*类与对象*

*C++基础预备知识*

*输入与输出*

*数组与向量类*

*If、if-else、switch*

*While、do-while、for*

*算术运算、逻辑运算*

*函数与递归*

*整形、浮点型、引用、指针、内置数组、数据储存类、其他数据类型*



大一上C++复习总结

Z、



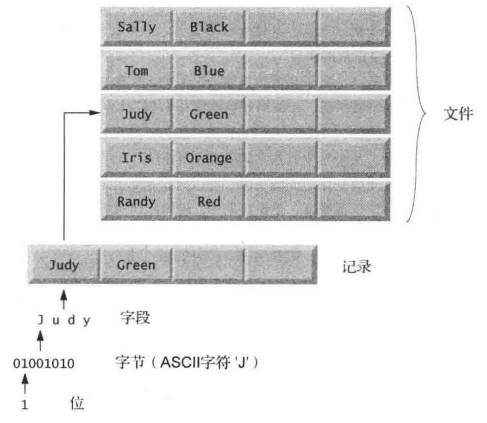
*1.计算机的组成：输入设备、输出设备、内存、外存（硬盘）、CPU、ALU（计算与逻辑单元）*



C++预备知识



摩尔定律：18个月左右电子产品性能翻倍

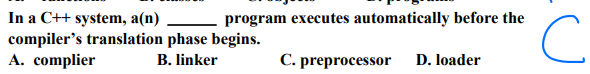


*2.数据的层次结构：位->字符->字段->记录->文件->数据库*

*Bit->character->field->record->file->database*

*3.机器语言(machine language)->汇编语言(assembly language)->高级语言(high-level language)*

*目标代码（object code）->源代码(source code)*

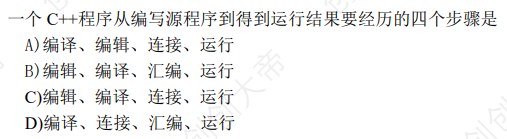


*#include<iostream>中的#*

*就是为了提醒编译器需要预编译*

*< >的是查找系统的目录*

*而类似于#include”GradeBook.h”的自写头文件就是在当前工作目录中去找*



*第一步编辑排除AD，然后高级语言不是汇编排除B，选择C*

*编辑：程序员编写代码*

*预编译：将头文件进行转换，比如#include<iostream>在预编译中被处理*

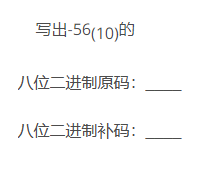
*编译：将源代码翻译成目标代码*

*链接：将库与目标代码链接*

*载入：存入磁盘*

*执行：执行文件*

*4.C++程序开发过程：编辑(edit)->预编译(preprocess)->编译(compile)->链接(link)->载入(load)->执行(execute)*



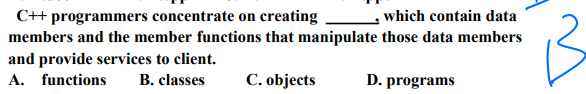
56=2^5+2^4+2^3

源码10111000

反码11000111

补码11001000

*6.C++是面向对象的高级语言，而C语言是面向过程的，C++是C的升级版*



*Unsigned short:*

*0-65535(2^16-1)*

*源码为：*

*1111 1111 1111 1111*

*转换为short后*

*（-32768-32767）*

*第一个是符号位负号*

*反码是：*

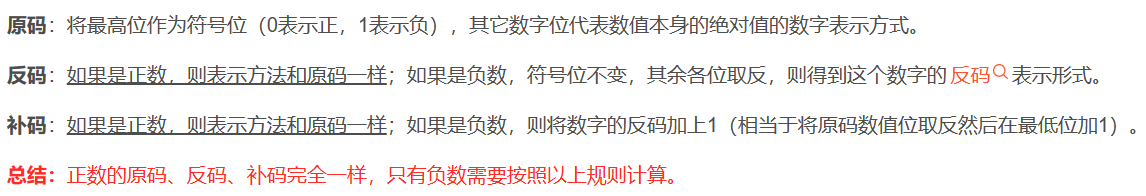
*1000 0000 0000 0000*

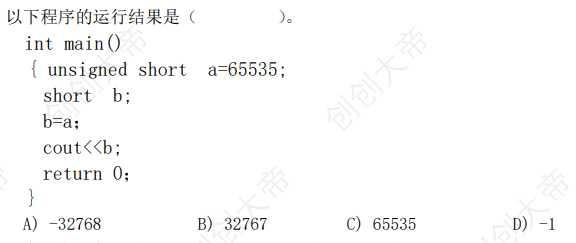
*补码是：*

*1000 0000 0000 0001*

*输出结果-1*

*5.源码、反码、补码*







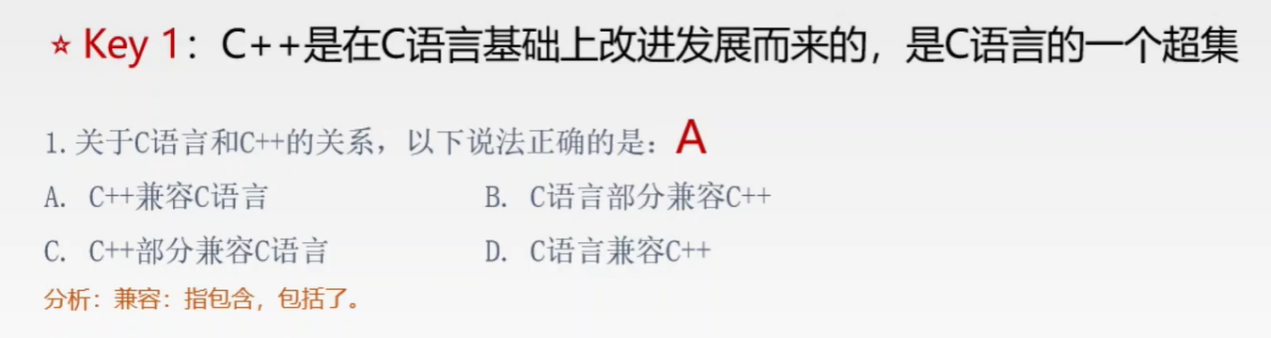
*C++有加号*

*Define不是关键词*

*define才是关键词*

*cout是标准库中的对象*

*不是C++自带关键词*



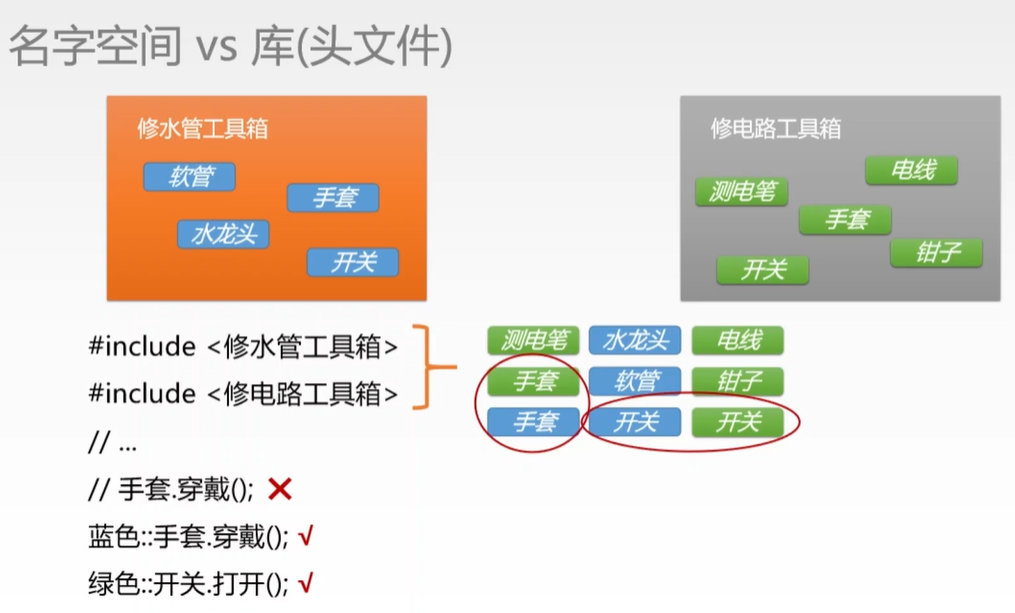
*8.标识符(identifier)变量名称*

*只能含有下划线、字母、数字*

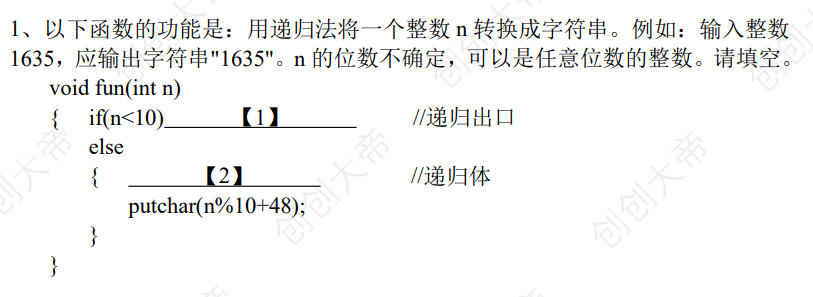
*开头是下划线或者字母*

*不可以使用关键词*

*7.名字空间*



*Using std::cout; 或者using namespace std;*



*布尔型、字符型、字符数组、字符串型、整型、浮点型、无符号、size\_t、引用、指针、数组、储存类型*



数据类型和储存类型





*2.字符型 1字节*

*Char用来储存字符，计算机一般用ASCII码*

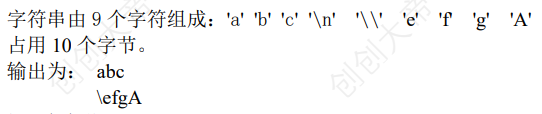
*要记的ASCII：0->48; A->65; a->97;*

*要记的转义字符：\n->newline; \t->tab;*

*\\->\; \101(三位数)（八进制）->A*

*\X41(X开头)(十六进制)->A*





*Putcahr(n+48);*

*Fun(n/10);*

*1.布尔型 1字节*

*Bool有两个值true(1) false(0) , 非0即true无论正负*

*3.字符数组 字符数+1 个字节*

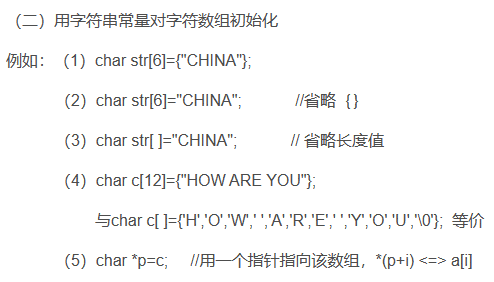
*由字符构成，并且最后一位必须是\0;*

*Char ch [6] = ”apple” 长度为6只能储存长度小于等于5的字符串*

*Ch[0]=’a’ ch[1]=’p’ …… ch[5]=’\0’*

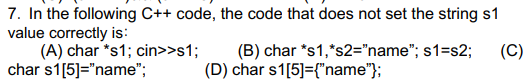
*输出字符数组名至’\0’停止，如果没有找到，则会继续输出一些乱码*

*char c[10]={'c',' ', 'p','r','o','g','r','a','m','\0'}; 依次赋值*

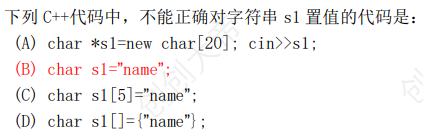


*Const char\*p=”hello” 左为常量指针，右为常量*

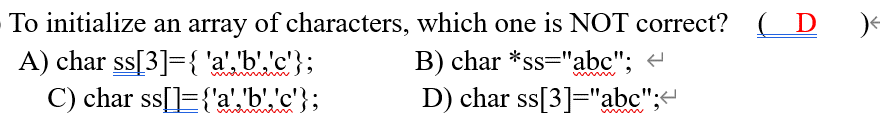
*Char\*p=c 动态的指针指向数组c的首地址，不可以直接等于”hello”,类型不同*



*A的问题是不可以使用野指针，B的问题是左边没有加const*



*A的new返回动态数组的首地址给指针s1；B左边字符右边字符串*



*虽然其他也有不对，但是D把const4给const3是报错，*

*5.整型 char 1字节 、 short 2 字节、 int 4 字节、 long 4 字节、 long long 8 字节*

*Short -2^15~2^15-1 unsigned short 0~2^16-1*

*整型除以整型结果是整型 4/8=0 截取*

*4.字符串 string类*

*String s1; 默认初始化*

*String s2(“hello”); 含参初始化*

*String s3(s2) string s3=”hello” 拷贝构造*

*S1.size() 返回字符串有效长度 64位操作系统string40字节*

*S1.clear() 清除有效字符 s1.empty() 检查有无有效字符*

*S1.resize(n) 变成长的为n的字符串 s1.resize(n, ‘a’) 多的用a填补*

*S1.push\_back(‘a’) 在最后插入一个a*

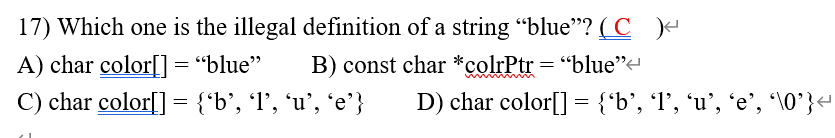
*S1.append(“bcd”) 在最后插入bcd*

*S1+=’e’ 在最后加一个e s1.+=”fgh” 在最后加fgh*

*S1.erase(n,m) 删除n-m(不含m)的字符*

*S1.substr(pos,len) pos为起始位置，len为长度 复制这一段*

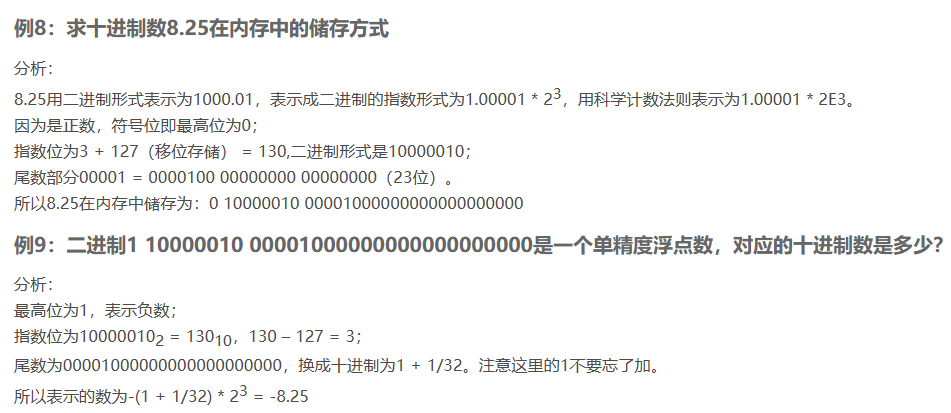
*S1.find(“a”)找a(字符或者字符串)的起始位置 s1.find(‘a’,pos) pos起始位置*



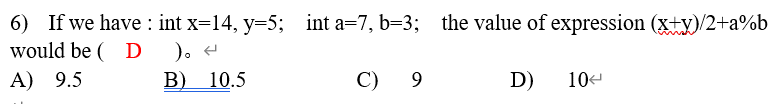
*缺少\0*

*6.浮点型 float 4 字节、double 8 字节 四舍五入*





 0\*50.0得到的是double的0，所以是0.0，因为0对应的Int，只不过cout输出的时候会省略多余的0



19/2=9 7%3=1 9+1=10



*较低级的数据类型会自动隐式转换成高级的数据类型*

*6/5=1 6/2.5=2.4 1+2.4+2.5= 5.9*

*9.引用 &*

*引用就是别名。比如有个同学叫张磊，给他取个外号叫张三石，这就是别名。*

*Int a =10; int &b = a; b相当于a 他们的地址是相同的*

*引用必须初始化*

*引用传递可以真正改变实参，而值传递不行*

*引用可以改变值，但是不能改变指向，类似int \*const p*

*所以引用可以const int &b 不可以改变值，叫做常引用*

*函数不可以返回函数中的局部变量或者对象的引用和指针*

*因为局部的存放于栈区，在函数执行完就会自动销毁，地址消失了不存在引用*

*但是静态变量可以使用引用和指针返回，因为它的生命周期是全局*

*8.size\_t*

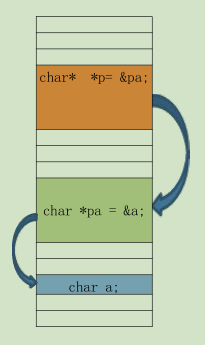
*size\_t 类型表示C中任何对象所能达到的最大长度，它是无符号整数。*

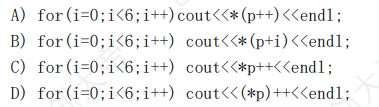
*只表示非负整数， 0~INTMAX*

*size\_t 在数组下标和内存管理函数之类的地方广泛使用，比如for循环*

*7.unsigned*

*无符号， 只有非负数， 比原来数据的正数范围扩大一倍*





指针变量可以自增 p++优先级高于\*

（\*p）已经解引用了

*10.指针\**

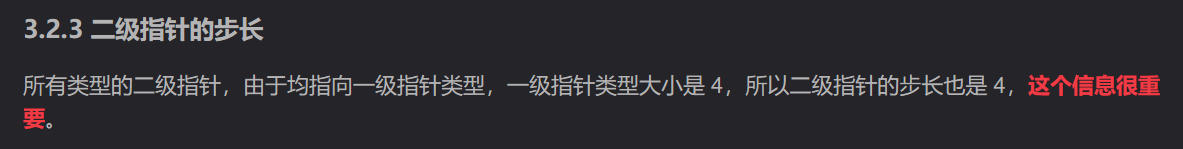
*指针变量所存的内容是内存的地址编号*

*二级指针是指向指针的指针， 我们可以通过它实现间接访问数据，和改变一级指针的指向问题*

*改变指向：*

*Int a = 10; int \*p = &a; int \*\*pp = &p;*

*\*pp = &b*



*指针访问数组：*

*int i, a[] = {3,4,5,6,7,3,7,4,4,6}; for (i = 0; i <= 9; i++)*

*{ cout << a[i] <<endl; }*

*很显然，它是显示 a 数组的各元素值。我们还可以这样访问元素，如下：*

*for (i = 0; i <= 9; i++)*

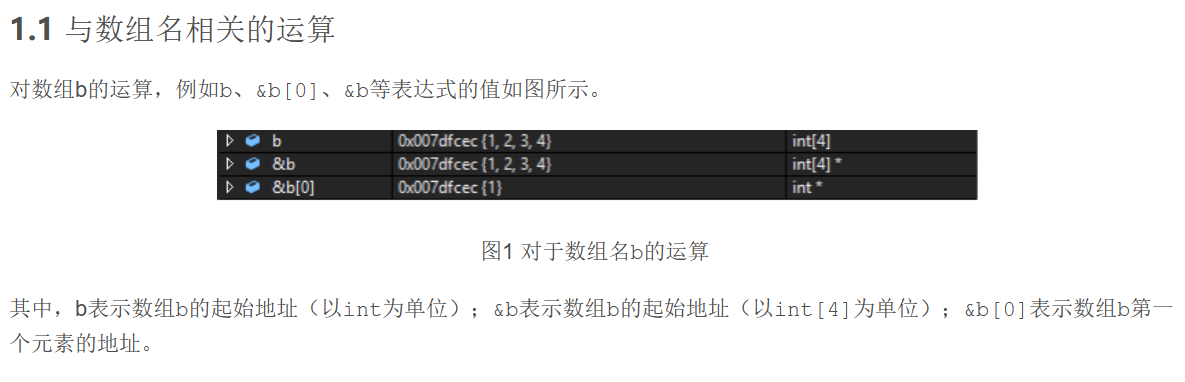
*{ cout << \*(a+i) << endl; } 它的结果和作用完全一样。*

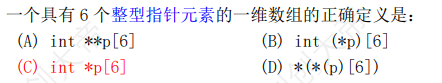
*数组名和指针的区别：*

*数组名是地址常量，不可改变 指针是变量可以改变*

*不可以使用野指针和空指针*

*11.数组*

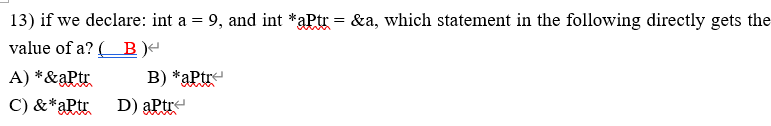




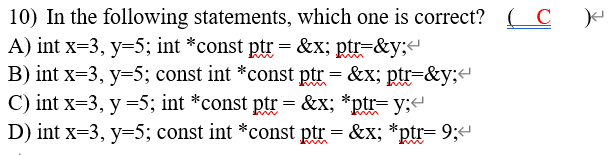
*int \*是数组的元素类型 p是数组名*



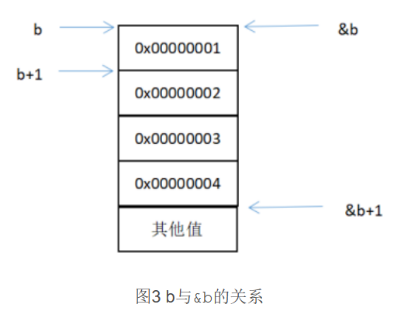
*P指向a[2][0]的地址，单位是一个元素，a[2][2]*



*Aprt是指针，解引用就行*



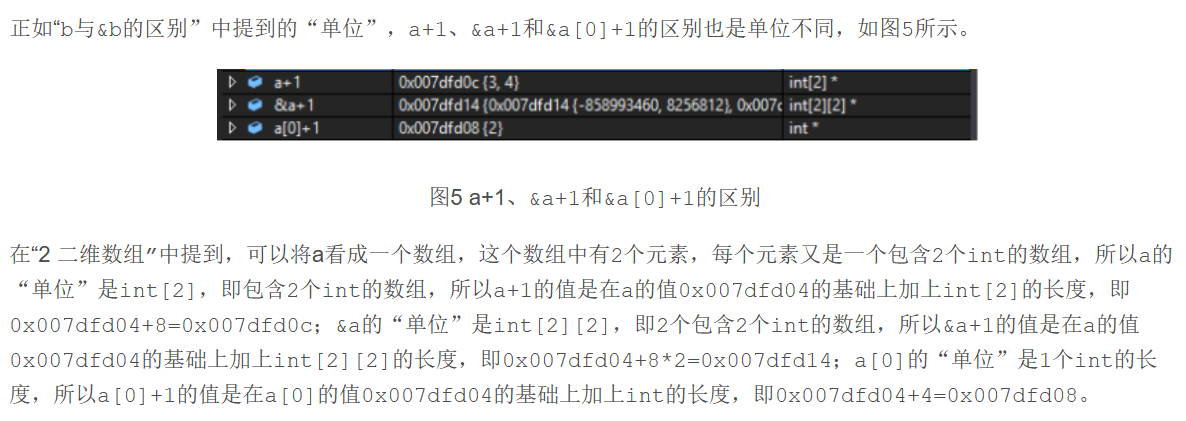
*A.指针常量不可以改变方向 B.不可改方向 C.给x赋值y D不可改变值和方向*





*B是数组的名字，他的单位应该是他的下一级元素，但是&b是数组的上一级，他的单位就是整个数组*

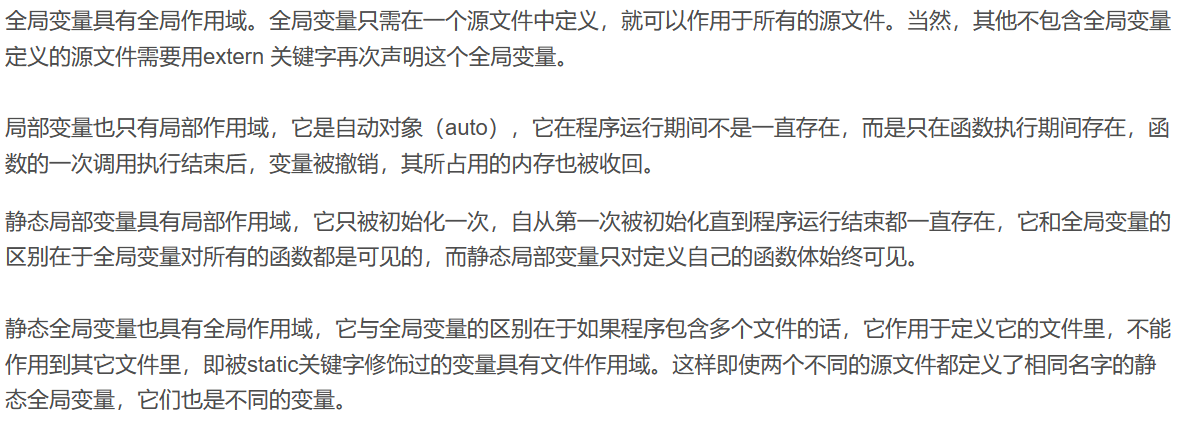


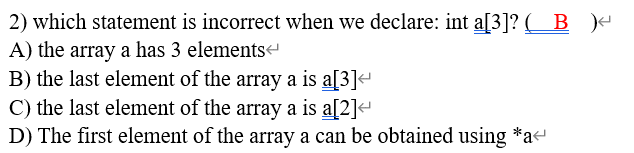


*A的单位是他的下一级元素，单位是一行的长度，&a的长度是整个二维数组，a[0]的单位是他的下一级，也就是一个元素*

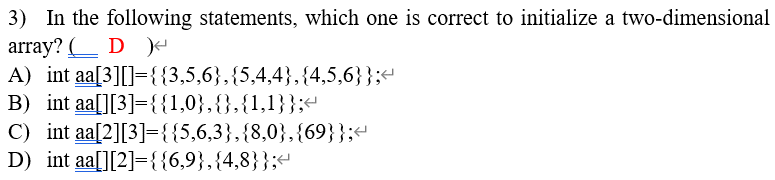
*12.储存类型*

*Auto 局部变量 static 静态变量 register 快速提取 extern 外部文件*



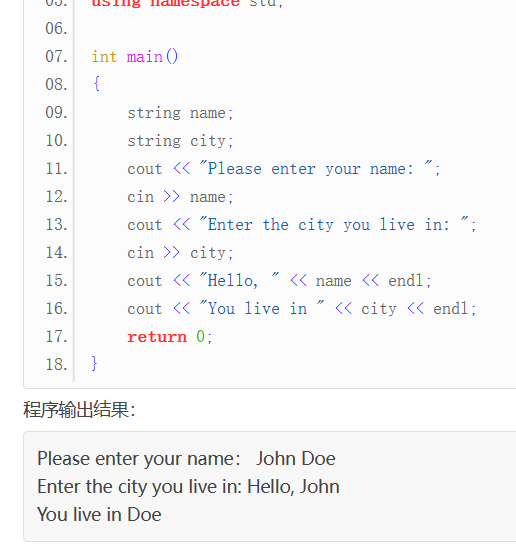


*定义的时候是长度，索引值0，1，2*



*二维数组必须要列数 可以写列不写行*

*A错 B列数为2 C行数为3*

**

1. *Cin*

*语句的执行结束是回车控制*

*Cin << 由空白符到空白符之间的内容，且不清除最后一个空白符*

*a.cin.get() 和cin.get(a) 读取一个字符（包括空白符）也不清理最后的空白符*

*cin.get(arr,n) n控制个数 也不处理最后的空白符*

*cin.getline(arr,n) 会清理最后的空白符*

*getline将最后的换行符变成\0也不会影响后续*

*不清理空白符 回车键会影响后续的输入输出*

*第一个cin读取John,残留空格Doe回车*

*下一个cin自动读取 Doe和回车*

*所以出现了问题*

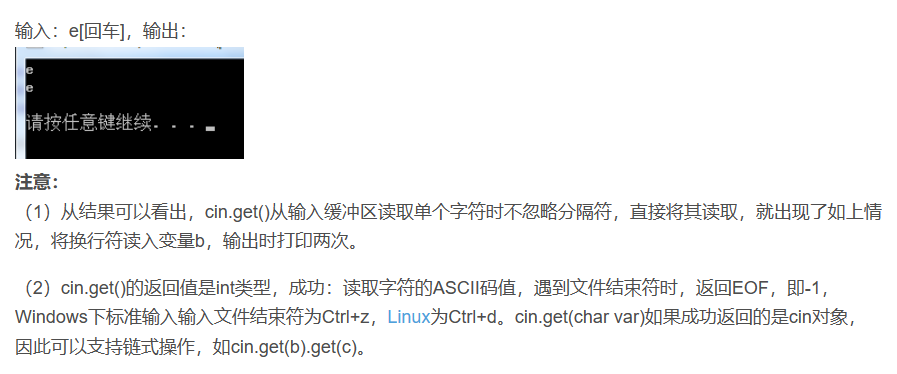
*Getline(cin,a) 识别到回车就结束*



输入与输出







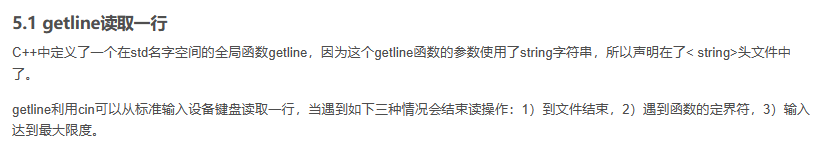




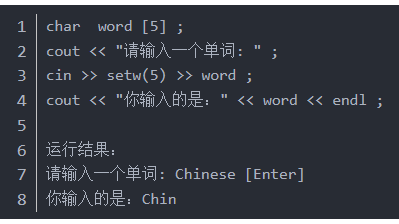












*3.科学计数法（浮点型）*



*1.23457e+07 1.234e-05 3.14159*

*2.#include<iomanip>库*

*In out manipulator 输入输出操纵器*

*Setw() 设置宽度， 非黏性，只影响紧跟着*

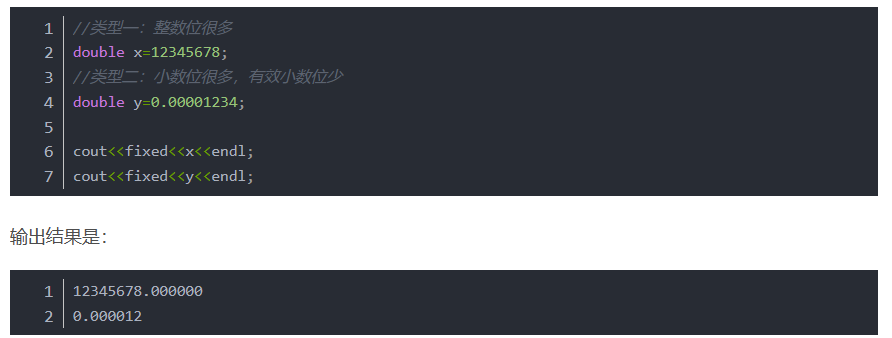
*Cin>>setw(20)>>a; a最多19个字符*

*Cout<<setw(10)<<endl; 靠右对齐*

*setprecision() 用来设置精度，默认为6，在没有fixed（定点型）的时候是整个有效位数，但是如果有fixed则是小数后的位数*



*默认有效6位数1.23460 最后一个0舍去 1.2346*



*4.cerr 用于输出警告*

1. *array类模板*

*定义*



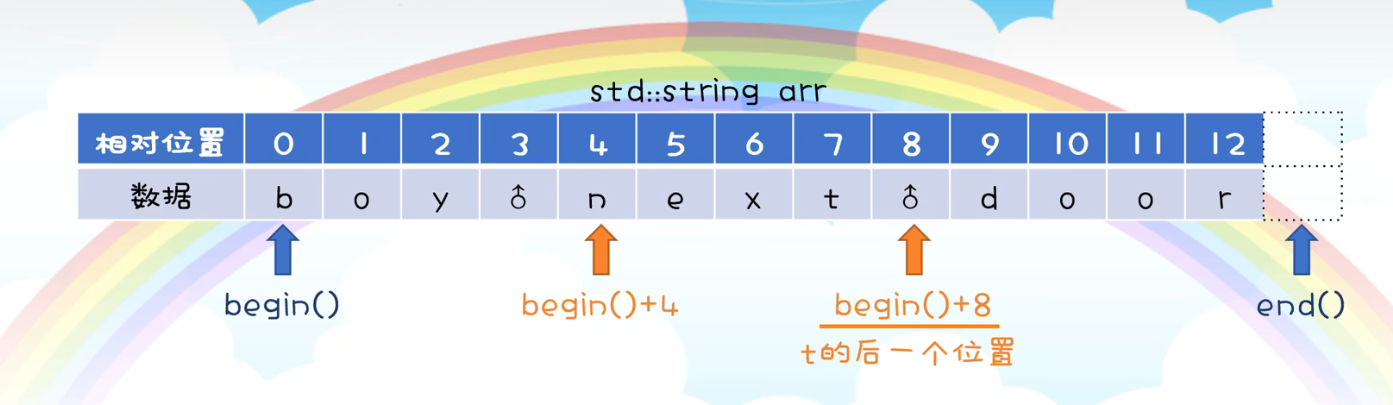


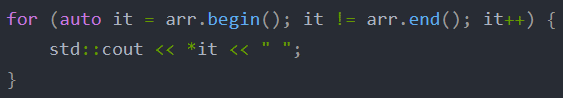
*常用使用*













数组与向量类



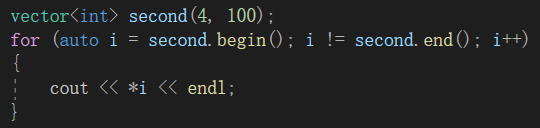
*2.vector类模板*

*定义：*

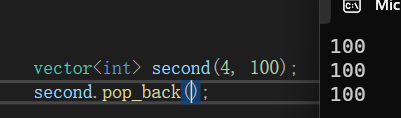
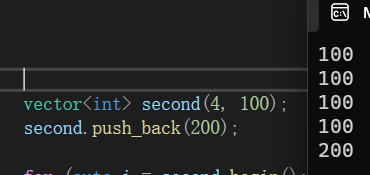


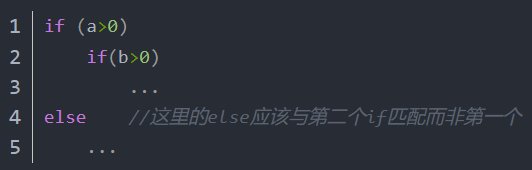
*Second: 构造4个100*

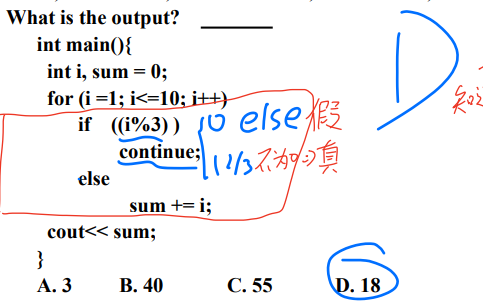
*用迭代器输出：*

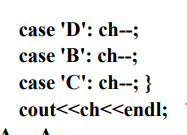
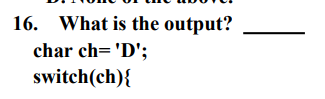


*常用语法：*





 *3+6+9=18*



*D->ch-3->ch=’A’*

*3.switch-case-default：*

*Case 后面必须是常量 case只是给入口，出口只看break*

*2.else摇摆问题*

*Else匹配与他最近的if*

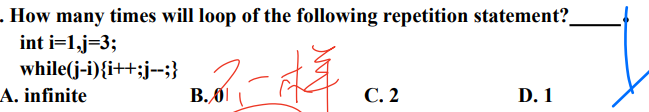
*没有语句块自然是离他最近的*

*1.如果不加花括号,if只能写一行，else只执行第一行，else后的语句视为顺序*

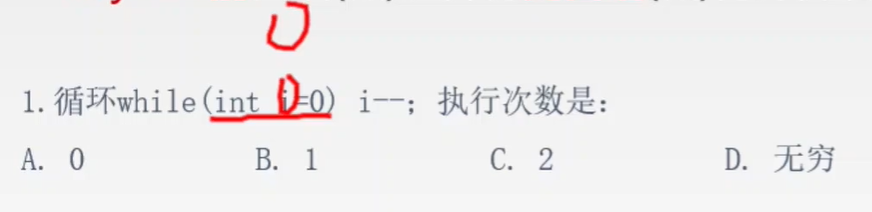


If、if-else、switch

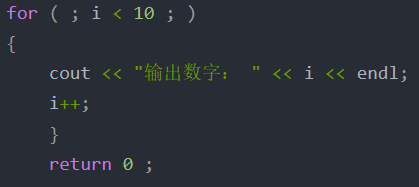
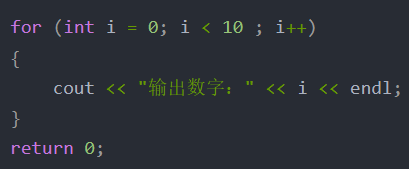


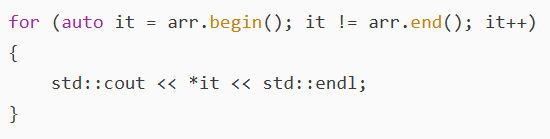
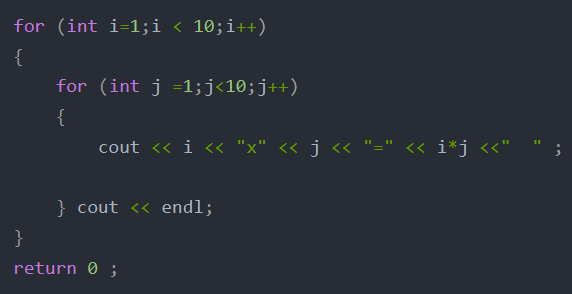


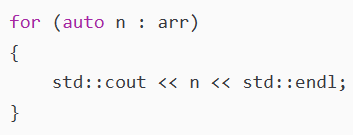
*第一轮 2 进入 第二轮 0 不进入*

*赋值运算换回自身引用 0*

*2.for语句*







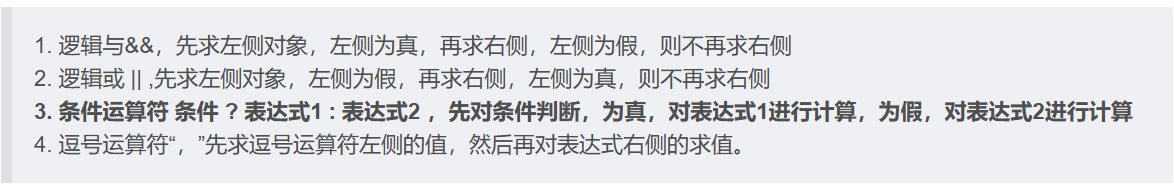
*上述方式是只读，如果需要修改arr里边的值，可以使用for(auto& n:arr)*

*1.while先判断后执行，do-while现执行后判断*



While、do-while、for



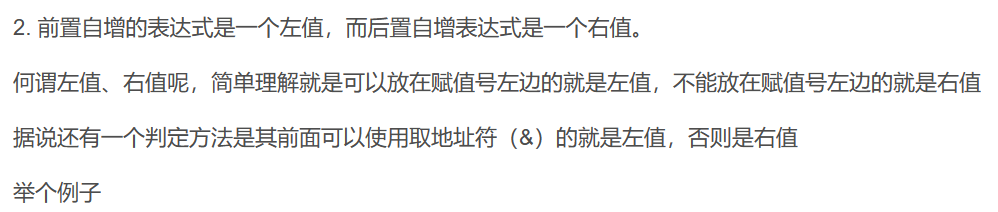






*1.\*p++ 是\*(p++) 2.++在前先计算后使用，++在后先使用后计算*

*3.&&比||的优先级高 4.? : 自右向左看 5. &&和|| 有短路功能*

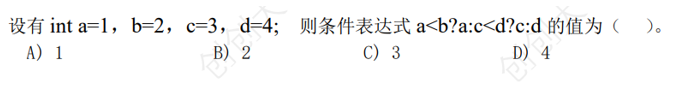


*后置优先级高于前置（右值高于左值）*



算术运算、逻辑运算

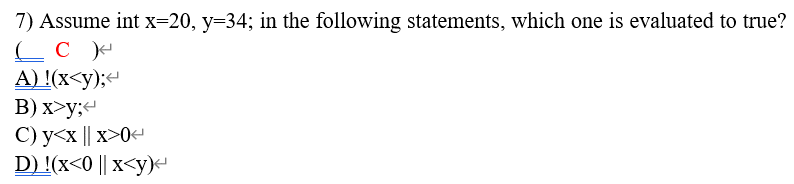




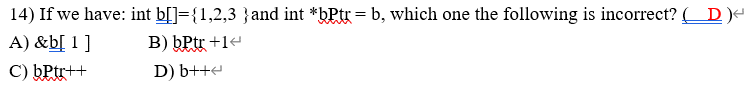
*(a<b) ? a : (c<d ? c: d) A*



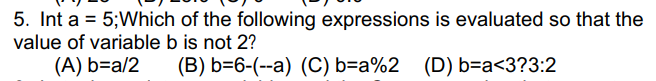
*先计算后使用*



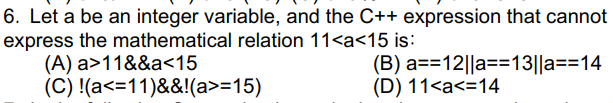
*A.false B.flase C.true D.false*



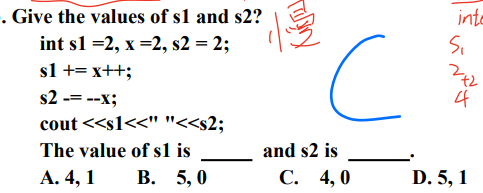
*b是常量不可以修改*



*A． 2 B. 2 C. 1 D. 2*

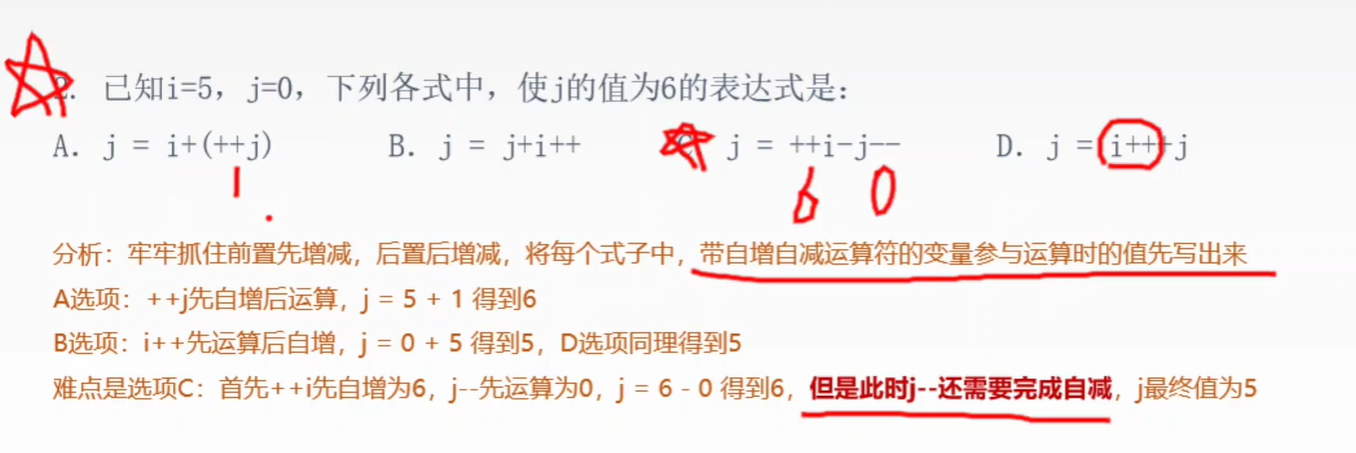


*D恒为1 0和1都小于14*

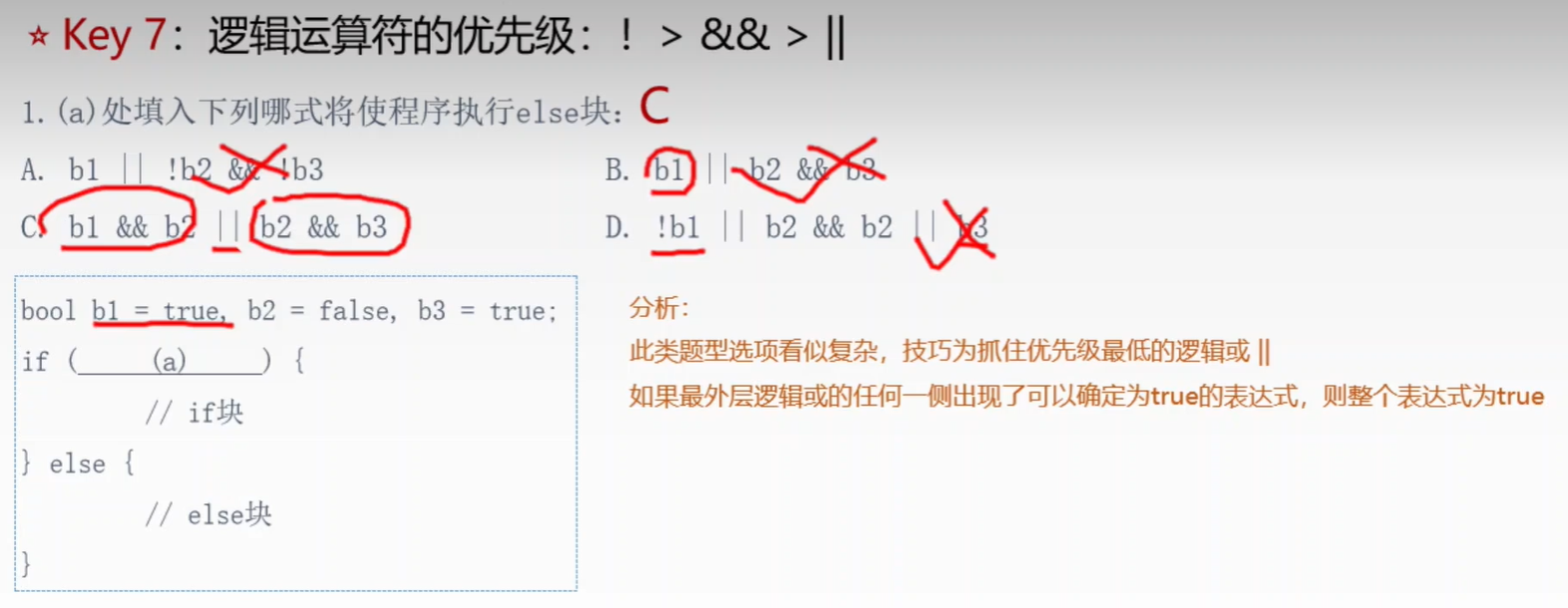


*S1=4 s2=0*

*任意x -2<x<2 都是1*

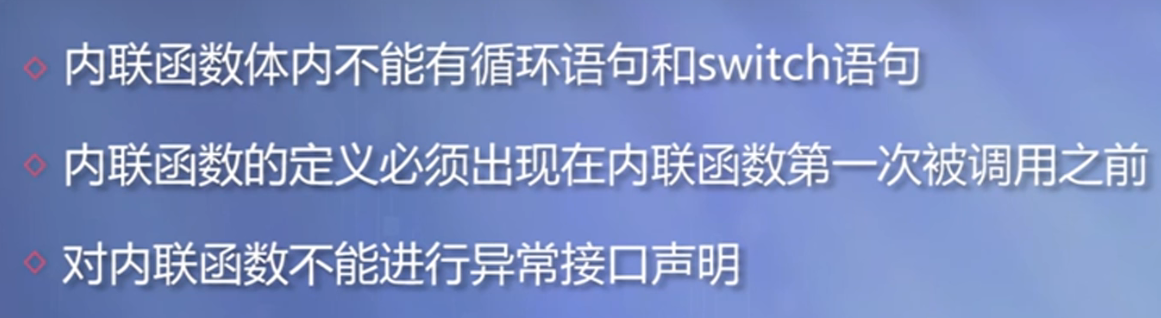


*C选项j=6之后还有自减操作，所以变成了5*



*&&的优先级高于|| 且有短路的功能*

*6.内联函数：由于函数体语句太简单，不需要进行额外开销，内联函数作用把函数体内语句放置调用处，减少传入和返回的消耗 （inline关键词）*



*5.函数模板：不需要知道形参一开始的数据类型*

*4.函数重载：名字相同功能不同*

*参数的类型 参数的个数 但是与返回值无关*

*3.默认值设置后所有的形参也必须有默认值*

*声明和定义只能有一个设置默认值*

*2. 数组传入函数时只传入了地址，没有传入数组长度*

1. *值传递不改变实参，引用传递和地址传递改变实参*



函数与递归



*9.递归的写法*

*函数声明->基础情况->递归体->操作*

*递归的本质：微操作+超级操作 参考罗汉塔问题*

*8.直接rand无法真正随机，每次运行结果都一样*



*语法 rand()%n + m 从m开始n个数，本质是取模运算*



*加上srand（）随机时间种子就可以做到真正的随机*

*7.作用域规则*

*四大作用域：语句块作用域、函数作用域、全局命名空间作用域、函数原型作用域*

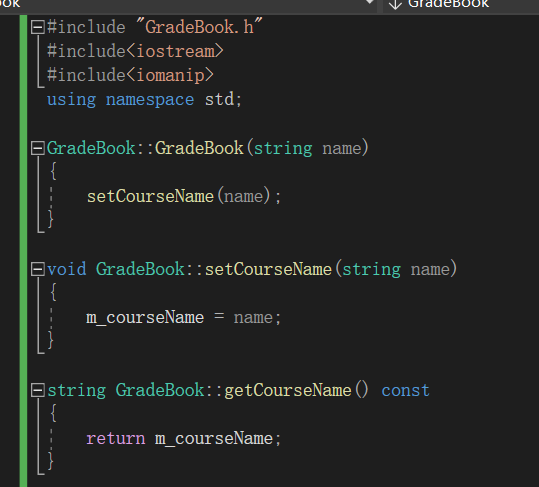
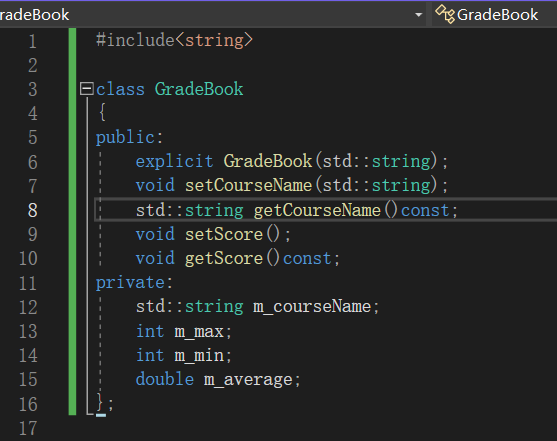
*语句块作用域：起始于标识符处，直到花括号结束 （形参和函数体内部的变量都是）*

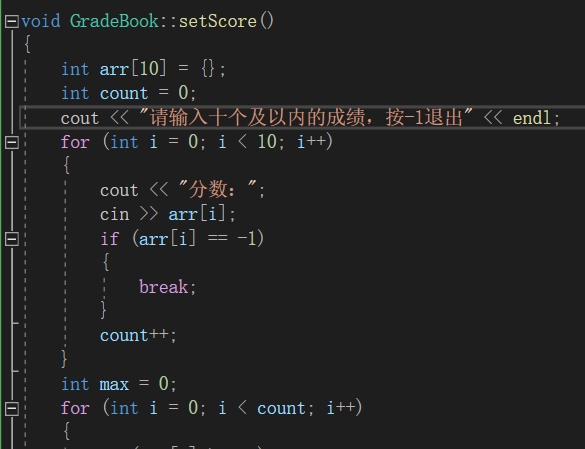
*函数作用域：标签和switch中的case是唯一的函数作用域*

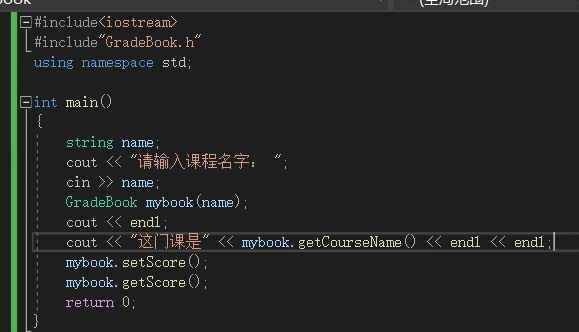
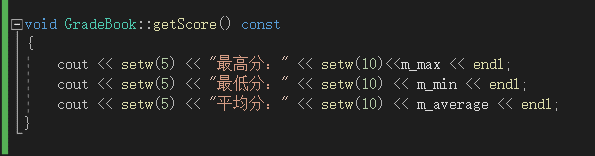


*全局命名空间作用域：全局变量，函数声明，函数原型*

*函数原型作用域：函数声明内的形参列表*









类与对象

