

## 5.5

5. 有以下程序结构, 请分析访问权限。考虑在什么情况下二者不能互相代替。

```
class A
{public:
    void f1();
    int i;
protected:
    void f2();
    int j;
private:
    int k;
};

class B: public A
{public:
    void f3();
protected:
    int m;
private:
    int n;
};

class C: public B
{public:
    void f4();
private:
    int p;
};

int main()
{A a1;
 B b1;
 C c1;
}
```

//A 为基类  
//B 为 A 的公用派生类  
//C 为 B 的公用派生类  
//a1 是基类 A 的对象  
//b1 是派生类 B 的对象  
//c1 是派生类 C 的对象

请问:

- (1) 在 main 函数中能否用 b1.i, b1.j 和 b1.k 引用派生类 B 对象 b1 中基类 A 的成员?
- (2) 派生类 B 中的成员函数能否调用基类 A 中的成员函数 f1 和 f2?
- (3) 派生类 B 中的成员函数能否引用基类 A 中的数据成员 i, j, k?
- (4) 能否在 main 函数中用 c1.i, c1.j, c1.k, c1.m, c1.n, c1.p 引用基类 A 的成员 i, j, k, 派生类 B 的成员 m, n, 以及派生类 C 的成员 p?
- (5) 能否在 main 函数中用 c1.f1(), c1.f2(), c1.f3() 和 c1.f4() 调用 f1, f2, f3, f4 成员函数?

- (6) 派生类 C 的成员函数 f4 能否调用基类 A 中的成员函数 f1, f2 和派生类中的成员函数 f3?

## 5.5 答：

(1). 在 main 函数中可以通过派生类 B 对象 b1 引用基类 A 中的成员 i，因为 i 的访问权限为 public，可以在 A 类外被访问；j 的访问权限为 protected，可以被基类 A 的派生类 B 访问，但不能在 main 函数中被直接访问。成员 k 的访问权限为 private 的成员 k 不能够在 A 类外被直接访问

(2). 可以。访问权限为 public 的成员函数 f1 能够在类 A 外被访问；访问权限为 protected 的成员函数 f2 能够被类 A 在派生类 B 为可见，能够被调用

(3). 能够引用 i 与 j，但不能引用 k；i 的访问权限为 public，可以在 A 类外被访问；j 的访问权限为 protected，可以被基类 A 的派生类 B 访问，故可在 B 的成员函数中使用；k 的访问权限为 private，不能够在 A 类外被直接访问。

(4). A 类中访问权限为 protected 的 j、private 的 k、B 类中访问权限为 protected 的 m、private 的 n、C 类中访问权限为 private 的 p 均不能在 main 函数中被引用；A 类中访问权限为 public 的 i 能够在 main 函数中被访问。因此只有 c1.i 是可行的

(5). 同上，访问权限为 protected 的成员函数 f2() 不能再 main 函数中调用，访问权限为 public 的成员函数 f1()、f3()、f4() 能够在 main 函数中被调用

(6). 访问权限为 public 或 protected 的成员函数对于派生类可见，因此都可以调用

## 5.7

7. 有以下程序, 请完成下面工作:

(1) 阅读程序, 写出运行时输出的结果。

(2) 然后上机运行, 验证结果是否正确。

(3) 分析程序执行过程, 尤其是调用构造函数的过程。

```
#include<iostream>
using namespace std;
class A
{public:
    A() {a=0;b=0;}
    A(int i) {a=i;b=0;}
    A(int i, int j) {a=i;b=j;}
    void display() {cout<<"a="<<a<<" b="<<b;}
private:
    int a;
    int b;
};
```

```
class B : public A
{public:
    B() {c=0;}
    B(int i):A(i) {c=0;}
    B(int i, int j):A(i, j) {c=0;}
    B(int i, int j, int k):A(i, j) {c=k;}
    void display1()
    {display();
      cout<<" c="<<c<<endl;
    }
private:
    int c;
};
```

```
int main()
{ B b1;
  B b2(1);
  B b3(1,3);
  B b4(1,3,5);
  b1.display1();
  b2.display1();
  b3.display1();
  b4.display1();
  return 0;
}
```

## 5.7 答：

(1). 运行结果：

``result

a=0 b=0 c=0

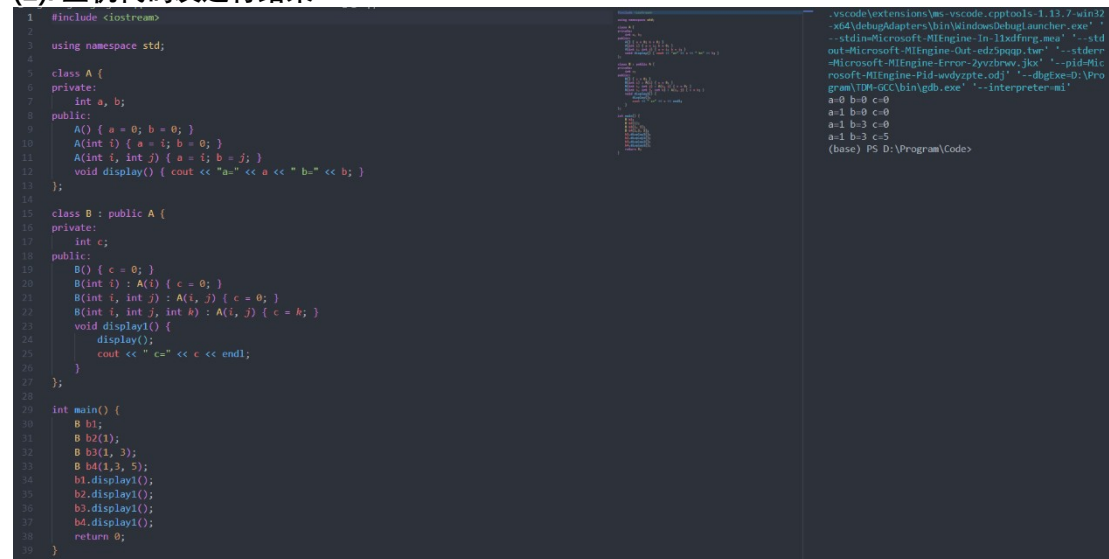
a=1 b=0 c=0

a=1 b=3 c=0

a=1 b=3 c=5

``

(2). 上机代码及运行结果：



```
1 #include <iostream>
2
3 using namespace std;
4
5 class A {
6 private:
7     int a, b;
8 public:
9     A() { a = 0; b = 0; }
10    A(int i) { a = i; b = 0; }
11    A(int i, int j) { a = i; b = j; }
12    void display() { cout << "a=" << a << " b=" << b; }
13 };
14
15 class B : public A {
16 private:
17     int c;
18 public:
19     B() { c = 0; }
20     B(int i) : A(i) { c = 0; }
21     B(int i, int j) : A(i, j) { c = 0; }
22     B(int i, int j, int k) : A(i, j) { c = k; }
23     void display1() {
24         display();
25         cout << " c=" << c << endl;
26     }
27 };
28
29 int main() {
30     B b1;
31     B b2(1);
32     B b3(1, 3);
33     B b4(1, 3, 5);
34     b1.display1();
35     b2.display1();
36     b3.display1();
37     b4.display1();
38     return 0;
39 }
```

Output:

```
a=0 b=0 c=0
a=1 b=0 c=0
a=1 b=3 c=0
a=1 b=3 c=5
(base) PS D:\ProgramCode>
```

(3). 程序分析：

创建对象 b1 时，将会调用构造函数`B()`与`A()`进行构造，a、b、c 都将会被赋值为 0；

创建对象 b2 时，将会调用构造函数`B(int i)`与`A(int i)`进行构造，i 的值将会被赋给成员 a

创建对象 b3 时，将会调用构造函数`B(int i, int j)`与`A(int i, int j)`进行构造，i、j 的值将会分别赋给 a、b

创建对象 b4 时，将会调用构造函数`B(int i, int j, int k)`与`A(int i, int j)`进行构造，i、j、k 的值将会分别赋给 a、b、c

程序最后通过分别调用每个对象的 display1 成员函数输出各成员的值

## 5.8

- (1) 阅读程序, 写出运行时输出的结果。
- (2) 然后上机运行, 验证结果是否正确。
- (3) 分析程序执行过程, 尤其是调用构造函数和析构函数的过程。

```
#include<iostream>
using namespace std;
class A
{public:
    A() {cout<<"constructing A "<<endl;}
    ~A() {cout<<"destructing A "<<endl;}
};

class B : public A
{public:
    B() {cout<<"constructing B "<<endl;}
    ~B() {cout<<"destructing B "<<endl;}
};

class C : public B
{public:
    C() {cout<<"constructing C "<<endl;}
    ~C() {cout<<"destructing C "<<endl;}
};

int main()
{C c1;
    return 0;
}
```

## 5.8 解：

(1). 因此输出结果应该为：

``result

constructing A

constructing B

constructing C

destructing C

destructing B

destructing A

``

(2).

· 上机代码与运行结果：

```
1 #include <iostream>
2
3 using namespace std;
4
5 class A {
6 public:
7     A() { cout << "constructing A" << endl; }
8     ~A() { cout << "destructing A" << endl; }
9 };
10
11 class B : public A {
12 public:
13     B() { cout << "constructing B" << endl; }
14     ~B() { cout << "destructing B" << endl; }
15 };
16
17 class C : public B {
18 public:
19     C() { cout << "constructing C" << endl; }
20     ~C() { cout << "destructing C" << endl; }
21 };
22
23 int main() {
24     C c1;
25     return 0;
26 }
```

版权所有 (C) Microsoft Corporation. 保留所有权利。  
安装最新的 PowerShell，了解新功能和改进！ <https://aka.ms/PSWindows>  
加载个人及系统配置文件用了 1191 毫秒。  
(base) PS D:\Program\Code> conda activate base  
(base) PS D:\Program\Code> & "c:\Users\MOBAVASHI\vscode\extensions\ms-vscode.cpptools-1.13.7-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe" "--stdin=Microsoft-MIEngine-In.jsabvyl0.c15" "--stdout=Microsoft-MIEngine-Out-zwefzppu.0lj" "--stderr=Microsoft-MIEngine-Error-glyxoul.zrb" --pid=Microsoft-MIEngine-Pid-55sep2ah3.q4p" --dbgExe=D:\Program\TDM-GCC\bin\gdb.exe" --interpreter=mi  
constructing A  
constructing B  
constructing C  
destructing C  
destructing B  
destructing A  
(base) PS D:\Program\Code>

· 可见输出结果符合(1)中的预期

(3). 根据派生类构造的规则，在实例化一个派生类对象时首先会根据其派生链从顶往下调用构造函数；在销毁一个对象时会根据其派生链从下往上调用析构函数，因此在构造 C 类对象 c1 时，会先调用派生类最顶端的 A 类的构造函数，然后调用 B 类的构造函数，最后再调用 C 类的构造函数；再销毁 c1 时则是相反的顺序

## 5.10

10. 将 5.8 节中的程序片段加以补充完善, 成为一个完整的程序。在程序中使用继承和组合。在定义 Professor 类对象 prof1 时给出所有数据的初值, 然后修改 prof1 的生日数据, 最后输出 prof1 的全部最新数据。

## 5.10 答:

### · 上机代码及运行结果:

```
1  #include <iostream>
2  #include <string>
3
4  using std::string;
5
6  class Teacher {
7  private:
8      int num;
9      string name;
10     char sex;
11 public:
12     Teacher(int n, string na, char s) {
13         num = n; name = na; sex = s;
14     }
15
16     virtual void show() {
17         std::cout << "num: " << num << std::endl;
18         std::cout << "name: " << name << std::endl;
19         std::cout << "sex: " << sex << std::endl;
20     }
21 };
22
23 class BirthDate {
24 private:
25     int year, month, day;
26 public:
27     BirthDate(int y, int m, int d) {
28         year = y; month = m; day = d;
29     }
30
31     void reset(int ny, int nm, int nd) {
32         year = ny; month = nm; day = nd;
33     }
34
35     friend std::ostream& operator<<(std::ostream& os, const BirthDate& bd) {
36         os << bd.year << "-" << bd.month << "-" << bd.day;
37         return os;
38     }
39 };
```

1\_Programing Language > Cpp > Practice > HomeWork > book > 221210\_10.cpp > Teacher

```
41 class Professor : public Teacher {
42 private:
43     BirthDate birth;
44 public:
45     Professor(int n, string na, char s, BirthDate b) : Teacher(n, na, s), birth(b) {}
46
47     void resetBirth(int ny, int nm, int nd) {
48         birth.reset(ny, nm, nd);
49     }
50
51     void show() {
52         Teacher::show();
53         std::cout << "birth: " << birth << std::endl;
54     }
55 };
56
57 int main() {
58     BirthDate b(1990, 1, 1);
59     Professor prof1(1, "zhang", 'm', b);
60     prof1.show();
61     prof1.resetBirth(1991, 2, 2);
62     prof1.show();
63     return 0;
64 }
```

```
num: 1
name: zhang
sex: m
birth: 1990-1-1
num: 1
name: zhang
sex: m
birth: 1991-2-2
```