1：float创建变量时是默认先创建double类型之后转换float 所以可在变量后加f 这样可直接生成float类型 如：float a=3.14f

2：goto命令 可以直接跳转到后接名称所在处 如：goto flag 则之后直接跳转到flag后的程序段 类似汇编段的概念

3：int \* const p 指针常量 指针所指向的地址不能改变 const修饰指针 但是所指向的值能够被改变

4：int const \*p 常量指针 指针指向的变量不能改变 但是所指的地址能够改变

5：const int \* const p 两者都不能改变 主要看const修饰的哪个值

6：const 可以加在函数形参中 防止参数被改变 如 void function(const \*p)

7：C++内存分区模型 在执行程序时一般将内存划分为四个区域 （1）代码区：存放函数的二进制代码 ，由操作系统管理，代码区是共享的同时是只读的，形成可多次调用一份文件和防止被贸然修改 （2）全局区：存放全局变量和静态变量以及常量 ，该区域的数据在程序结束后有操作系统释放。 （3）栈区：由编译器自动分配释放，存放函数的参数值，局部变量等 （4）堆区：由程序员分配和释放，若程序员不释放，在程序结束时由操作系统回收。

8：在代码运行前会存在全局区与代码区，全局区的常量包括字符串常量和全局常量 先全局才有常量 局部常量在栈区

9：不要返回局部变量的地址 因为在函数退出后局部变量就被注销了，所以该地址得不到原值

10：new 利用new可以开辟一块地址到堆区，返回一个地址，如：int \*p =new int（10） 因创建在堆空间所以不会在函数退出时被释放，可以被传出使用，类似全局变量 但需要自己手动释放，释放采用delete操作符

11：new创建数组 int \* a=new int [n] ，但在释放这个数组时就需要 delete a[ ] 来释放 加一个[ ] ，创建数组后的使用方法就是和基本数组一样，如 a[2]

12：引用 ：给变量取一个别名，还是代表原变量 。用法 ：int &b=a 就是&的使用 此时b和a都可以操纵定义是的变量值 a=10。注意：引用要初始化，也就是在定义时就要指向一个变量名，切不可以再更改指向另外一个变量名。

13：引用的用处可以在函数传入值时使用 就可以像地址一样改变原值并带出 如：void function(&a,&b)，然后在使用此函数时是 function(a,b),这就是将a，b引用 之后在函数内的改变会改变外在的a，b值。可以将引用视为和地址对应的数据类型。所以函数也可以返回引用类型 ，注意不要返回函数内部局部变量，这样会被程序释放，造成乱码。同时引用类型的函数可以作为左值被赋值，就是相当于返回一个变量名。

实例：#include <iostream>

using namespace std;

int& function()

{

Static int a = 10;

return a;

}

int main()

{

//引用的返回

int& ref = function();

cout << ref << endl;

//引用的返回为左值

function() = 20;

cout << ref << endl;

system("pause");

return 0;

}

14：引用的本质：就是指针，发现为引用就被转换为 &ref=a 得到 int \*const ref=&a ref=20就得到 \*ref=20 所以为指针常量 所指的地址不能改变但是能够改变指向的值

15：int &ref =10该方法不可行 因为没有指向一块变量地址 但是const int &ref =10可行，这相当于编译器加了一个步骤 int temp=10 const int &ref =temp 相当于创建了一个临时变量temp 就是一块可用地址了。

16：C++中函数的形参中是可以有默认值的，如 int function(int a=10),此时可以不用传入参数，因为有默认值，也可传入使用。注意：（1）若在某个参数开始设有默认值，则之后的参数也要有默认值，否则报错，不能出现如int function(int a,int b=10,int c)这种情况。（2）函数声明和函数定义中的设置默认参数只能存在一个，不能两者同时设置默认参数。

17：占位参数 如int function(int ) 占一个参数位置，同时调用时需要传入一个值。占位参数可以被默认值 如int function(int =10)

18：函数重载：作用是让函数名可以相同，提高复用性。条件：（1）同一个作用域下（2）函数名称相同（3）函数参数类型不同或者个数不同或者顺序不同。其实就是如果参数不同就可以确定不同的函数，及时函数名相同。也就是函数参数是是函数定义中的重要组成部分，可以作为辨别标志。但是当返回类型不同是不可以认为是两个函数的。

19：类的数据类型 用 class来定义 在类中数据是属性，函数是行为 同时由权限之分。实例 class circle

{

public:

int r;

double zhouchang()

{

return r;

}

};

也通过类的名称作为数据类型定义对象 如 circle a；之后调用它的属性和行为通过“.”来使用访问。就是另类的结构体但有区别，。

20：类的权限有三种：（1）公共权限 public 成员在类内可以访问，类外也可以访问（2）保护权限：protected 成员在类内可以访问，类外不能访问 （3）私有权限：private 成员类内可以访问，类外不能访问。保护权限子类可以访问，但私有权限子类不能访问。类内与类外的区别在于：在类的定义里面和实例化后对象的访问内部属性和行为，分别为类内和类外。

21：C++中struct结构可以定义函数，和class类相似，只是在结构体中的都默认权限为公共权限public 。在class类中的默认权限是私有权限private 。在C++中struct定义一个结构体后，创建结构体可以省略struct这个词了。只用结构体类型名 和类一样。

22：构造函数与析构函数进行初始化。



23：构造函数可以重载，加上一个参数表示，同时在创建类对象时加上括号参数，即可选择有参构造函数进行初始化。拷贝构造函数：参数变为一个对象的引用，此时相当于复制那个对象。调用构造函数时不要加括号。其他使用方法还包括另外两种，显示法和隐式法

24：C++中构造函数的调用时机通常三种情况：（1）使用一个已经创建完毕的对象来初始化一个对象（2）值传递的方式给函数参数传值（3）以值得方式返回局部对象

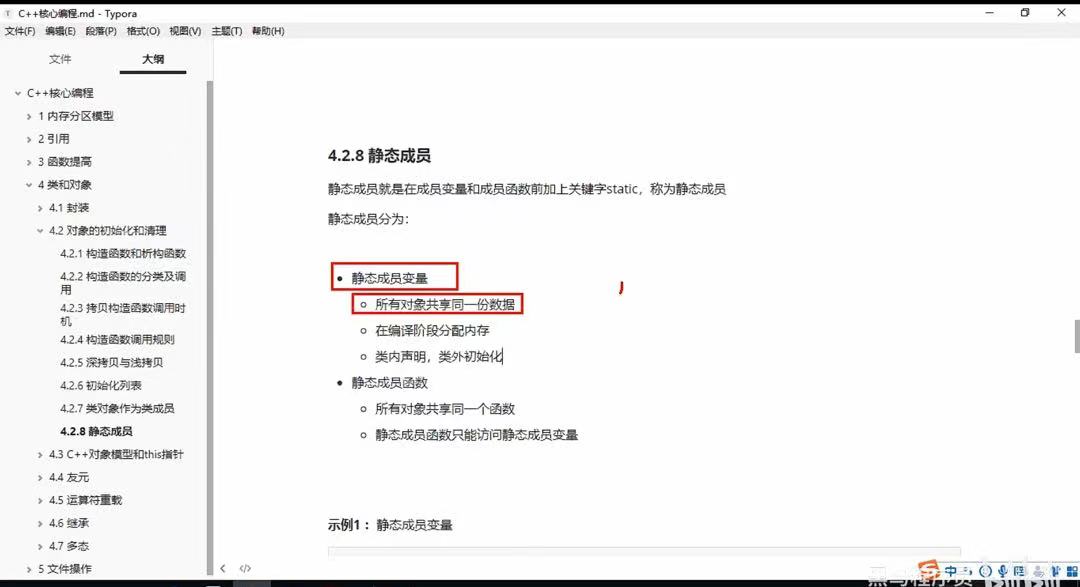
25：C++构造函数调用规则：如果用户自定义有参构造函数，C++不再提供默认无参构造，但会提供默认拷贝构造函数，如果用户定义拷贝构造函数，C++不会提供其他构造函数。一般默认情况下，C++默认三种构造函数都默认创建。

26：深浅拷贝：浅拷贝，简单得复制拷贝得操作。深拷贝，在堆区重新申请空间，进行拷贝操作。深拷贝是为了防止在堆区创建变量时重复释放变量空间。如果有属性在堆区开辟得，一定要自己提供拷贝构造函数，防止浅拷贝带来得问题。

27：类中的初始化列表：在构造函数后接“：属性一，属性二” 可以在构建对象时直接初始化，也可进行传值，在构造函数中传参，后续属性赋值为参数即可。

28：类中可存在其他类的对象，此时是先构造其他类的对象，在构造本身的对象，析构的顺序则是相反。

29：类内声明，类外初始化，在外初始化时要加上类名限制，防止被认为是全局变量。但权限不是被外共享时访问不到在外。



30：在一个类中，只有非静态变量在对象上，其他的如函数，静态变量，静态函数都不在对象上，且空对象内存为1字节。

31：当对象中函数形参和实参名称相同时，为区分两者，可以用this这个指针指向本对象，这就可以通过this->属性来确认为本对象的属性。This为自带指针，会自动指向当前对象，并且可以通过这个指针返回这个对象。\*this。

32：在成员函数后面加上const可以将函数修饰成常量，修饰的是this的指向，让this的指向也不能更改。要在常量函数中更改属性，需要在属性前加上一个mutable，此时在常对象中也可更改。

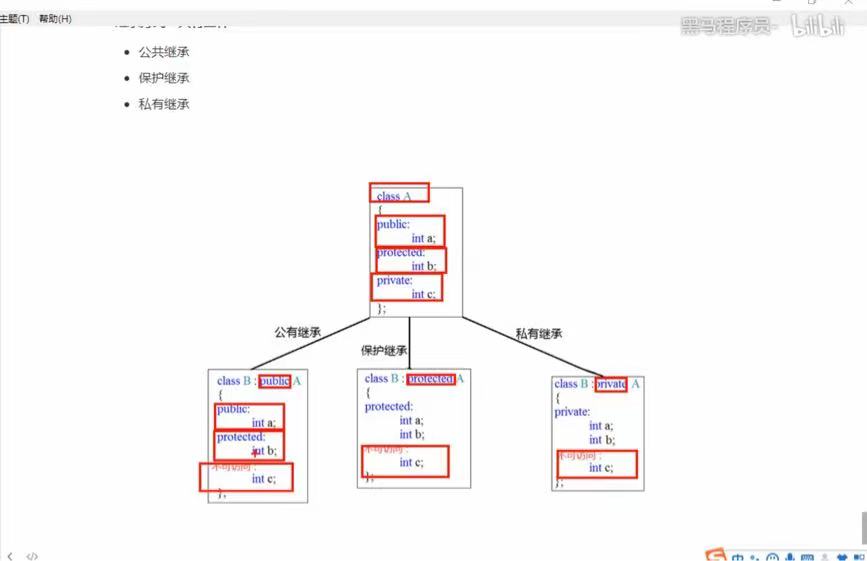


33：友元，friend 目的是让一个函数或者类，访问另一个类中的私有变量。三种实现，1，全局函数做友元。在对象定义中加函数声明，同时前面加friend 2，类做友元，实现方式参考全局函数 。3，成员函数做友元，

34：运算符重载：对已有的运算符重新定义，赋予另外一种功能，以适应不同的数据类型。就是对于内置的数据类型能够计算，但对于自定义的可能造成误解。实现方式1：通过成员函数重载，2：通过全局函数重载。 加关键字operate 后接想要重载的运算符。注意：内置的数据类型运算符是不能发生重载的。不要滥用重载。

35：能够对函数重载

36：继承，class 子类 ：继承方式 父类 继承方式包括1，公共继承 2，保护继承 3，私有继承



37：在父类的所有非静态成员属性都会被子类继承下去，父类中私有的属性只是访问不到，但却是被继承了。

38：继承中构造的顺序，先构造父类，再子类，析构相反。

39：当子类与父类出现同名的成员时，访问字类的同名成员时，直接访问。访问父类则需要加作用域。也就是出现相同的，子类的会覆盖父类的，要访问加作用域。

40：继承中同名的静态成员在子类上如何访问？处理方式和上述一样，加上作用域即可。也可通过类名来访问。

41：一个子类可以继承多个类。语法为：class 子类 ：继承方式 父类1，继承方式 父类2

用逗号隔开即可。有同名时加上作用域区分即可。

42：菱形继承：两个子类继承一个父类，又有一个子类继承了上述的两个子类，形状构成菱形，称为菱形继承，钻石继承。但是这样会导致一个父类的数据被重复复制了两份，浪费空间。解决方法：在继承前加上关键字virtual变为虚继承，语法 class 子类 ：virtual 继承方式 父类。这样就可以将之只复制成一份。底层为复制数据变为成为指向复制出的父类数据的指针。

43：多态：1：静态多态，函数重载和运算符重载，复用函数名。2：动态多态：派生类和虚函数实现运行时多态。两者的区别：静态多态的函数地址早绑定，编译阶段确定函数地址，动态多态的函数地址晚绑定，运行阶段确定函数地址。动态的创建方法就是加上virtual关键字。动态多态创建条件：1，要有继承存在。2，子类要重写父类中的虚函数。要使用虚函数则是用父类的指针或引用作为形参，子类对像作为实参。

44：动态虚函数的实质是子类虚函数重写时会覆盖原来指向父类虚函数的指针指向，改为子类虚函数地址。

45：纯虚函数与抽象类。纯虚函数语法：virtual 返回类型 函数名 （参数列表）=0 .当有纯虚函数存在时，该类被称为抽象类。注意：抽象类无法创建对象，子类继承时必须重写纯虚函数，否则也是抽象类。

46：虚析构与纯虚析构：可以避免子类创建在堆区的函数时，父类析构无法将其访问。如果父类析构函数为纯虚析构，还是要在外将其定义，以防止父类释放空间漏过。

47：文件操作需要包含头文件<fstream> 文件打开方式可以配合使用，利用 | 操作符，例如用二进制方式写文件 ios::binary | ios::out 。用<<向文件中输入内容





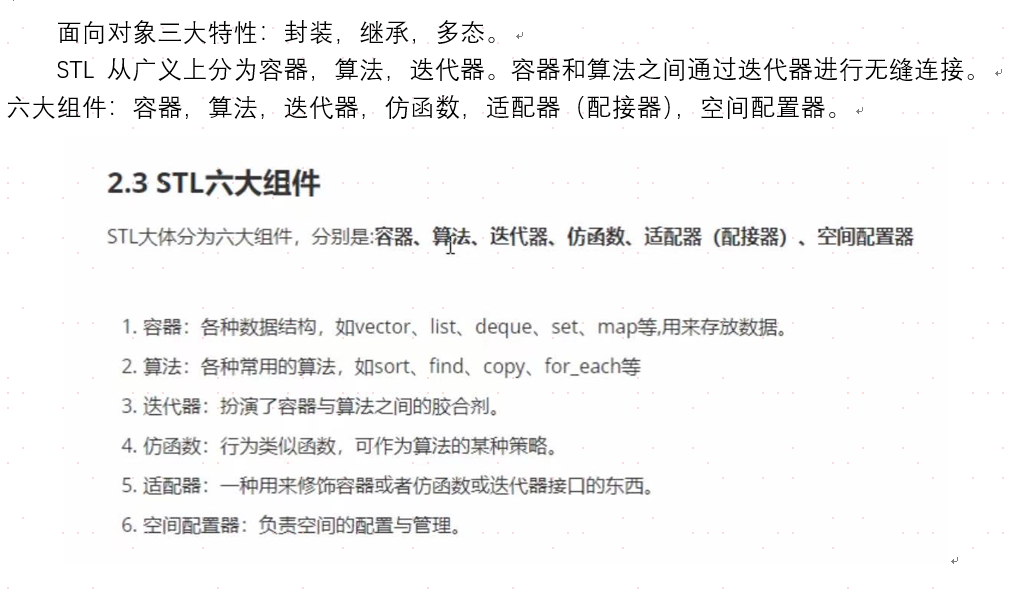
48：读取文件则是和上相反，主要是读数据，用>>将数据读出。如：fs>>buff 可以将数据读出到buff中。或者使用函数 getline() 参数为目标存储地址，个数 。第三种方式：get（）函数读取一个值。用 is\_open函数判断是否打开文件成功。

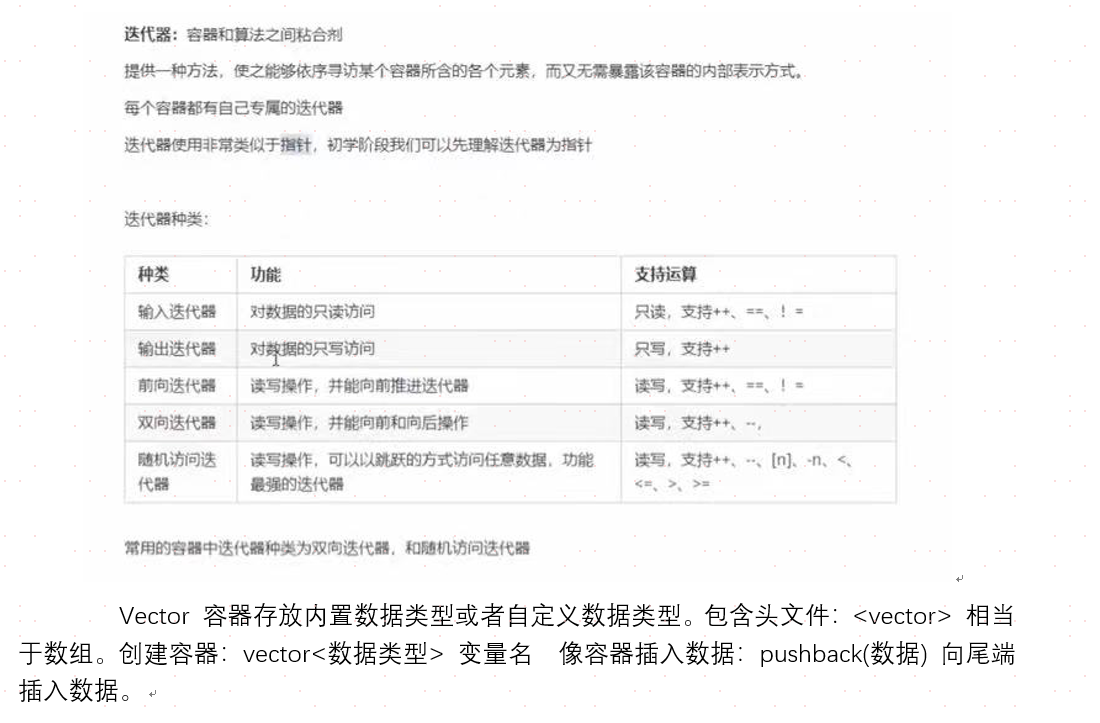
49：二进制输入函数 write,参数为地址和大小。输出函数则是read函数。参数和上一致。

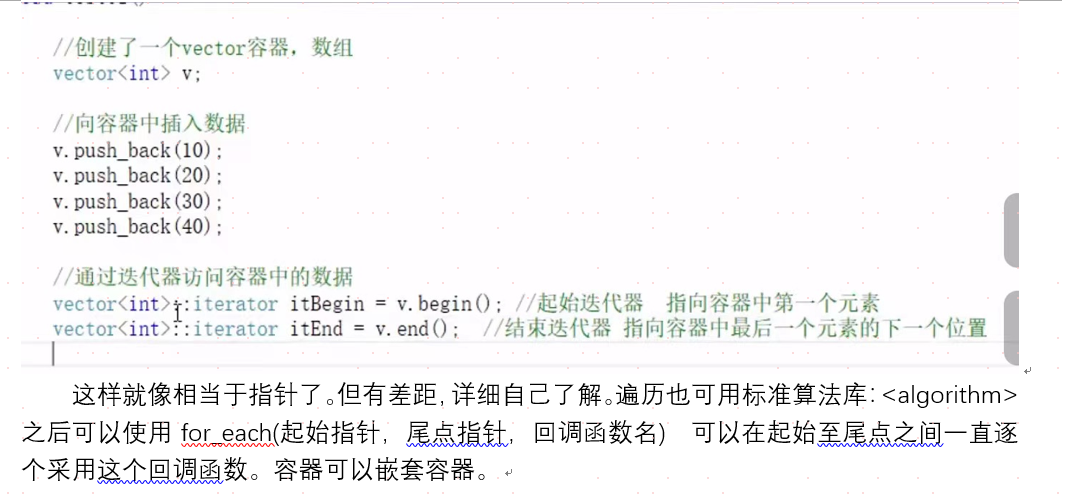
50：C++提供泛型编程，利用模板实现，两种模板机制，函数模板和类模板。1：函数模板的作用：建立一个通用函数，其函数返回值类型和形参类型可以不指定具体类型，由一个虚拟的类型来代表。用关键字：template 表示。之后就可以将之作为一种通用类型使用，也就是一种数据类型为T，在传参时会自动转换。也可在使用时加<数据类型>告诉函数后面是什么数据类型，不用编译器自己判断。Typename可以替换成class。 注意事项：自动推导数据类型需要推导出的类型必须为一致的才能使用模板。通用数据类型必须能够被确定类型。

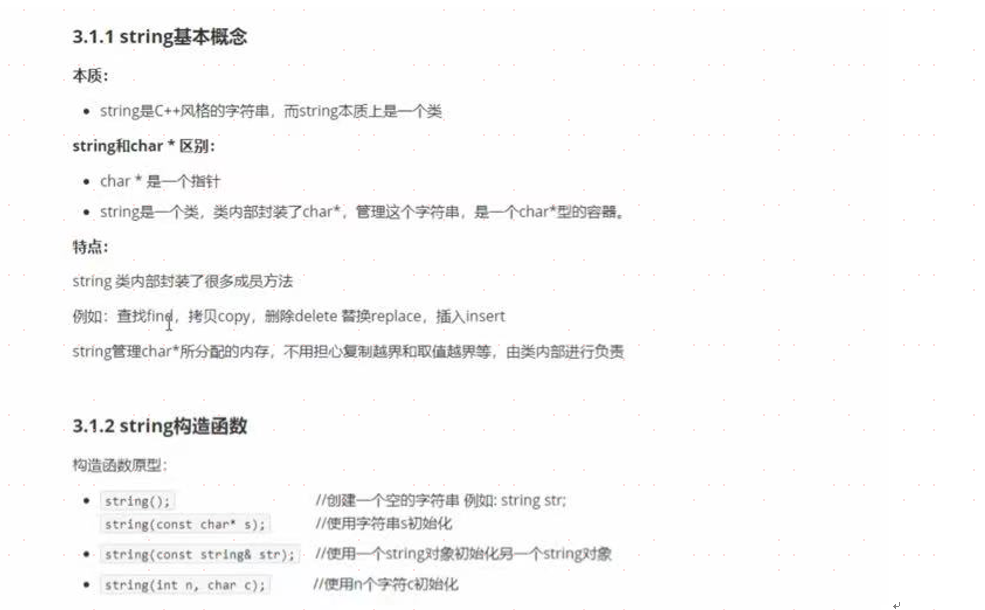


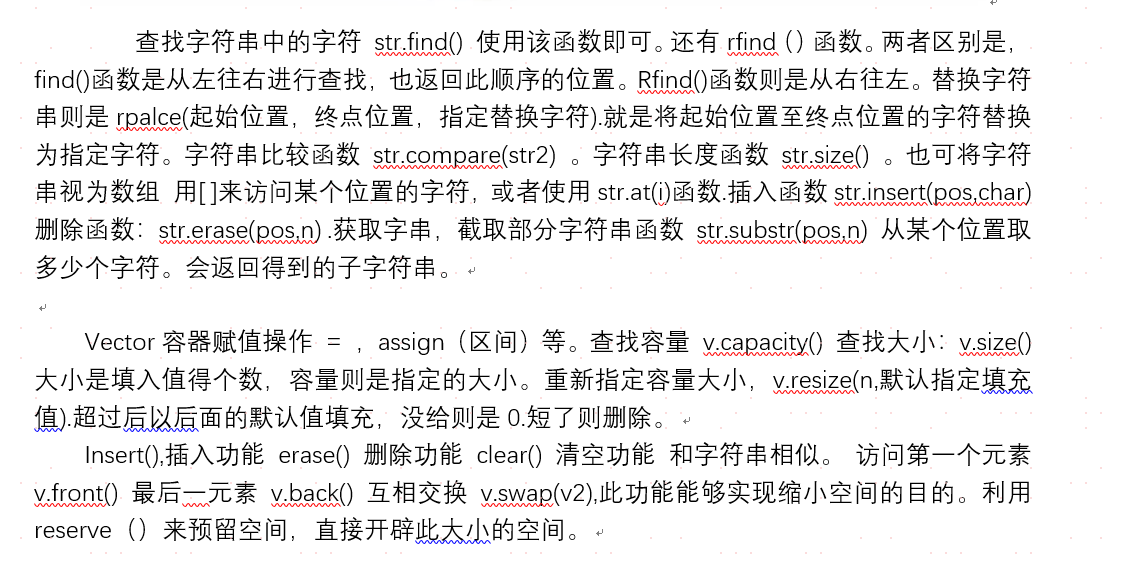
51：普通函数和函数模板的区别：1，普通函数调用·可以发生隐式类型转换 2，函数模板如果采用自动类型转换的方式则不能采用隐式转换，但用显式指定类型，则可以发生隐式类型转换。 隐式类型转换是指传入一个不是一样类型的参数，自动进行转换成形参的类型。

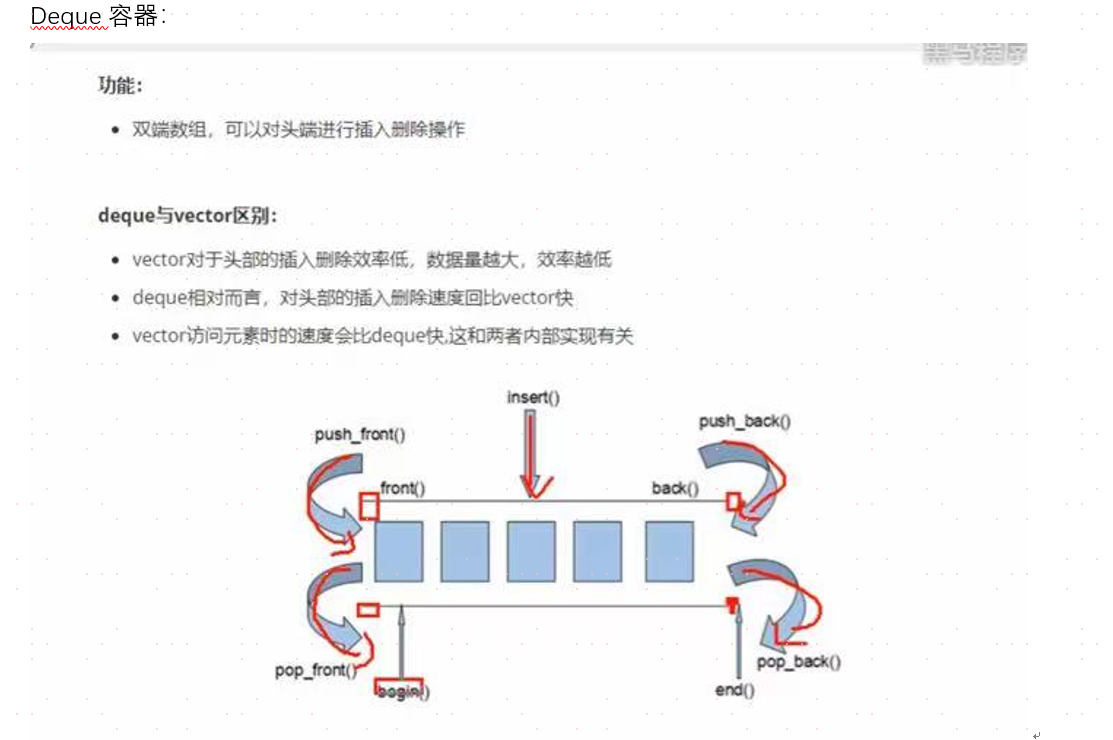


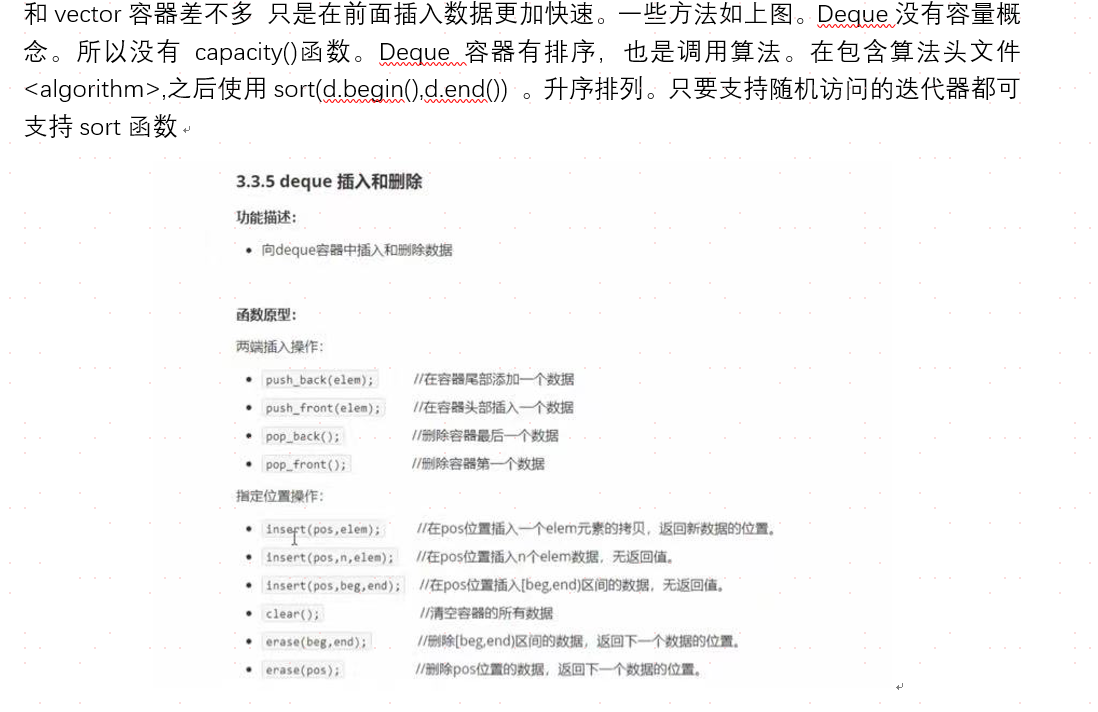




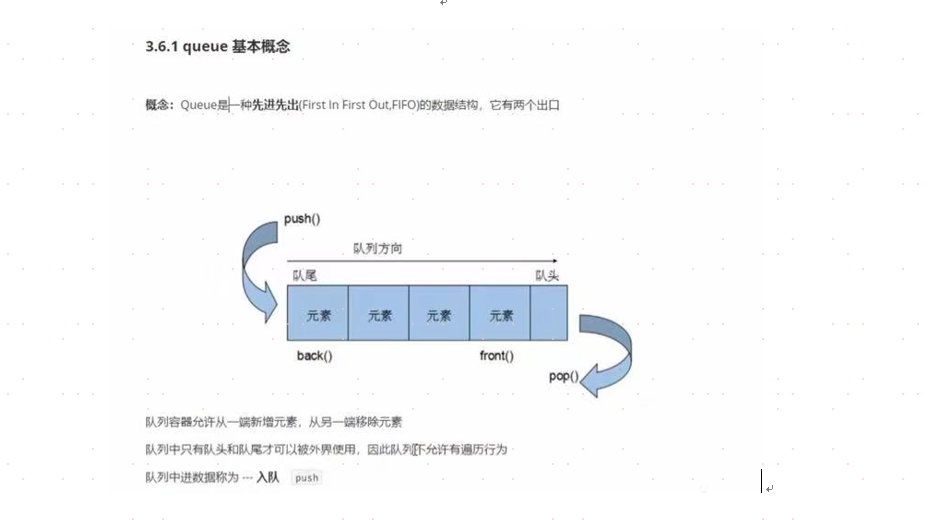




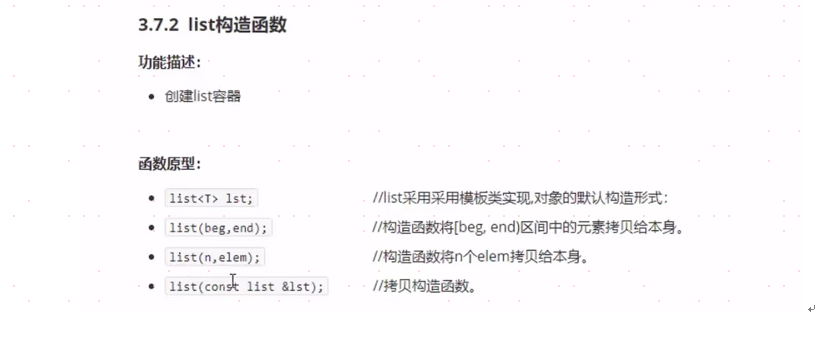


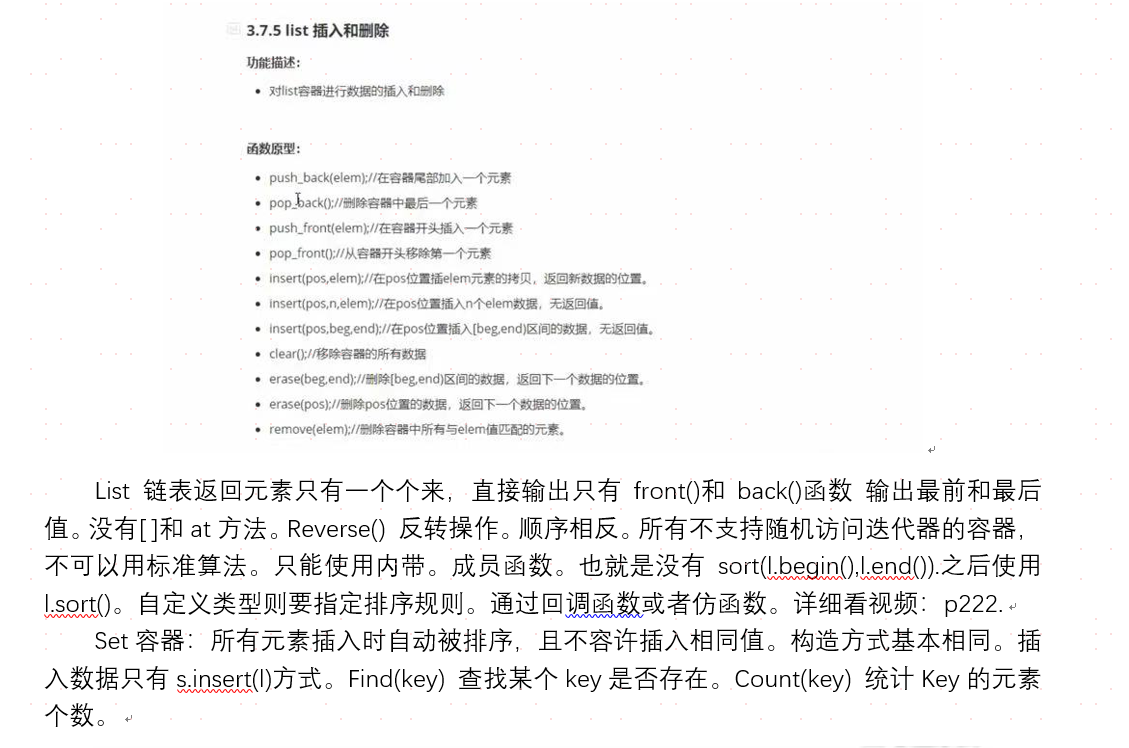


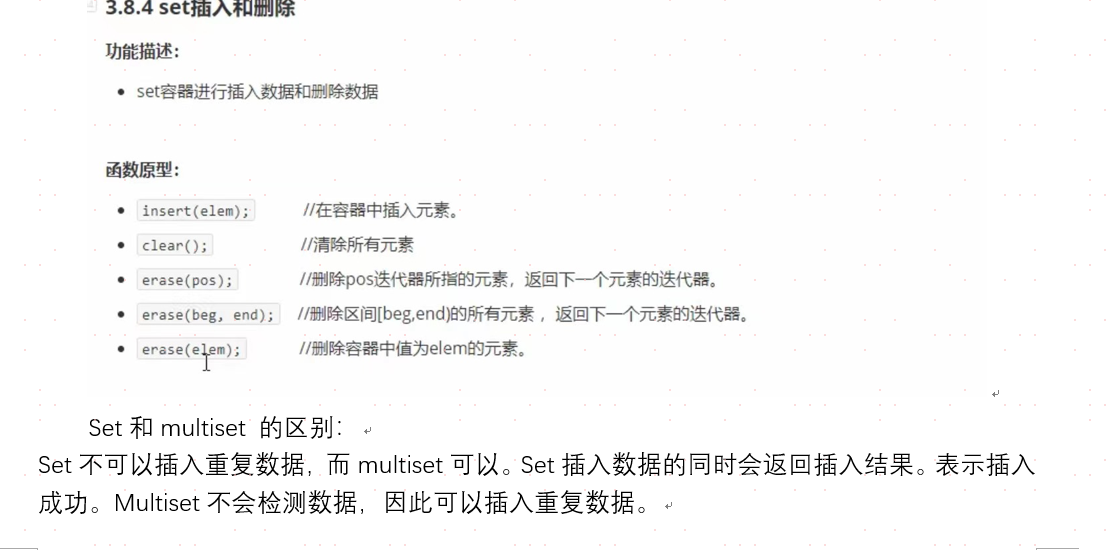




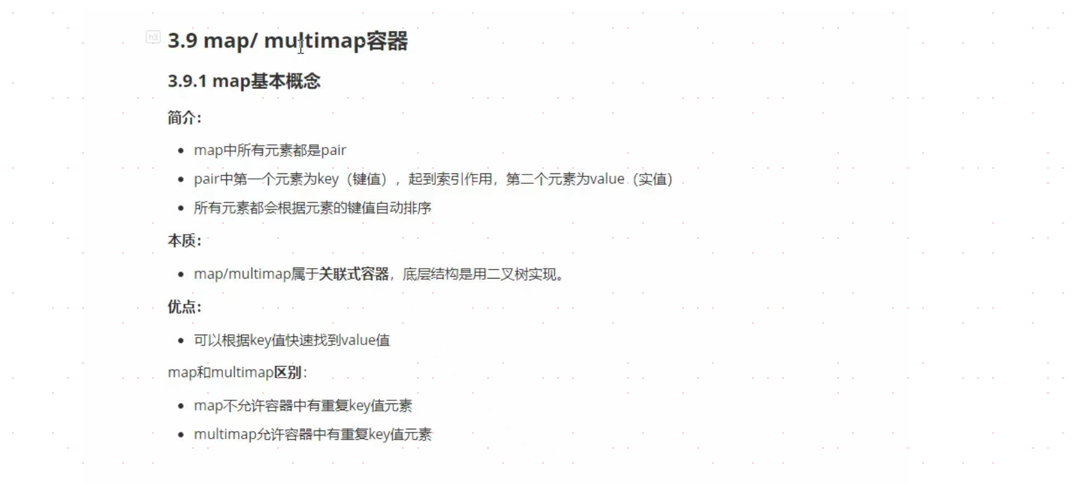


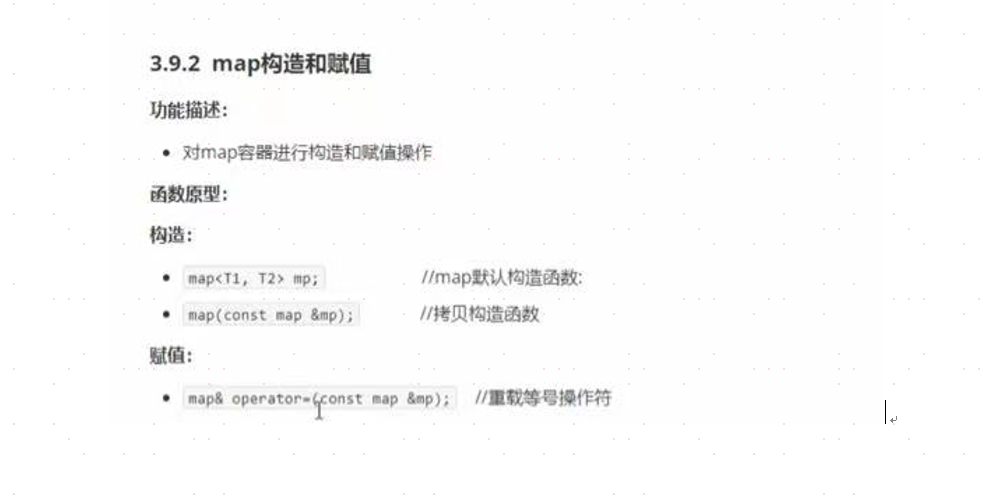


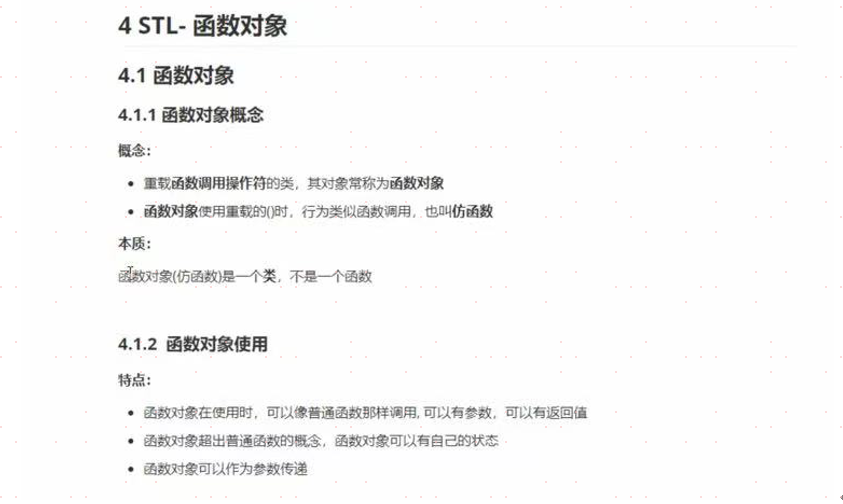


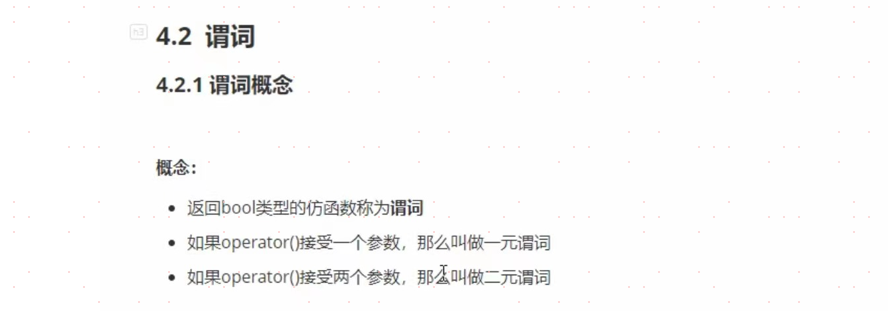




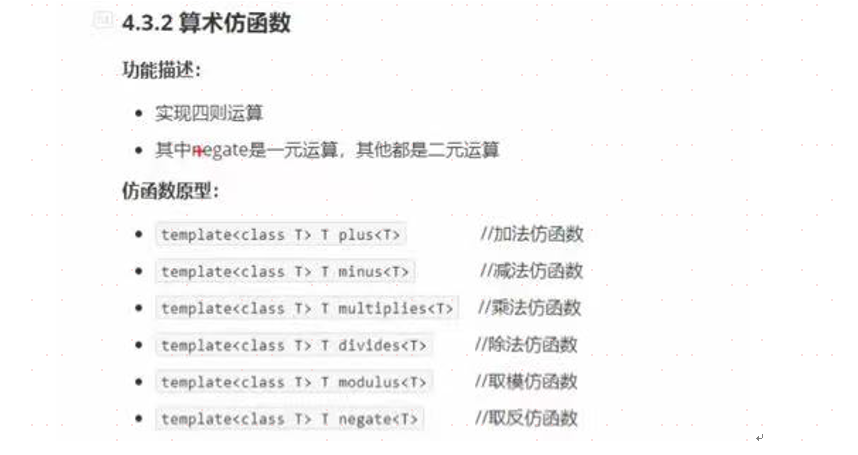




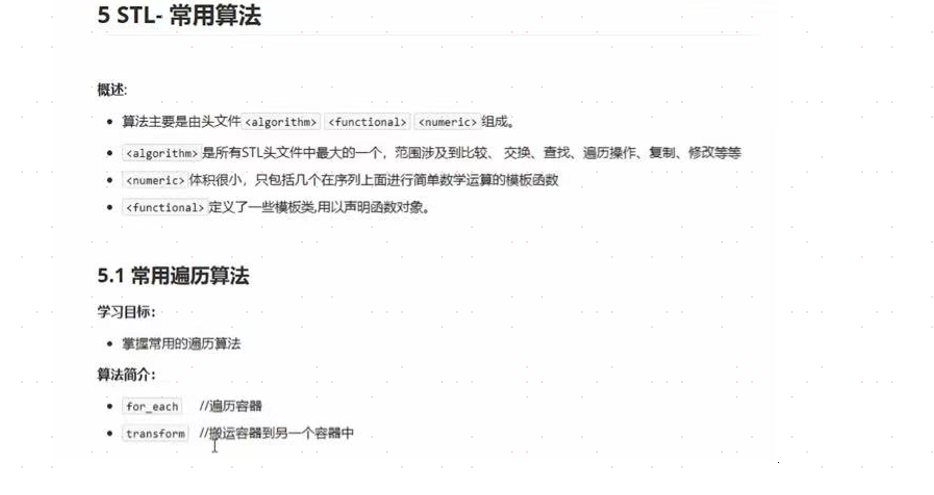


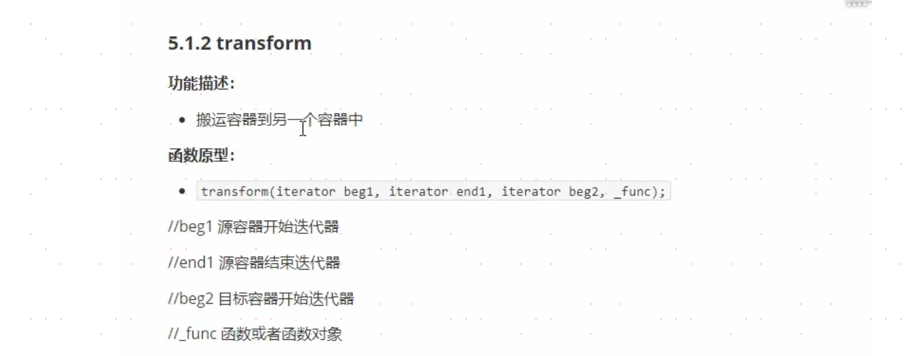


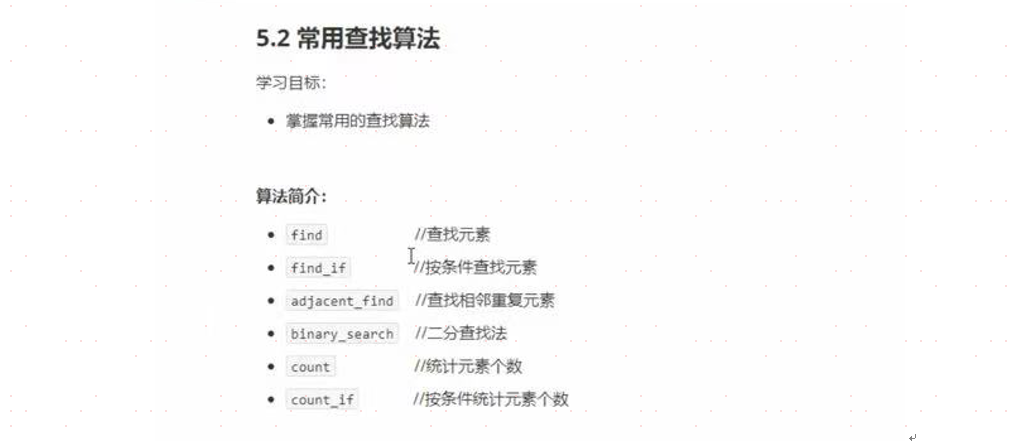


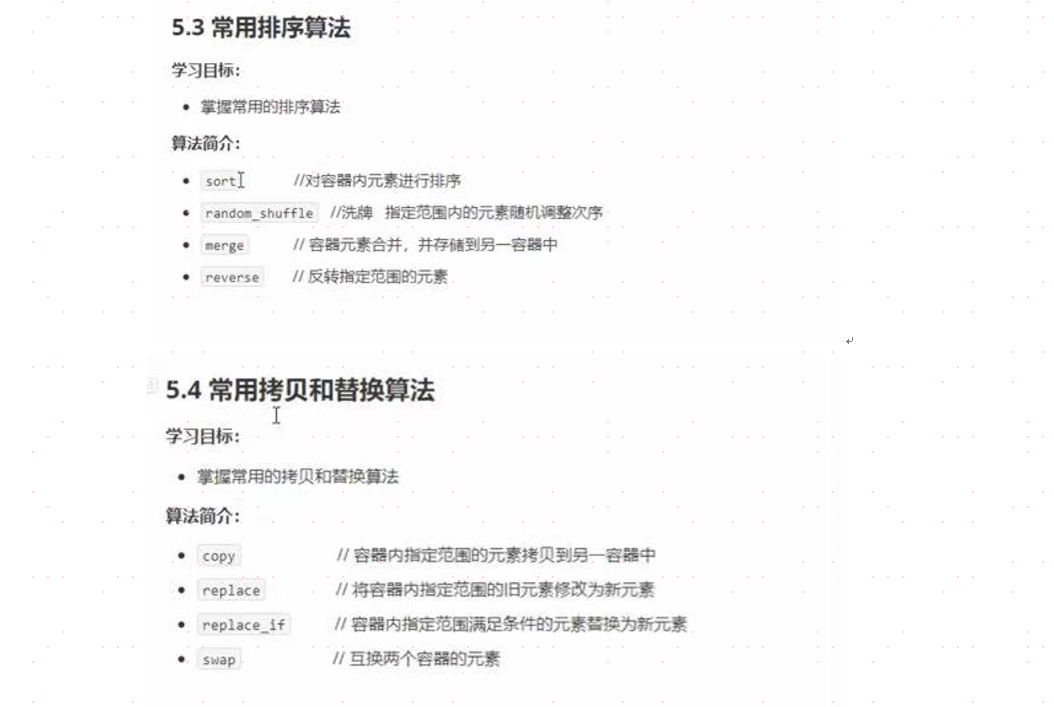


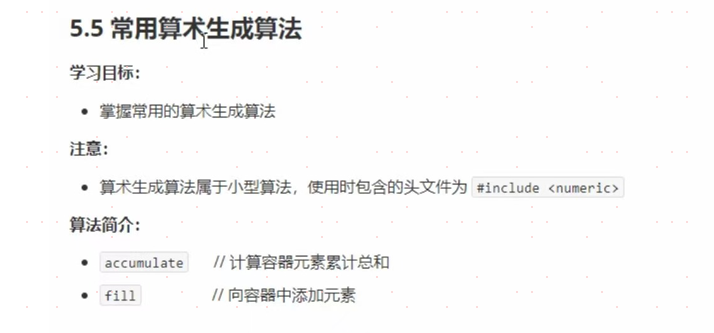


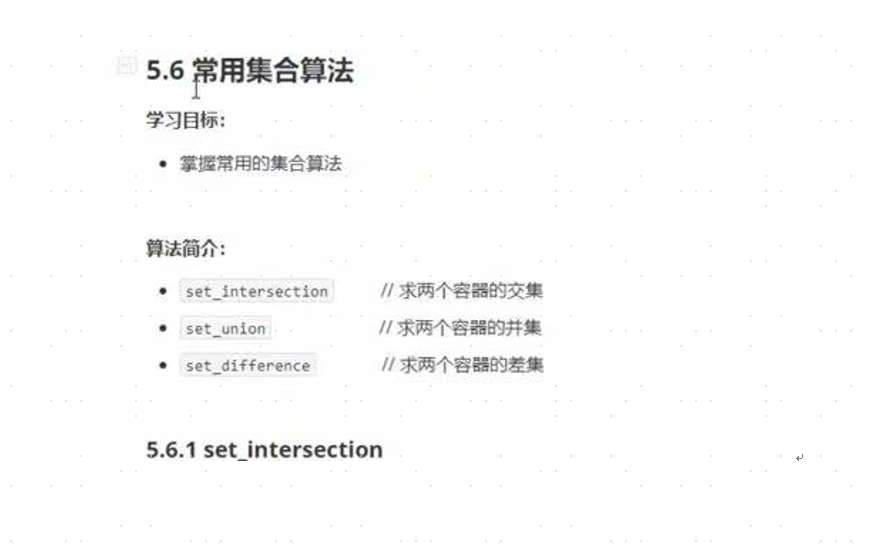




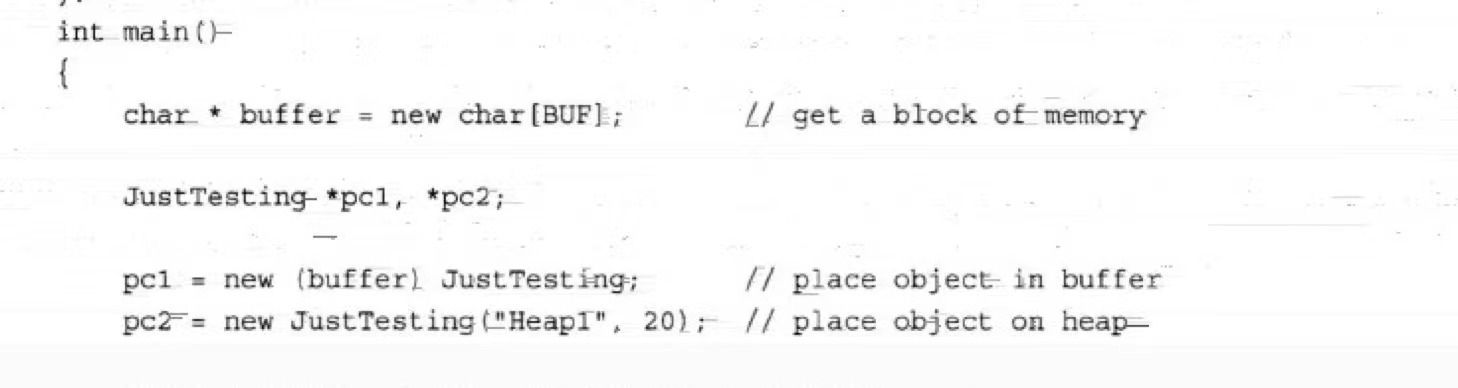








52. new运算符可以在后加上定位地址，实现在指定地址分配内存。更具体的应该是在指定位置创建对象，不包括分配内存，只有指定地址的功能，分配内存由new运算符实现。也是要先将定位new的对象删除后才释放内存空间。如下面的pc1，为指定位置的定位new运算符用法。详见c++ prime plus p458



53. 在类中创建常量的方法：类声明中使用枚举，或者静态变量。不是创建如const常量，因为这样只会在对象创建时才使用，而前两种方法可以在该类的所有对象中共享。详见c++ prime plus p372

54. 构造函数中可以使用较为特殊的初始化方法。叫做成员初始化列表，成员初始化列表由逗号分隔的初始化列表组成，前面带逗号，它位于参数列表的右括号后面，函数体左括号之前。Const类成员与被声明为引用的类成员必须使用此方法。因为必须在构造函数运行前先创建常量（常量不能被赋值，所以在函数内不能被赋值，只能采用该方法） C++ 11中可以使用常规方法赋值。如： 详见 c++ prime plus p464



55. 多态 虚函数，动态实现。相较与普通指针或引用，将根据指针或引用的类型来确定调用的函数，而多态虚函数后，将根据实际对象类型来调用对应的函数。实现一种基类指针实现多个方法函数。普通指针下，只能调用对应类型的。基类指针只能调用基类函数，尽管它指向子类。