Assignment -33 A Job Ready Bootcamp in C++, DSA and IOT MySirG

Virtual Function, Abstract Class

1. Create a base class called shape. Use this class to store two double type values that

could be used to compute the area of figures. Derive two specific classes called

triangle and rectangle from the base shape. Add to the base class, a member

function get\_data() to initialise base class data members and another member

function display\_area() to compute and display the area of figures. Make

display\_area() as a virtual function and redefine this function in the derived classes to

suit their requirements.

Using these three classes, design a program that will accept dimensions of a

triangle or a rectangle interactively, and display the area.

Remember the two values given as input will be treated as lengths of two sides in

the case of rectangles, and as base and height in the case of the triangles, and used

as follows:

Area of rectangle = x \* y Area of triangle = 1/2 \* x \* y

#include <iostream>

using namespace std;

class shape

{

protected:

double x, y;

public:

void setData(int x, int y)

{

this->x = x;

this->y = y;

}

virtual void displayArea() = 0;

};

class Triangle : public shape

{

public:

void displayArea()

{

cout << "\nTriangle area is: " << x \* y / 2.0;

}

};

class Rectangle : public shape

{

public:

void displayArea()

{

cout << "\nRectangle area is: " << x \* y;

}

};

int main()

{

Triangle t;

Rectangle r;

t.setData(13, 10);

r.setData(14, 18);

t.displayArea();

r.displayArea();

return 0;

}

2. Extend the above program to display the area of circles. This requires the addition of

a new derived class 'circle' that computes the area of a circle. Remember, for a circle

we need only one value, its radius, but the get\_data() function in the base class

requires two values to be passed. (Hint: Make the second argument of get\_data()

function as a default one with zero value.)

#include <iostream>

using namespace std;

class shape

{

protected:

double x, y;

public:

void setData(int x, int y = 0)

{

this->x = x;

this->y = y;

}

virtual void displayArea() = 0;

};

class Triangle : public shape

{

public:

void displayArea()

{

cout << "\nTriangle area is: " << x \* y / 2.0;

}

};

class Rectangle : public shape

{

public:

void displayArea()

{

cout << "\nRectangle area is: " << x \* y;

}

};

class Circle : public shape

{

public:

void displayArea()

{

cout << "\nCircle area is: " << 3.14 \* x \* x;

}

};

int main()

{

Triangle t;

Rectangle r;

Circle c;

t.setData(13, 10);

r.setData(14, 18);

c.setData(5);

t.displayArea();

r.displayArea();

c.displayArea();

return 0;

}

3. Using the concept of pointers, write a function that swaps the private data values of

two objects of the same class type.

#include <iostream>

using namespace std;

class A

{

private:

int x;

public:

A(int x) { this->x = x; }

void swap(A \*num)

{

int temp = x;

x = num->x;

num->x = temp;

}

void showData() { cout << "X = " << x << endl; }

};

int main()

{

A obj1(5), obj2(3);

obj1.showData();

obj2.showData();

obj1.swap(&obj2); // you can use also reference

obj1.showData();

obj2.showData();

return 0;

}

4. Create a base class called shape. Use this class to store 2 double type values that

could be used to compute the area of figures. Derive 2 specific classes called triangle

and rectangle from the base shape. Add to the base class a member function

get\_data() to initialise base class data members and another member function

display\_area() to compute and display the area of figures. Make display\_area() as a

virtual function and redefine this function in derived classes to suit their requirements.

Using these 3 classes, design a program that will accept the dimensions of the

shapes interactively and display area.

#include <iostream>

using namespace std;

class shape

{

protected:

double l1, l2;

public:

void setData(int x, int y)

{

l1 = x;

l2 = y;

}

virtual void displayArea() = 0;

};

class Triangle : public shape

{

public:

void displayArea() { cout << "\nArea of triangle is: " << 0.5 \* l1 \* l2; }

};

class Rectangle : public shape

{

public:

void displayArea() { cout << "\nArea of Rectangle is: " << l1 \* l2; }

};

int main()

{

Triangle t;

Rectangle r;

t.setData(1, 2);

r.setData(1, 2);

t.displayArea();

r.displayArea();

return 0;

}

5. Create a base class called Photon. Use this class to store double type value of

wavelength that could be used to calculate photon energy. Create class

calculate\_photonEnergy which will photon energy.

Using these classes, calculate photon energy.

#include <iostream>

#include <math.h>

using namespace std;

class Photon

{

protected:

double waveLength;

Photon(double waveLength) { this->waveLength = waveLength / pow(10, -10); }

};

class calculatePhotonEnergy : public Photon

{

private:

double energy;

public:

calculatePhotonEnergy(int p) : Photon(p) {}

void calEnergy()

{

// Photon energy formula

// E = hc/lemda

// where h = 6.626\*10^-34 js

// c = 3 \* 10^-8 ms

energy = (6.626 \* (pow(10, -34)) \* 3 \* pow(10, 8)) / waveLength;

cout << "\nEnergy of photon is: " << energy;

}

};

int main()

{

calculatePhotonEnergy e(0.5);

e.calEnergy();

return 0;

}

6. Extend above to display the area of circles. For a circle,only one value is needed i.e.

radius but in get\_data() function 2 values are passed.

#include <iostream>

using namespace std;

class shape

{

protected:

double x, y;

public:

void setData(int x, int y = 0)

{

this->x = x;

this->y = y;

}

virtual void displayArea() = 0;

};

class Triangle : public shape

{

public:

void displayArea()

{

cout << "\nTriangle area is: " << x \* y / 2.0;

}

};

class Rectangle : public shape

{

public:

void displayArea()

{

cout << "\nRectangle area is: " << x \* y;

}

};

class Circle : public shape

{

public:

void displayArea()

{

cout << "\nCircle area is: " << 3.14 \* x \* x;

}

};

int main()

{

Triangle t;

Rectangle r;

Circle c;

t.setData(13, 10);

r.setData(14, 18);

c.setData(5);

t.displayArea();

r.displayArea();

c.displayArea();

return 0;

}

7. Create a base class called Matrix. Use this class to store 4 int type values that could

be used to calculate determinants and create matrices. Create class

calculate\_determinant which will calculate the determinant of a matrix.

Using these classes, calculate the determinant of the matrix.

#include <iostream>

#include <math.h>

using namespace std;

class Matrix

{

protected:

int num1, num2, num3, num4;

Matrix(int x, int y, int z, int p)

{

num1 = x;

num2 = y;

num3 = z;

num4 = p;

}

};

class cal\_determinant : public Matrix

{

private:

int determinant;

public:

cal\_determinant(int x, int y, int z, int p) : Matrix(x, y, z, p) {}

void calDeterminant()

{

determinant = fabs(num1 \* num4 - num2 \* num3);

cout << "\nDeterminant of matrix is: " << determinant;

}

};

int main()

{

cal\_determinant obj(1, 2, 3, 4);

obj.calDeterminant();

return 0;

}

8. Create a base class called proof. Use this class to store two int type values that could

be used to prove that triangle is a right angled triangle. Create a class compute

which will determine whether a triangle is a right angled triangle.

Using these classes, design a program that will accept dimensions of a triangle, and display the result.

(Summary: Prove that triangle is a right angled triangle using pythagoras theorem).

#include <iostream>

using namespace std;

class Proof

{

protected:

int x, y;

public:

Proof(int x, int y)

{

this->x = x;

this->y = y;

}

};

class Compute : public Proof

{

public:

Compute(int x, int y) : Proof(x, y) {}

void rightAngleTriangle()

{

}

};

int main()

{

Compute c(1, 2);

c.rightAngleTriangle();

return 0;

}

9. Create a base class called volume. Use this class to store two double type values

that could be used to compute the volume of figures. Derive two specific classes

called cube and sphere from the base shape. Add to the base class, a member

function get\_data() to initialise base class data members and another member

function display\_volume() to compute and display the volume of figures. Make

display\_volume() as a virtual function and redefine this function in the derived

classes to suit their requirements.

Using these three classes, design a program that will accept dimensions of a cube

or a sphere interactively, and display the volume.

#include <iostream>

using namespace std;

class volume

{

protected:

double x, y;

public:

void setData(int x, int y = 0)

{

this->x = x;

this->y = y;

}

virtual void displayVolume() = 0;

};

class cube : public volume

{

public:

void displayVolume() { cout << "\nCube volume is: " << x \* x \* x; }

};

class sphere : public volume

{

public:

void displayVolume() { cout << "\nSphere volume is: " << (4 \* 3.14 \* x \* x) / 3; }

};

int main()

{

cube c;

c.setData(2);

sphere s;

s.setData(5);

c.displayVolume();

s.displayVolume();

return 0;

}

10. Create a base class called shape. Use this class to store two double type values that

could be used to compute the area of figures. Derive two specific classes called

square and parallelogram from the base shape. Add to the base class, a member

function get\_data() to initialise base class data members and another member

function display\_area() to compute and display the area of figures. Make

display\_area() as a virtual function and redefine this function in the derived classes to

suit their requirements.

Using these three classes, design a program that will accept dimensions of a

square or a parallelogram interactively, and display the area.