Assignment -32 A Job Ready Bootcamp in C++, DSA and IOT MySirG

Overriding, overloading, constructor in inheritance

1. Create a class FLOAT that contains one float data member. Overload all the four

arithmetic operators so that they can operate on the objects of FLOAT.

#include <bits/stdc++.h>

using namespace std;

class FLOAT

{

private:

float value;

public:

FLOAT() {}

FLOAT(int value) { this->value = value; }

FLOAT operator+(FLOAT x)

{

FLOAT temp;

temp.value = value + x.value;

return temp;

}

FLOAT operator-(FLOAT x)

{

FLOAT temp;

temp.value = value - x.value;

return temp;

}

FLOAT operator\*(FLOAT x)

{

FLOAT temp;

temp.value = value \* x.value;

return temp;

}

FLOAT operator/(FLOAT x)

{

FLOAT temp;

temp.value = value / x.value;

return temp;

}

void display()

{

cout << value << endl;

}

};

int main()

{

FLOAT f1(5), f2(4), f3;

f1 = 6;

f3 = f1 / f2;

f3.display();

return 0;

}

2. Define a class Rectangle and overload area function for different types of data type.

#include <iostream>

using namespace std;

class Rectangle

{

private:

double x, y;

public:

double area(int x, int y)

{

return x \* y;

}

double area(double x, double y) // both for float and double

{

return x \* y;

}

double area(int x, double y)

{

return x \* y;

}

double area(double x, int y)

{

return x \* y;

}

};

int main()

{

Rectangle r;

cout<<r.area(5,4)<<endl;

cout<<r.area(5,4.4)<<endl;

cout<<r.area(5.3,4)<<endl;

return 0;

}

3. Define a base class Animals having member function sound() . Define another

derived class from Animals class named Dogs. You need to override the sound

function of the base class in the derived class.

#include <iostream>

using namespace std;

class Animal

{

private:

string sound;

public:

string Sound() { return sound; }

void setSound(string sound) { this->sound = sound; }

};

class Dog : public Animal

{

private:

string dogSound;

public:

string Sound() { return dogSound; }

void setSound(string dogSound) { this->dogSound = dogSound; }

};

int main()

{

Animal cat;

cat.setSound("Meow");

cout << "\nCat sound is: " << cat.Sound();

Dog petDog;

petDog.setSound("Bhauu");

cout << "\nPet Dog sound is: " << petDog.Sound();

return 0;

}

4. Define a class Addition that can add 2 or 3 numbers of different data types using

function overloading.

#include <iostream>

using namespace std;

class Addition

{

private:

double sum;

public:

void add(int x, int y) { sum = x + y; }

void add(double x, double y) { sum = x + y; }

void add(int x, int y, int z) { sum = x + y + z; }

void add(double x, double y, double z) { sum = x + y + z; }

void showSum() { cout << "\nSum is: " << sum; }

};

int main()

{

return 0;

}

5. Define a class A having multiple constructors. Define another class B derived from

class A. Create derived class constructors and show use of constructor in this single

inheritance.

#include <iostream>

using namespace std;

class Person

{

protected:

string name;

int age;

public:

Person() {}

Person(string name, int age)

{

this->name = name;

this->age = age;

}

};

class Student : public Person

{

private:

int rank;

public:

Student() : Person() {}

Student(int rank, string name, int age) : Person(name, age)

{

this->rank = rank;

}

void showData()

{

cout << "\nStudent Name: " << name << " Age: " << age << " Rank: " << rank;

}

};

int main()

{

Student s(1, "Sachin Payasi", 19);

s.showData();

return 0;

}

6. C++ Program to illustrate the use of Constructors in multilevel inheritance of your

choice.

#include <iostream>

using namespace std;

class Person

{

protected:

string name;

int age;

public:

Person() {}

Person(string name, int age)

{

this->name = name;

this->age = age;

}

void displayPerson()

{

cout << "\nName: " << name << " Age: " << age;

}

};

class Student : public Person

{

protected:

int rank;

public:

Student() : Person() {}

Student(int rank, string name, int age) : Person(name, age)

{

this->rank = rank;

}

void showData()

{

displayPerson();

cout << " Rank: " << rank;

}

};

class Topper : public Student

{

private:

int batch;

public:

Topper() : Student() {}

Topper(int batch, int rank, string name, int age) : Student(rank, name, age)

{

this->batch = batch;

}

void displyeTopper()

{

showData();

cout << " Batch is: " << batch;

}

};

int main()

{

Topper s(3,1, "Ajay Soni", 19);

s.displyeTopper();

return 0;

}

7. C++ Program to illustrate the use of Constructors in single inheritance of your choice.

#include <iostream>

using namespace std;

class Person

{

protected:

string name;

int age;

public:

Person() {}

Person(string name, int age)

{

this->name = name;

this->age = age;

}

};

class Student : public Person

{

private:

int rank;

public:

Student() : Person() {}

Student(int rank, string name, int age) : Person(name, age)

{

this->rank = rank;

}

void showData()

{

cout << "\nStudent Name: " << name << " Age: " << age << " Rank: " << rank;

}

};

int main()

{

Student s(1, "Ajay Soni", 19);

s.showData();

return 0;

}

8. Write a C++ program to find the factorial of a number using copy constructor

#include <iostream>

using namespace std;

class Factorial

{

private:

int n, fact;

public:

Factorial(int x)

{

n = x;

fact = 1;

}

Factorial(Factorial &f)

{

n = f.n;

fact = 1;

}

void calculate()

{

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

fact \*= i;

}

void show()

{

cout << "\nFactorial is: " << fact;

}

};

int main()

{

Factorial f1(6);

f1.calculate();

f1.show();

Factorial f2(f1);

f2.calculate();

f2.show();

return 0;

}

9. Write a C++ program to calculate the area of triangle, rectangle and circle using

constructor overloading. The program should be menu driven.

#include <bits/stdc++.h>

using namespace std;

class Area

{

private:

float areaOfTriangle = 0, areaOfRectangle = 0, areaOfCircle = 0;

public:

Area() {}

Area(int length, int width) { areaOfRectangle = length \* width; }

Area(int r) { areaOfCircle = 3.14 \* r \* r; }

Area(int a, int b, int c)

{

float s = (a + b + c) / 2.0;

areaOfTriangle = sqrt(s \* (s - a) \* (s - b) \* (s - c));

}

void displayAreaTriangle()

{

if (areaOfTriangle == 0)

cout << "This is not a valid triangle.." << endl;

else

cout << "Area of triangle is: " << areaOfTriangle << endl;

}

void displayAreaRectangle()

{

cout << "Area of Rectangle is: " << areaOfRectangle << endl;

}

void displayAreaCircle()

{

cout << "Area of circle is: " << areaOfCircle << endl;

}

};

int main()

{

while (true)

{

cout << "\n1. calculate area of triangle";

cout << "\n2. calculate area of rectangle";

cout << "\n3. calculate area of circle";

cout << "\n4. Exit";

cout << "\nEnter your choice: ";

int choice;

cin >> choice;

system("cls");

if (choice == 1)

{

cout << "\nEnter the value of edge: ";

int a, b, c;

cin >> a >> b >> c;

Area a1(a, b, c);

a1.displayAreaTriangle();

}

else if (choice == 2)

{

int length, width;

cout << "\nEnter the value of length and width: ";

cin >> length >> width;

Area a2(length, width);

a2.displayAreaRectangle();

}

else if (choice == 3)

{

int radius;

cout << "\nEnter the radius: ";

cin >> radius;

Area a3(radius);

a3.displayAreaCircle();

}

else if (choice == 4)

{

exit(0);

}

else

{

cout << "\nWrong option..";

}

}

return 0;

}

10. Create a C++ class for player objects with the following attributes: player no., name,

number of matches and number of goals done in each match. The number of

matches varies for each player. Write a parameterized constructor which initializes

player no., name, number of subjects and creates an array for number of goals and

number of matches dynamically.

#include <iostream>

using namespace std;

class Player

{

private:

int playerNo, noOfMatches;

string playerName;

int \*goals;

public:

Player()

{

cout << "\nEnter player No.: ";

cin >> playerNo;

cout << "\nEnter Player name: ";

cin >> playerName;

cout << "\nEnter Player match: ";

cin >> noOfMatches;

goals = new int[noOfMatches];

cout << "\nEnter no. of goals in per match: ";

for (int i = 0; i < noOfMatches; i++)

{

cout << i + 1 << " ";

cin >> goals[i];

}

}

void display()

{

cout << "\nPlayer No.: " << playerNo;

cout << "\nPlayer name: " << playerName;

cout << "\nEnter Player match: " << noOfMatches;

cout << "\n no. of goals in per match: \n";

for (int i = 0; i < noOfMatches; i++)

cout << i + 1 << " " << goals[i] << endl;

}

};

int main()

{

Player p;

p.display();

return 0;

}