Static function example and Scope Resolution Operator :--

#include<iostream>

using namespace std;

class X

{

public: static int i;

};

class Y

{

int num=10;

public: void display()

{

cout<<"Number \t"<<num;

}

};

int X::i=1;

int main()

{

X x;

cout<<"Static value is \t"<<x.i<<endl;

Y y;

y.display();

}

Structure example:--

#include<iostream>

using namespace std;

struct College

{

char name[50];

int pincode;

float fee;

};

int main()

{

College c;

cout<<"Enter College Name:";

cin.getline(c.name,50); //getline() is a liberary function, and used to read a line from input..

cout<<"Pincode:";

cin>>c.pincode;

cout<<"College Fee:";

cin>>c.fee;

cout<<"-------------------------------------\t"<<endl;

cout<<"Display the College Details:\t"<<endl;

cout<<"Name of the College:\t"<<c.name<<endl;

cout<<"Pincode of the College:\t"<<c.pincode<<endl;

cout<<"Admission Fee:\t"<<c.fee<<endl;

}

Array of Structure Example:--

#include <iostream>

using namespace std;

struct student

{

int roll\_no;

string name;

int phone\_number;

};

int main(){

struct student stud[2];

int i;

for(i=0; i<2; i++){

cout << "Student " << i + 1 << endl;

cout << "Enter roll no" << endl;

cin >> stud[i].roll\_no;

cout << "Enter name" << endl;

cin >> stud[i].name;

cout << "Enter phone number" << endl;

cin >> stud[i].phone\_number;

}

for(i=0; i<2; i++){

cout << "Student " << i + 1 << endl;

cout << "Roll no : " << stud[i].roll\_no << endl;

cout << "Name : " << stud[i].name << endl;

cout << "Phone no : " << stud[i].phone\_number << endl;

}

return 0;

}

Encapsulation Example:--

#include<iostream>

using namespace std;

class Encapsulation

{

private:

int x;

public:

void set(int a)

{

x =a;

}

int get()

{

return x;

}

};

int main()

{

Encapsulation e;

e.set(5);

cout<<e.get();

return 0;

}

Abstraction Example:--

#include <iostream>

using namespace std;

class Abstraction{

private:

int a, b, c;

public:

void sum(int x, int y)

{

a = x;

b = y;

c = a + b;

cout<<"Sum is : "<<c<<endl;

}

};

int main()

{

Abstraction a;

a.sum(10,10);

return 0;

}

Constructor Example:--

#include <iostream>

using namespace std;

class test {

public:

int y, z;

test()

{

y = 40;

z = 10;

}

};

int main()

{

test a;

cout <<"the sum is: "<< a.y+a.z;

return 1;

}

Constructor and Destructor example:--

#include<iostream>

using namespace std;

class test {

public:

int y, z;

test()

{

y = 6;

z = 10;

}

~test()

{

}

};

int main()

{

test a;

cout <<"the sum is: "<< a.y+a.z;

return 0;

}

Constructor Overloading example:--

#include <iostream>

using namespace std;

class Person {

private:

int age;

public:

Person() {

age = 20;

}

Person(int a) {

age = a;

}

int getAge() {

return age;

}

};

int main()

{

Person person1, person2(45);

cout << "Person1 Age = " << person1.getAge() << endl;

cout << "Person2 Age = " << person2.getAge() << endl;

return 0;

}

Function Overloading:--

#include <iostream>

using namespace std;

class Cal

{

public:

int add(int a,int b)

{

cout<<a+b<<endl;

}

int add(int a, int b, int c)

{

cout<<a+b+c<<endl;

}

};

int main(void) {

Cal C;

cout<<C.add(10,20)<<endl;

cout<<C.add(12,20,23);

return 0;

}

Friend Function Example:--

#include <iostream>

using namespace std;

class Distance {

private:

int meter;

friend int addFive(Distance);

public:

Distance(): meter(0)

{

}

};

int addFive(Distance d)

{

d.meter=5;

return d.meter;

}

int main() {

Distance D;

cout << "Distance: " << addFive(D);

return 0;

}

Friend Class Example:--

#include <iostream>

using namespace std;

class A

{

int x =5;

friend class B;

};

class B

{

public:

void display(A &a)

{

cout<<"value of x is : "<<a.x;

}

};

int main()

{

A a;

B b;

b.display(a);

return 0;

}