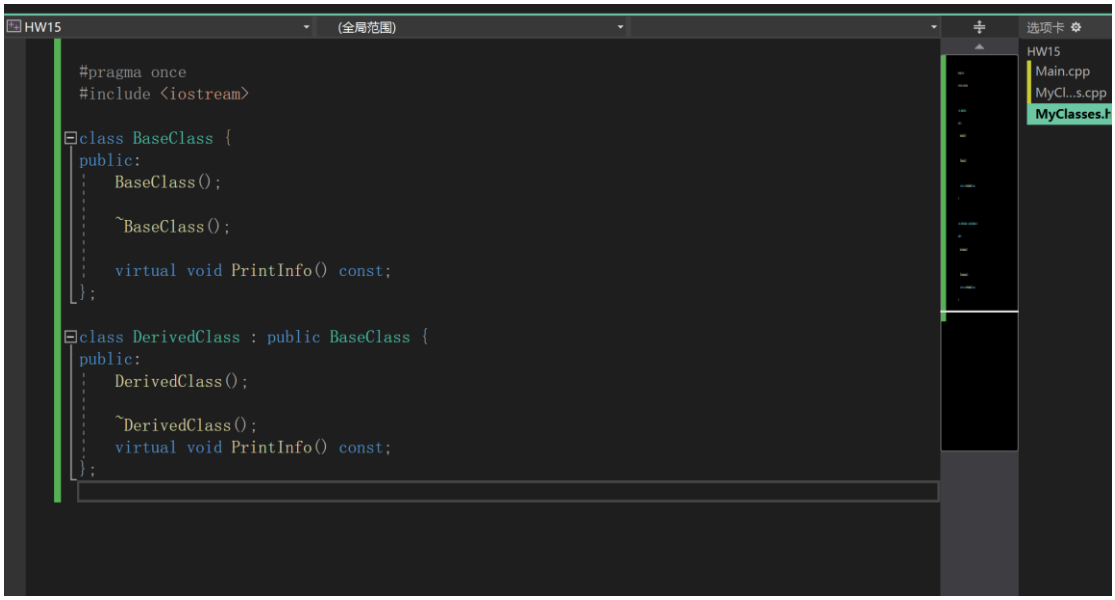
```cpp
#include "MyClasses.h"

int main() {
    BaseClass baseObj;
    DerivedClass derivedObj;
    return 0;
}
```

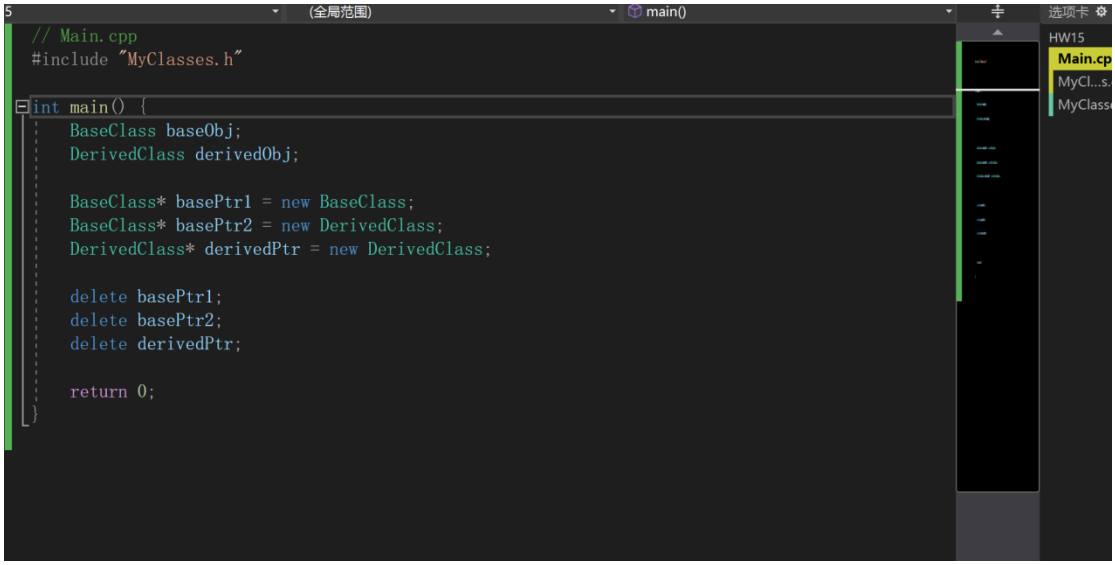
```cpp
#pragma once
#pragma once
#include <iostream>

class BaseClass {
public:
    BaseClass();

    ~BaseClass();
};

class DerivedClass : public BaseClass {
public:
    DerivedClass();
    ~DerivedClass();
};
```

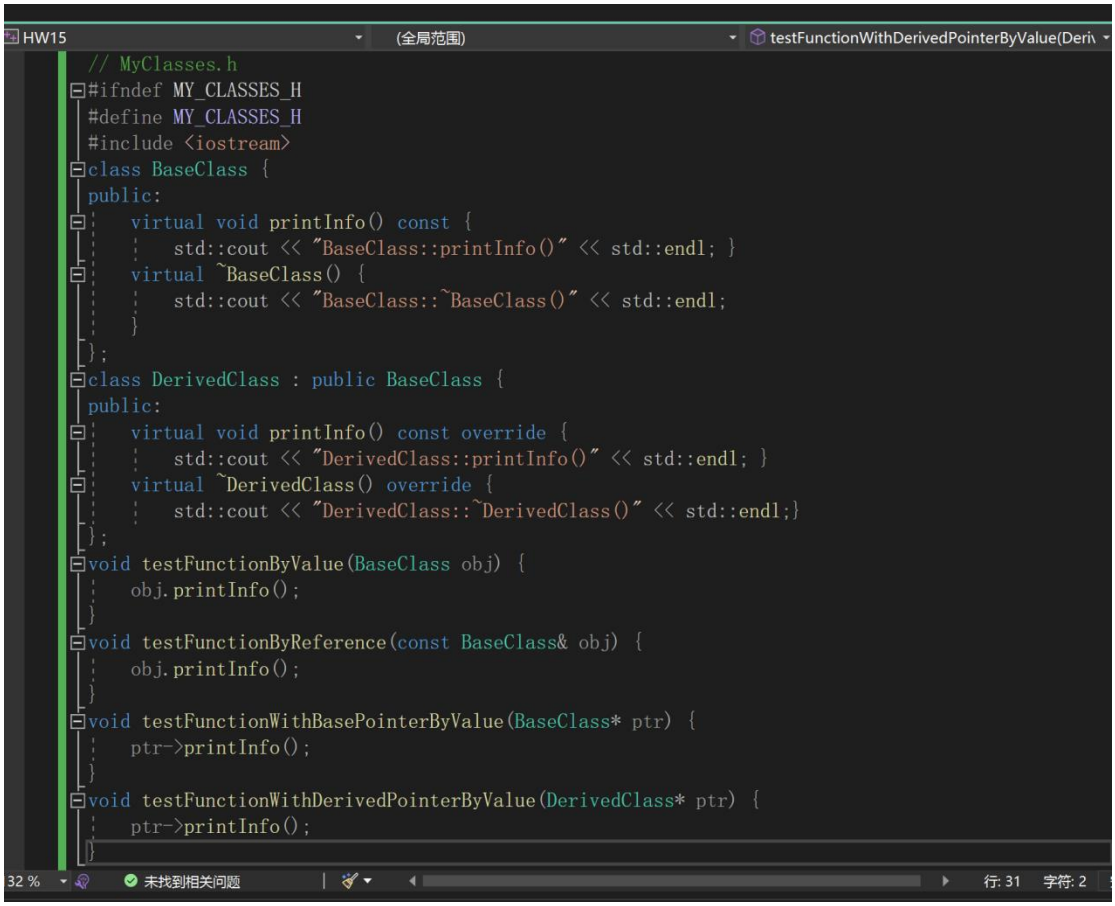
```cpp
#include "MyClasses.h"

BaseClass::BaseClass() {
    std::cout << "BaseClass constructor" << std::endl;
}

BaseClass::~BaseClass() {
    std::cout << "BaseClass destructor" << std::endl;
}

void BaseClass::PrintInfo() const {
    std::cout << "BaseClass PrintInfo" << std::endl;
}
DerivedClass::DerivedClass() {
    std::cout << "DerivedClass constructor" << std::endl;
}

DerivedClass::~DerivedClass() {
    std::cout << "DerivedClass destructor" << std::endl;
}

void DerivedClass::PrintInfo() const {
    std::cout << "DerivedClass PrintInfo" << std::endl;
}
```

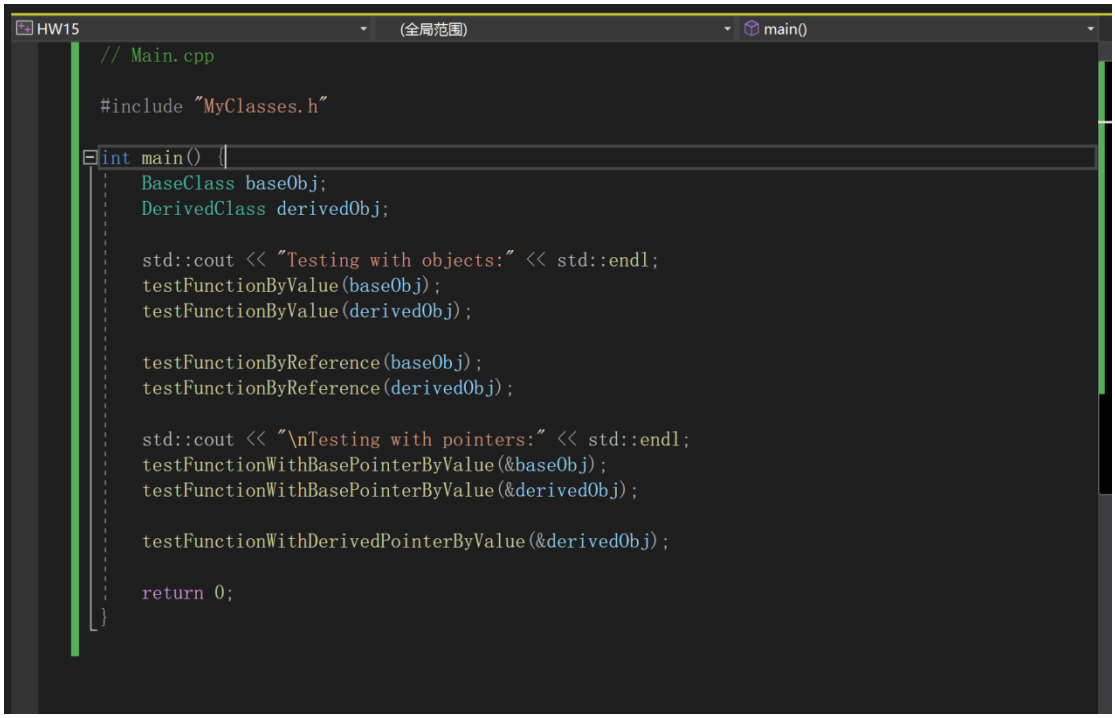


```cpp
#pragma once
#include <iostream>

class BaseClass {
public:
    BaseClass();

    ~BaseClass();

    virtual void PrintInfo() const;
};

class DerivedClass : public BaseClass {
public:
    DerivedClass();

    ~DerivedClass();
    virtual void PrintInfo() const;
};
```


HW15
(全局范围)
main()
选项卡
// Main.cpp
#include "MyClasses.h"
int main() {
BaseClass baseObj;
DerivedClass derivedObj;
BaseClass* basePtr1 = &baseObj;
BaseClass* basePtr2 = &derivedObj;
DerivedClass* derivedPtr = &derivedObj;
std::cout << "Calling PrintInfo using pointers:" << std::endl;
basePtr1->PrintInfo();
basePtr2->PrintInfo();
derivedPtr->PrintInfo();
return 0;
}


(全局范围)
main()
选项卡
// Main.cpp
#include "MyClasses.h"
int main() {
BaseClass baseObj;
DerivedClass derivedObj;
BaseClass* basePtr1 = new BaseClass;
BaseClass* basePtr2 = new DerivedClass;
DerivedClass* derivedPtr = new DerivedClass;
delete basePtr1;
delete basePtr2;
delete derivedPtr;
return 0;
}


HW15
(全局范围)
testFunctionWithDerivedPointerByValue(Deri
// MyClasses.h
#ifndef MY_CLASSES_H
#define MY_CLASSES_H
#include <iostream>
class BaseClass {
public:
virtual void printInfo() const {
std::cout << "BaseClass::printInfo()" << std::endl; }
virtual ~BaseClass() {
std::cout << "BaseClass::~BaseClass()" << std::endl;
}
};
class DerivedClass : public BaseClass {
public:
virtual void printInfo() const override {
std::cout << "DerivedClass::printInfo()" << std::endl; }
virtual ~DerivedClass() override {
std::cout << "DerivedClass::~DerivedClass()" << std::endl;}
};
void testFunctionByValue(BaseClass obj) {
obj.printInfo();
}
void testFunctionByReference(const BaseClass& obj) {
obj.printInfo();
}
void testFunctionWithBasePointerByValue(BaseClass* ptr) {
ptr->printInfo();
}
void testFunctionWithDerivedPointerByValue(DerivedClass* ptr) {
ptr->printInfo();
}
32 %
未找到相关问题
行: 31
字符: 2


HW15
(全局范围)
main()
// Main.cpp
#include "MyClasses.h"
int main() {
BaseClass baseObj;
DerivedClass derivedObj;
std::cout << "Testing with objects:" << std::endl;
testFunctionByValue(baseObj);
testFunctionByValue(derivedObj);
testFunctionByReference(baseObj);
testFunctionByReference(derivedObj);
std::cout << "\nTesting with pointers:" << std::endl;
testFunctionWithBasePointerByValue(&baseObj);
testFunctionWithBasePointerByValue(&derivedObj);
testFunctionWithDerivedPointerByValue(&derivedObj);
return 0;
}