
  1. In the class declaration:
     void set(unsigned hours, unsigned minutes, char am_pm);
  2. In the class implementation:
     void Time::set(unsigned hours, unsigned minutes, char am_pm) {… myHours = hours; myMinutes = minutes; ……}
  3. In the driver, instead of
     mealTime.myHour = 8; //error!

     mealTime.set(8, 0, ‘P’);

21

Overloading Functions

- Multiple functions with the same name, but with different parameter lists

```
Time();
Time(unsigned initHours,
     unsigned initMinutes,
     char initAMPM);
```

- Why overloading functions?

22

Summary

- Example class: **`Time`** (chap. 4.3)
- Class constructors (chap. 4.4)
- Other class operations (chap 4.5)
- Next Lecture
  - Overloading operators (chap 4.5)

23

References

- Larry Nyhoff, *ADTs, Data Structures, and Problem Solving with C++*, 2nd Edition, Prentice-Hall, 2005
- Walter Savitch, *Problem Solving with C++*, 6th Edition, Addison-Wesley, 2006
- Dr. Meng Su's Lecture Notes
  http://cs.bd.psu.edu/~mus11/122Fa06/cse122Fa06.htm

24