창의적 소프트웨어 프로그래밍 Lab 8

Handed out : Thu, Oct 13, 2022

Due : Thu, Oct 13, 2022, 23:59 (NO SCORE for late submissions!)

Submit your file on LMS.

Thu, Oct 20, 2022 : Midterm

Fri, Oct 21, 2022 : NO class

1. Write a program for basic shapes.
    A. Define the class Circle and class Rectangle for circles and rectangles, respectively.
    B. Each class has member functions calculating the area and the perimeter of each shape.
    C. class Circle has the x, y coordinate values of the center point and the radius as member variables.
    D. class Rectangle has the x, y coordinate values of the top-left and bottom-right corners as member variables.
    E. Repeatedly print out the information of the shapes created according to user input
    F. Use 3.14 for pi.
    G. This program should take user input repeatedly
    H.
        i. 'C', x coordinate of center, y coordinate of center, radius - Create a circle.
        ii. 'R', x coordinate of top left corner, y coordinate of top left corner, x coordinate of bottom right corner, and y coordinate of bottom right corner - Create a rectangle
        iii. 'Q' - Quit the program
        iv. Assume that all inputs are valid. You do not need to handle wrong inputs.

I. Output: The area and perimeter of the shape

J. Files to submit:

i. main.cpp – main() must be in this file.

ii. shapes.h – Class definitions

iii. shapes.cpp – Class member function definitions (implementations)

iv. A CMakeLists.txt to generate the executable.

```
$ ./simple_shape

shape?
C 1 1 1
area: 3.14, perimeter: 6.28

shape?
R 1 5 5 1
area: 16, perimeter: 16

shape?
Q
$
```

2. Write a program for 'minimal fighter'.

A. Define the class MinimalFighter that has the following members:

i. private member variables

1. int mHp – Health point

2. int mPower – Attack power

3. FighterStatus mStatus – Invalid, Alive, or Dead - defined by enum in the below code skeleton

ii. Constructors

1. MinimalFighter() – Initialize a fighter with mHp=0, mPower=0, mStatus=Invalid

2. MinimalFighter(int _hp, int _power) – Initialize a fighter with give hp and power values and mStatus=Alive or Dead (see the rules below).

iii. public member functions

1. int hp() – get mHp
2. int power() – get mPower
3. FighterStatus status() – get mStatus
4. void setHp(int _hp) – set mHp
5. void hit(MinimalFighter *_enemy) – run Hit command (see the rules below)
6. void attack(MinimalFighter *_enemy) – run Attack command (see the rules below)
7. void fight(MinimalFighter *_enemy) – run Fight command (see the rules below)

B. Rules

i. All fighters will die when their health point is less than or equal to 0 (status = Dead).

ii. Hit – Two fighters attack each other at the same time. Attacking reduces the opponent's health point by attack power.

iii. Attack – Fighter1 attacks Fighter2. Fighter2's health is reduced by attack power, and Fighter1's attack power is reduced to 0. Fighter1's health point remains unchanged.

iv. Fight – Two fighters attack each other until one of them dies. If they die at the same time, both are considered dead.

v. Example:

H2, P1 hit H1, P1 -> H1, P1 / DEAD (H0, P1)
H3, P1 hit H3, P1 -> H2, P1 / H2, P1
H1, P3 attack H3, P1 -> H1, P0 / DEAD
H4, P1 fight H3, P1 -> H3, P1 / H2, P1 -> H2, P1 / H1, P1 -> H1, P1 / DEAD

C. This program should take user input repeatedly

D. Use the code skeleton (minimal_fighter.h, main.cpp).

i. You must complete the definition of class MinimalFighter in minimal_fighter.h.

ii. You must create an additional C++ source file for member function definitions.

iii. Just use main.cpp as is. Do not modify it.

E. Input:

i. The health point and attack power of Fighter1 (positive integers or zero), Action (H, A, F), The health point and attack power of Fighter2

ii. 'q' to quit the program

F. Output: Output the state of both fighters after the action (in the case of Fight, only the final result)

G. Files to submit:

i. main.cpp – The same file in the code skeleton.

ii. minimal_fighter.h – Class definitions

iii. minimal_fighter.cpp – Class member function definitions (implementations)

iv. A CMakeLists.txt to generate the executable

```
$ ./minimal_fighter
2 1 H 1 1
H1, P1 / DEAD
3 1 H 3 1
H2, P1 / H2, P1
1 3 A 3 1
H1, P0 / DEAD
4 1 F 3 1
H1, P1 / DEAD
q
$
```

3. Write a program for a bank account manager.

A. Class Account has the account ID and balance as member variables

B. Class AccountManager has accounts (as a dynamically or statically allocated array., do not use STL) and and the number of accounts opened as member variables.

C. Class AccountManager has member functions for depositing, withdrawing,

transferring and checking balances.

D. The maximum amount the account can hold is 1 million won, the minimum amount is 0 won, and there is one deposit / withdrawal / transfer limit.

E. Display the balance of accounts after each task.

F. Newly created account balance must be 0 won.

G. The maximum number of accounts the account manager can hold is 10, and the ID are given from 0 to 9 in order of account creations.

H. You need to handle exceptions for creating more than 10 accounts.

I. You don't need to handle command typos or type errors.

J. This program should take user input repeatedly

K. Input:

   i. 'N' – Create an account

   ii. 'D', account ID, deposit amount - Deposit

   iii. 'W', account ID, withdrawal amount – Withdraw

   iv. 'T', from account ID, to account ID, amount – Transfer

   v. 'Q' – Quit the program

L. Output: The result of each task (see the following example)

M. Files to submit:

   i. main.cpp – main() must be in this file.

   ii. accounts.h – Class definitions

   iii. accounts.cpp – Class member function definitions (implementations)

   iv. A CMakeLists.txt to generate the executable

```
$ ./simple_account

Job?
D 0 50000
Account does not exist
```

```
Job?
N
Account for user 0 registered
Balance of user 0: 0

Job?
D 0 50000
Success: Deposit to user 0 50000
Balance of user 0: 50000

Job?
W 0 100000
Failure: Withdraw from user 0 100000
Balance of user 0: 50000

Job?
N
Account for user 1 registered
Balance of user 1: 0

Job?
T 1 0 30000
Failure: Transfer from user 1 to user 0 30000
Balance of user 0: 50000
Balance of user 1: 0

Job?
T 0 1 30000
Success: Transfer from user 0 to user 1 30000
Balance of user 0: 20000
Balance of user 1: 30000

Job?
Q
$
```