CS2 PRAC Exam Program: Rectangle

Define a Rectangle class that has the following members:

width: double

length: double

Rectangle();

Rectangle(w:double, l:double);

setWidth(w: double) : void

setLength(l: double): void

getWidth(): double

getLength(): double

getArea(): double

|  |  |
| --- | --- |
| **Answer Key** |  |
| class Rectangle {  private:  double width;  double length;  public:  Rectangle(){};  Rectangle(double w, double l){  width = w;  length = l;  };  void setWidth(double w){width=w};  void setLength(double){length = l};  double getWidth(){return width};  double getLength(){return length};  double getArea();  };  double Rectangle::getArea(){  return width\*length;  }  HAS TO BE DEFINED OUTSIDE OF THE CLASS (use scope resolution operator and class name) | |

CS2 PRAC Exam Program: Instructor Class

3. The outline of the Instructor class is given below. Complete the implementation following the specifications. (22 points)

struct Course {

string name;

string term;

double studentAvgGrade;

};

class Instructor {

private:

string name;

string department;

int ID;

Course \*courseTaught;

int courseNum;

int courseEntered;

public:

//constructor that set the name, dept, ID, the size of the courseTaught and allocates memory for it

Instructor(string name, string dept, int id, int s) {

this->name = name;

department = dept;

ID = Id;

courseNum = s;

courseTaught = new Course[s];

courseEnetered = 0;

}

//default instructor

Instructor() {

name = “”;

department = ””;

ID = 0;

courseNum = 0;

courseEntered = 0;

}

//destructor

~Instructor() { //4 points

if (courseNum > 0) {delete [] courseTaught;}

}

void addCourse(string n, string term, double avgGrade);

}

//your code for the regular implementation of the addCourse function

ANSWER:

if (courseEnetered >= courseNum)

{

Course \*tmp;

tmp = new Course[courseNum + 1];

for (int i=0; i<courseNum; i++){

tmp[i] = courseTaught[i];

}

if (courseNum > 0)

{

delete [] courseTaught;

}

courseTaught = tmp;

courseNum ++;

}

courseTaught[courseEntered].name = n;

courseTaught[courseEntered].term = term;

courseTaught[courseEntered].studentAveGrage = avgGrade;

courseEntered ++;

CS2 PRAC Exam Program: Array of structs

Assume a structured type, PrecinctReport, with these fields, address (a string ), felonies, murders, and robberies. Declare an array named allPrecincts with NPRECINCTS elements, each of type PrecinctReport. (NPRECINCTS is a pre-declared constant.)

ANSWERS:

PrecinctReport allPrecincts[NPRECINCTS];

OR

struct PrecinctReport

{

string address;

string felonies;

string murders;

string robberies;

};

PrecinctReport allPrecincts[NPRECINCTS];

OR

struct PrecinctReport allPrecincts[NPRECINCTS];

CS2 PRAC Exam: Operator Overloading

Assume the existence of a Window class with integer data members width and height. Overload the << operator for the Window class -- i.e., write a nonmember ostream-returning function that accepts a reference to an ostream object and a constant reference to a Window object and sends the following to the ostream: 'a (width x height) window' (without the quotes and with width and height replaced by the actual width and height of the window. Thus for example, if the window had width=80 and height=20, << would send 'a (80 x 20) window' to the ostream object.) Don't forget to have the function return the proper value as well. Assume the operator has been declared a friend in the Window class.

friend ostream &operator <<(ostream &os, const Window &window) {

os << "a (" << window.width << " x " << window.height << ") window";

return os;

}

OR

friend ostream& operator << (ostream& stm, Window& w)

{

return stm << "a ("<<w.width << " x " << w.height << ") window";

}

OR

friend ostream &operator <<(ostream &os, const Window &window) {

os << "a (" << window.width << " x " << window.height << ") window";

return os;

}

CS2 PRAC EXAM: More Overloading

 Assume a class named Collection exists. Write the header for a member function that

overloads the [] operator for that class.

ANSWER:

Collection Collection::operator[](const Collection &sub)

CS2 PRAC EXAM: Defining a Windows class

Objects of the Window class require a width (integer) and a height (integer) be specified (in that order) upon definition. Define an object named window, of type Window, corresponding to a 80 x 20 window.

ANSWER:

Window window(80, 20);

-OR-

Window window = Window(80, 20);

-OR-  
  
width = 80;

height = 20;

Window window(width, height);

-OR

Window window(width = 80, height = 20);

U1 Arrays of objects

Assume a class Window with accessor method getWidth that accepts no parameters and returns an integer. Assume further an array of 3 Window elements named winarr, has been declared and initialized . Write a sequence of statements that prints out the width of the widest window in the array .

int maxIndex = 0;

for (int i = 1; i < 3; i++)

if (winarr[i].getWidth() > winarr[maxIndex].getWidth()) maxIndex = i;

cout << winarr[maxIndex].getWidth() << endl;

OR

if(winarr[0].getWidth() > winarr[1].getWidth())

{

if(winarr[0].getWidth() > winarr[2].getWidth())

cout << winarr[0].getWidth();

else

cout << winarr[2].getWidth();

}

else

{

if(winarr[1].getWidth() > winarr[2].getWidth())

cout << winarr[1].getWidth();

else

cout << winarr[2].getWidth();

}

OR

if(winarr[0].getWidth()>winarr[1].getWidth() &&

winarr[0].getWidth()>winarr[2].getWidth())

cout<<winarr[0].getWidth();

else if(winarr[1].getWidth()>winarr[0].getWidth() &&

winarr[1].getWidth()>winarr[2].getWidth())

cout<<winarr[1].getWidth();

else if(winarr[2].getWidth()>winarr[0].getWidth() &&

winarr[2].getWidth()>winarr[1].getWidth())

cout<<winarr[2].getWidth();

else if(winarr[0].getWidth()==winarr[0].getWidth() &&

winarr[1].getWidth()==winarr[1].getWidth())

cout<<winarr[2].getWidth();

OR

int maxIndex = 0;

for (int i = 1; i < 3; i++)

if (winarr[i].getWidth() > winarr[maxIndex].getWidth()) maxIndex = i;

cout << winarr[maxIndex].getWidth() << endl;

OR

int x = winarr[0].getWidth();

int y = winarr[1].getWidth();

int z = winarr[2].getWidth();

if (x > y && x > z) {

cout<< x;

} else if (y > z) {

cout<<y;

} else {

cout<<z;

}

OR

int x = winarr[0].getWidth();

int y = winarr[1].getWidth();

int z = winarr[2].getWidth();

if(x > y && x > z){

cout << x;

}

else if(y > x && y > z){

cout << y;

}

else{

cout << z;

}

OR

int max\_width = winarr[0].getWidth();

max\_width = max(max\_width, winarr[1].getWidth());

max\_width = max(max\_width, winarr[2].getWidth());

cout<<max\_width;