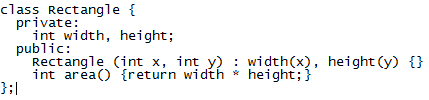
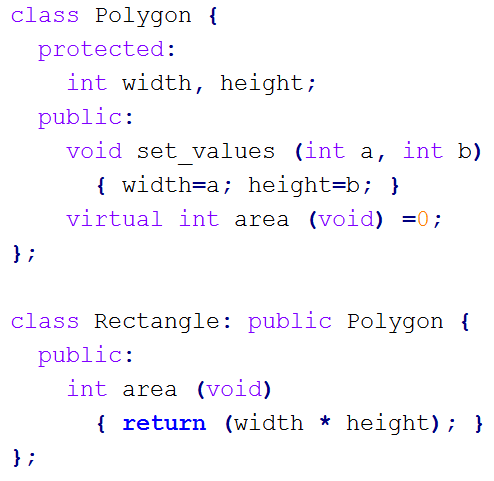
Write a simple class that uses with a private variable and a public member function.



Create an interface class with a pure virtual function. Create another class that inherits from it, and give a concrete implementation of the pure virtual function



Virtual functions in C++ are internally implemented using a virtual function pointer table. Summarize in your own words how a calling virtual functions work using a vtable

Vtable for a class is like a series of memories.

Each unit of memory records a JMP address.

For every class that consists a virtual function, a vtable will be created.

Also this vtable will be shared by every instances of this class.

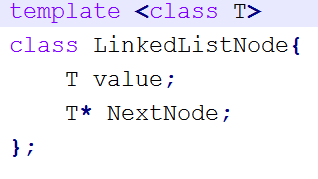
If there are N virtual function, the size of vtable will be 4N.

When using pointer to manage derived class, vtable is very important,

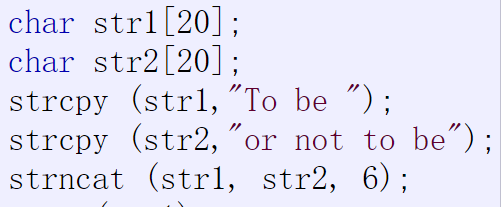
Since it points out every function that needs to be called.

Object created -> virtual table pointer added to hidden member of this object -> hidden code generated in the constructor of each class.

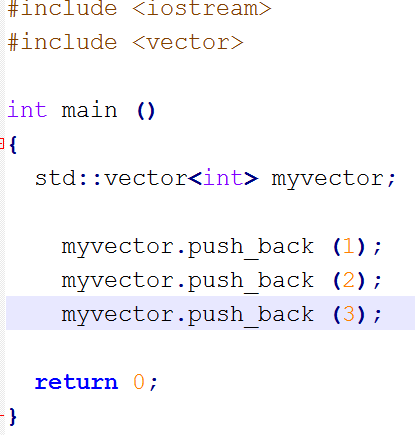
Write a simple class template “LinkedListNode”, which stores an object of type “T” and a pointer to the next node of type “LinkedListNode\*”. It doesn’t need to have any member functions



Concatenate two strings using the std::string class.



Create an empty std::vector then push\_back() the values 1,2,3 to it.



Create a std::map and insert into it the key-value pairs {“one”,1}, {“two”,2}, and {“three”,3}.

int main(int argc, char \*argv[])

{

std::map<std::string, int > numbers;

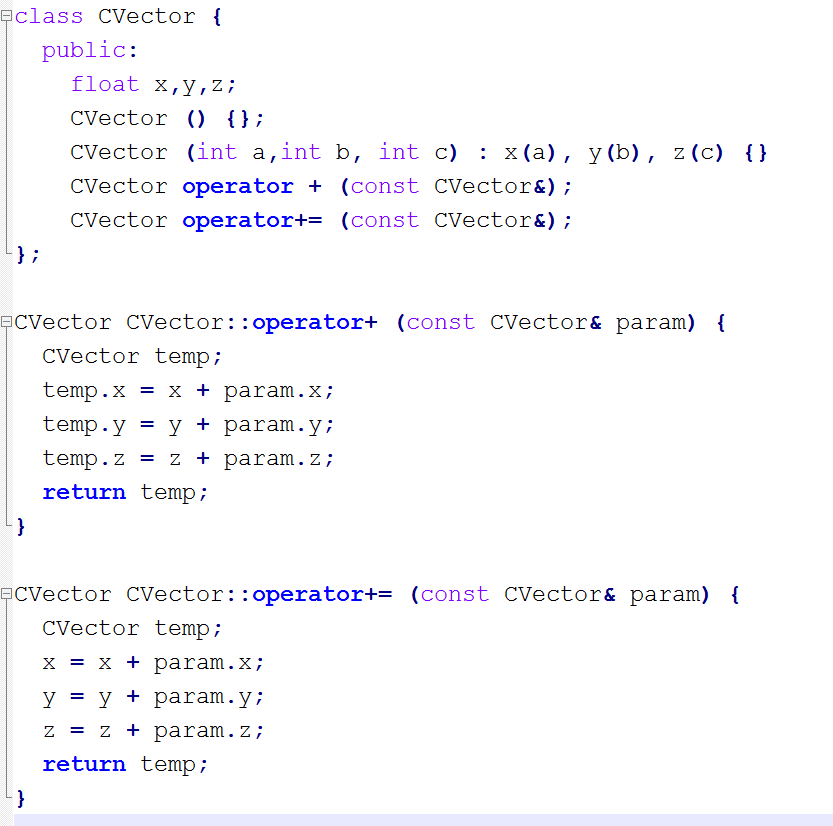
numbers.insert(std::pair<std::string,int>("one",1));

numbers.insert(std::pair<std::string,int>("two",2));

numbers.insert(std::pair<std::string,int>("three",3));

}

Define operator+ and operator+= for the simple 3D vector class below: struct Vector3 { float x, y, z; };



CVector & CVector::operator+=(const CVector &param){

x += param.x;

y += param.y;

z += param.z;

return \*this;

}