AIM: WAP to implement Constructor Overloading

#include<iostream>

using namespace std;

class Student{

public:

string name;

int sid, YOB, age;

Student(){}

Student(string name, int sid, int yob, int age){

this->name = name;

this->sid = sid;

this->YOB = yob;

this->age = age;

}

Student(string name, int sid, int Yob){

this->name = name;

this->sid = sid;

this->YOB = Yob;

this->age = 2023 - this->YOB;

}

void info(){

cout<<"student name "<<this->name;

cout<<" with student id "<<this->sid;

cout<<" and age is "<<this->age<<endl;

}

};

int main(){

Student tannu;

Student raj = Student("Raj", 1001, 2003, 19);

Student rajVerma("Raj Verma", 1002, 2006);

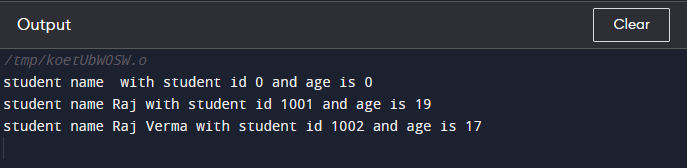
tannu.info();

raj.info();

rajVerma.info();

}

Output:



AIM: WAP to implement Copy Constructor

#include<iostream>

using namespace std;

class Student{

public:

string name;

int sid, YOB, age;

Student(string name, int sid, int Yob){

this->name = name;

this->sid = sid;

this->YOB = Yob;

this->age = 2023 - this->YOB;

}

Student(Student &obj){

this->name = obj.name;

this->sid = obj.sid;

this->age = obj.age;

this->YOB = obj.YOB;

}

void info(){

cout<<"student name "<<this->name;

cout<<" with student id "<<this->sid;

cout<<" and age is "<<this->age<<endl;

}

};

int main(){

Student rajVerma("Raj Verma", 1001, 2003);

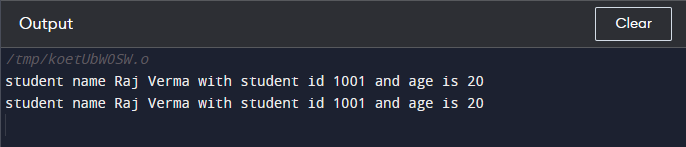
rajVerma.info();

Student rajVerma2 = Student(rajVerma);

rajVerma2.info();

}

Output:



AIM: WAP to implement private inheritance

#include <iostream>

using namespace std;

class Base{

protected:

int a = 0;

int b = -1;

int c = 1;

};

class Derived: private Base{

public:

int get\_a(){

return a;

}

int get\_b(){

return b;

}

int get\_c(){

return c;

}

};

int main() {

Derived obj;

int a = obj.get\_a();

int b = obj.get\_b();

int c = obj.get\_c();

cout<<a<<endl;

cout<<b<<endl;

cout<<c<<endl;

return 0;

}

Output:



AIM: WAP to implement Public Inheritance

#include<iostream>

using namespace std;

class Base{

public:

int pub;

private:

int pvt;

protected:

int pro;

public:

Base(){

pub=0;

pvt=-1;

pro=1;

}

};

class Derived: public Base{

public:

void info(){

cout<<pub<<endl;

//cout<<pvt<<endl;

cout<<pro<<endl;

}

};

int main(){

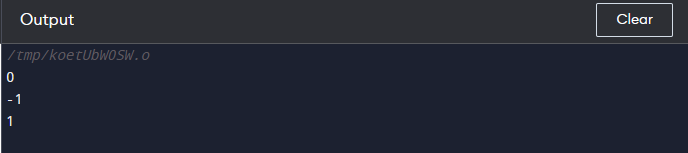
Derived obj;

obj.info();

return 0;

}

Output:



AIM: WAP to implement the use cases of scope resolution operator

#include<iostream>

using namespace std;

int num = 10;

namespace temp{

int num;

};

int main(){

int num;

num = -20;

::num = 10;

temp::num = 0;

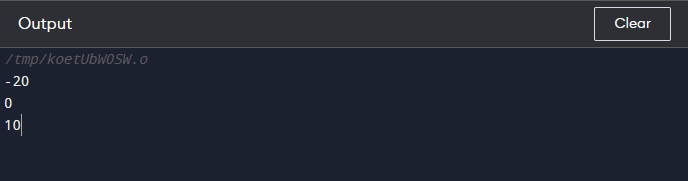
cout<<num<<endl;

cout<<temp::num<<endl;

cout<<::num;

}

Output:



AIM: WAP to take user input in constructor

#include<iostream>

using namespace std;

class Student{

public:

string name;

int sid, YOB, age;

Student(string name, int sid, int Yob){

this->name = name;

this->sid = sid;

this->YOB = Yob;

this->age = 2023 - this->YOB;

}

void info(){

cout<<"student name "<<this->name;

cout<<" with student id "<<this->sid;

cout<<" and age is "<<this->age<<endl;

}

};

int main(){

int id, yob;

string name;

cout<<"Enter name: ";

cin>>name;

cout<<"Enter student id: ";

cin>>id;

cout<<"Enter year of birth: ";

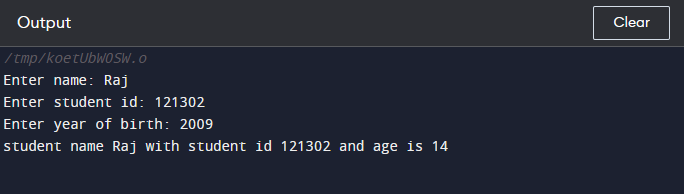
cin>>yob;

Student rajVerma(name, id, yob);

rajVerma.info();

}

Output:



AIM: WAP to implement operator overloading

#include <iostream>

using namespace std;

class Count {

private:

int value;

public:

Count() :{

value = 5;

}

void operator ++ () {

++value;

}

void display() {

cout << "Count: " << value << endl;

}

};

int main() {

Count count1;

++count1;

count1.display();

return 0;

}

Output:

