CS202 – Programming Systems

Week 03
Constructor - Detructor

Cảm ơn thầy Trần Duy Quang đã cung cấp template cho môn học

Department of Software Engineering-FIT-VNU-HCMUS

1

Notes

Create a single solution/folder to store your source code in a week.

Then, create a project/sub-folder to store your source code of each assignment.

The source code in an assignment should have at least 3 files:

- A header file (.h): struct definition, function prototypes/definition.
- A source file (.cpp): function implementation.
- Another source file (.cpp): named YourID_Ex01.cpp, main function. Replace 01 by id of an assignment.

Make sure your source code was built correctly. Use many test cases to check your code before submitting to Moodle.

2

Content

In this lab, we will review the following topics:

- How do constructors and destructors work in C++?

3 Assignments

A: YY: 01

H: YY: 05

For assignment 01 – 04, please do the following tasks:

1. Implement **at least 5 constructors**. Also add a `cout` statement to show that the method in called.
2. Implement the destructor. Also add a `cout` statement to show that the method in called.
3. Implement a `toString()` method.
4. Implement a `clone()` method.

Finally, in the main function, create a series of objects and output them to the console. Run the program and inspect the output.

For example, with the Point2D class:

1. 5 constructors
    a. `Point2D();`
    b. `Point2D (int x);`
    c. `Point2D (int x, int y);`
    d. `Point2D (const Point2D &other);`
    e. `Point2D (string s); // s = "15,-2"`
    f. `cout` statement: `cout << "Point2D::Default cons" << endl;`
2. Destructor: `~Point2D();`
    a. `cout` statement: `cout << "Point2D::Destructor" << endl;`
3. `toString()` method: Point 15, -2 => string "15,-2)
4. `clone()` method: `p2 = p1.clone();`

3.1 Assignment 1

Class. Point2D: x, y

3.2 Assignment 2

Class Triangle: Point2D A, B, C

3.3 Assignment 3

Class MyIntArray: int*a, int n

3.4 Assignment 4

Student: int id, char* fullname, char* address, double gpa

3.5 Assignment 5

Draw a class diagram and write a program in C++ with OOP style that can be used by a small theater to sell tickets for performances. The theater's auditorium has 15 rows of seats, with 30 seats in each row. The program should display a screen that shows which seats are available and which are taken. For example, the following screen shows a chart depicting each seat in the theater. Seats that are taken are represented by an * symbol, and seats that are available are represented by a # symbol:

```
                        Seats
            123456789012345678901234567890
Row  1      ***###***###*########*****####
Row  2      ####*************####*******##
Row  3      **###**********########****###
Row  4      **######**************##******
Row  5      ********#####*********########
Row  6      ##############************####
Row  7      #######************###########
Row  8      ************##****############
Row  9      #########*****############****
Row 10      #####*************############
Row 11      #**********#################**
Row 12      #############********########*
Row 13      ###***********########**######
Row 14      ##############################
Row 15      ##############################
```

Here is a list of tasks this program must perform:

- When the program begins, it should ask the user to enter the seat prices for each row. The prices can be stored in a separate array. (Alternatively, the prices may be read from a file.)
- Once the prices are entered, the program should display a seating chart similar to the one shown above. The user may enter the row and seat numbers for tickets being sold. Every time a ticket or group of tickets is purchased, the program should display the total ticket prices and update the seating chart.

- The program should keep a total of all ticket sales. The user should be given an option of viewing this amount.
- The program should also give the user an option to see a list of how many seats have been sold, how many seats are available in each row, and how many seats are available in the entire auditorium.

Input Validation: When tickets are being sold, do not accept row or seat numbers that do not exist. When someone requests a particular seat, the program should make sure that seat is available before it is sold.