1.//simple class and object:

#include<iostream>

using namespace std;

class operating

{

public:

void show()

{

cout<<"class operating";

}

};

int main()

{

operating o1;

o1.show();

return 0;

}

2.// friend class

#include<iostream>

using namespace std;

class A

{

void show()

{

cout<<"class A"<<endl;

}friend class B;

};

class B

{

public:

void show()

{

A a1;

a1.show();

cout<<"class B"<<endl;

}

};

int main()

{

B b1;

b1.show();

return 0;

}

3.// friend function

#include<iostream>

using namespace std;

class os

{

void show1()

{

cout<<"class os"<<endl;

}friend void show2();

};

void show2()

{

os o1;

o1.show1();

cout<<"jai shree ram";

}

int main()

{

show2();

return 0;

}

4.//constructor

#include<iostream>

using namespace std;

class A

{

public:

A()

{

cout<<"class A constructor";

}

};

int main()

{

A a1;

return 0;

}

5.//constructor overloading

#include<iostream>

using namespace std;

class student

{

public:

student(int x,int y)

{

cout<<"sum="<<x+y<<endl;

}

student(int x,int y,int z)

{

cout<<"sum="<<x+y+z<<endl;

}

};

int main(){

student c1(4,8);

student c2(2,5,8);

return 0;

}

6.//Inheritance

#include<iostream>

using namespace std;

class apple

{

public:

void show1()

{

cout<<"class apple"<<endl;

}

};

class fruit:public apple

{

public:

void show()

{

cout<<"class fruit"<<endl;

}

};

int main()

{

fruit f1;

f1.show1();

f1.show();

return 0;

}

7.//function overloading

#include<iostream>

using namespace std;

void show(int a,int b)

{

cout<<"sub="<<a-b<<endl;

}

void show(int a,int b,int c)

{

cout<<"sub="<<a-b-c<<endl;

}

void show(int a,int b,int c,int d)

{

cout<<"sub="<<a-b-c-d<<endl;

}

int main()

{

show(10,8);

show(9,10,6);

show(8,12,4);

return 0;

}

8.//method overriding

#include<iostream>

using namespace std;

class A

{

public:

void show()

{

cout<<"class a"<<endl;

}

};

class B:public A

{

public:

void show()

{

cout<<"class b"<<endl;

}

};

int main()

{

B b1;

b1.show();

return 0;

}

9.//Data encapsulation

#include<iostream>

using namespace std;

class A

{

int x,y;

public:

void get(int a,int b)

{

x=a;

y=b;

}

void show()

{

cout<<"x="<<x<<endl;

cout<<"y="<<y<<endl;

}

};

int main()

{

A a1;

a1.get(10,20);

a1.show();

return 0;

}

10.//Data abstraction

#include<iostream>

using namespace std;

class A

{

public:

int cube(int x)

{

return x\*x\*x;

}

};

int main()

{

A a1;

cout<<a1.cube(10)<<endl;

cout<<a1.cube(5)<<endl;

return 0;

}