第五章

5-1

作用域讨论的是标识符的有效范围，作用域即是一个标识符在程序正文中有效的区域。

1. 函数原型作用域

在函数原型声明时参数的作用范围。

1. 局部作用域
2. 形参的作用域
3. 函数体内声明的变量
4. 类作用域
5. 命名空间作用域

5-2

可见性讨论的是标识符是否可以被引用，程序运行到某一点，能够引用到的标识符，即该处可见的标识符。

一般规则：（1）标识符要声明在前，引用在后

1. 在同一作用域中，不能声明同名的标识符
2. 在没有互相包含关系的不同的作用域中声明的同名标识符，互不影响
3. 如果在两个或多个具有包含关系的作用域中声明了同名标识符，则外层标识符在内层不可见

5-5&5-6

静态数据成员：不属于任何一个对象，其数据是描述类的所有对象共同特征的一个数据项。

静态成员函数：也属于整个类，由同一个类的所有对象共同拥有，可以通过类名或对象名来调用。

5-9

都不是。友元关系是不能传递的，是单向的，是不能被继承的。

5-10

可以。

Private:

Static Int x=0;

第六章

6-6

“\*”：（1）声明指针：数据类型 \*标识符，如：int \*p

1. 获取指针所指向的变量的值：如：cout<<\*p

“&”：（1）声明引用：如：int &r

（2）得到对象的地址：如：pa=&a

6-7

指针类型是专门用来存放内存地址的变量类型。

指针中存储的地址就是另外一个对象的地址，地址中的内容就是另一个对象的内容。

6-11

1.语言形式的差异：指针存储的是地址，而引用把地址的概念隐藏起来了，但在引用运行时的实现机制中，还不得不借助于地址。赋值差异：普通指针可以多次被赋值，也就是说可以多次更改它所指的对象，而引用只能在初始化时指定被引用的对象。

2.引用不能替代指针：

1. 引用只能在初始化时指定被引用对象，若指针所指向的对象要更改或用分支语句确认时，引用不能替代
2. 指针的值可以是空指针，但引用没有空引用
3. 用new动态创建的对象或数组，需要用指针来存储它的地址
4. 以数组形式传递大批量数据时，需要用指针类型接受参数

6-15

Const int \*p1 是指向常量的指针，此时不能通过指针改变所指对象的值，但指针本身可以改变，指向另外的对象。

Int \* const p2 是指针类型的常量，这时指针本身的值不能被改变。

6-18

#include<iostream>

Using namespace std;

Int fn1(){

Int \*p=new int (5);

return \*p;

}

Int main(){

Int a=fn1();

cout<<”the value of a is:”<<a;

return 0;

}

错误：指针指向的内存空间没有被释放。

改正：

#include<iostream>

Using namespace std;

Int fn1(){

Int \*p=new int (5);

return \*p;

}

Int main(){

Int \*a=fn1();

cout<<”the value of a is:”<<\*a;

delete \*a;

return 0;

}

6-19

typedef long(\*LongFunction1)(int);

typedef long(A::\*LongFunction2)(int);