C++Notes2

c++ 存储类型和占内存大小 ： short 半个机器字长 int 一个机器字长 long 1个或者2个机器字长 在32位系统 int long 是一样的

机器字长：与机器位数有关

c++ 字符串拼接 两个字符串直接 用“”分割不需要+ 但是好像这样没意义？无法拼接费字符串的值

“\”反斜杠在c++表示续行符 将当前行的代码与下一行做同一行处理； 用在多行宏命令时方便查看代码

“\”之后不能添加任何符号 ；包括空格

整型数值赋值 无符号书unsigned 会取模后赋值，有符号signed 由编译器决定实际值：一般是与无符号相同，但不能保证

c++ 的初始化不是赋值！！！ 创建对象并且赋初始值。

Int a = 1； 初始化a = 1 包括了内存分配？

a =2 赋值 修改了内存中的值？

C++内置类型的初始化， 在函数体之外的变量都会初始化为0 而在函数体内的不初始化 处于未定义状态，

建议对每个内置类型对象进行初始化

声明&定义

定义：变量只有一个定义，定义时 申请分配了内存，

声明：向程序声明 变量的名字和类型 定义也是一种声明方式， 通过extern 关键字来声明已定义的变量

引用：变量的另一种称呼，会修改原变量

const 引用： 指向const 对象的引用

typedef 重新定义类型的名字；

用途：

隐藏特定类型的实现，强调类型的目的（这个变量类型是干嘛的）

简化复杂的定义，

目前类型的目的；

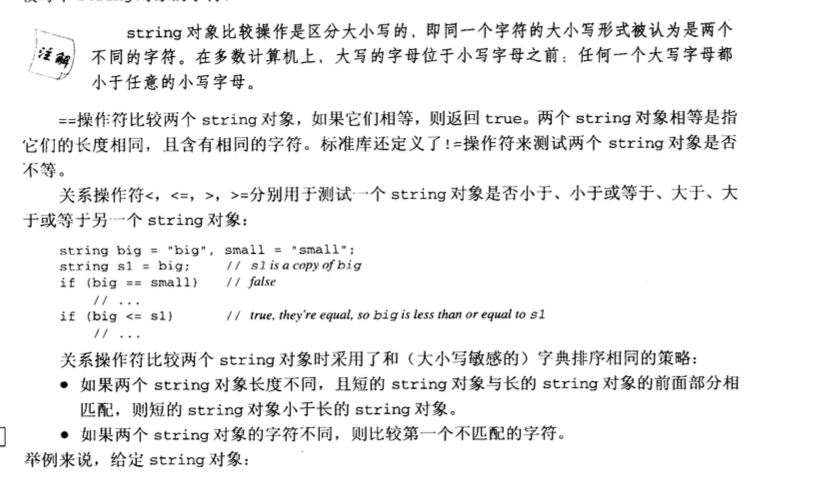
设计类： 从一个类需要完成的操作开始分析， 通过操作分析类需要的数据， 注意 类结束需要“;”

类成员变量在定义的时候未初始化， 而在构造函数里初始化

头文件在include进来的时候是需要编译事件的所以应该尽量轻量化

头文件设计的一些规则： 头文件用于声明而不是定义，不要在头文件定义变量，

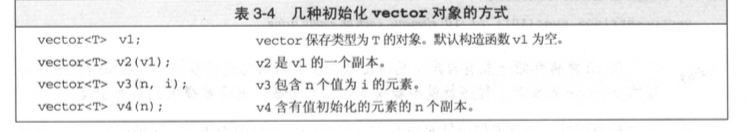
c++ string 可以比较大小



string 复制包含了释放str1 的内存 分配狗内存来存储str2

vector 类型存储同一类型的对象集合，称为容器，

类模板 泛型编程



vector 对象可以自增内存大小 动态存储内容

iterator 迭代器： 来遍历 容器类； 是一个指针 通过\*来访问值

begin（） end（） 返回迭代器用户遍历。

优选vector 类 少用数组和指针 指针通过累加可以之下下一个内存存储单元 一连串的内存 每端内存分配着同一种类型的大小的内存单元

字符数组在 用字符串来初始化时会自动插入空字符“\0”

指针：

避免使用未初始化的指针 ， 空指针错误！

在解应用\*P时都会错误 使用值之前尽量初始化所有值 ，指针是尤其

指针的初始化及赋值： 0 对象地址 对象的下一个地址 另一个指针

void\* 指针 可以保存任何类型的对象的地址 只管地址 不管值

指向const对象的指针 const double \*ptr 标准指向const double的指针， ptr 本身不是const的 他指向内存位置的对象是const的 同理 const 对象只能 对应 指向 const 指针，但是指向const的指针可以接受非const对象 但是无法通过prt 修改对象值

:不能保证指向const 对象的指针的值不被修改 因为他可以指向非const 对象 从而原对象可以直接修改

const 指针 指针的不能修改

int a = 0；

int \*const ptr = @a； ptr 需要定义时初始化，并且不能修改ptr的值 但是可以修改ptr 指针指向对象的值。

指向const 对象的const 指针 ，

const double pi = 3.1415926

const double \*const ptr\_pi = &pi； 指向const对象的const 指针

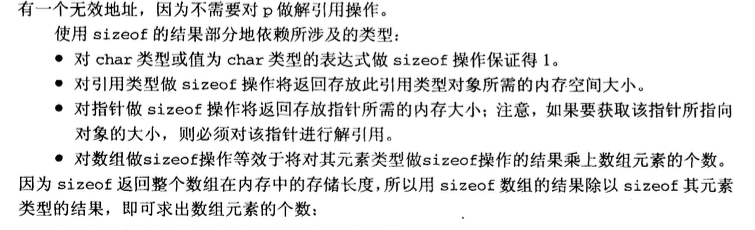
typedef string \*pstring

const pstring ctr 类型为const 指针，

动态分配的内存 堆，存储动态分配的对象，new delete 需要对应使用delete[] 删除数组

strlen（）返回不含空字符的字符串长度

sizeof 操作符 返回对象类型的长度，size\_t 单位字节



new delete

new 在堆创建一个对象并且返回对象地址 初始化指针对象。

new int ；未初始化对象 new int（）；初始化了对象

new 可能抛出异常

new的对象需要显示调用delete 方法来释放内存。

Delete 必须对应new

Delete 之后指针变为未定义状态， 因此正常delete之后需要将指针置为0

显示类型转换： static\_cast dynamic\_cast const\_cast reinterpret\_cast

double var;

static\_cast<int> (var); double -> int

const\_cast 转换变量const 性质

static\_cast 是系统隐士执行的转换 不关心精度损失

函数不能返回数组 但是可以返回指针- 指针可以指向数组

默认实参 可以省略参数 编译器自动补全 上默认实参

函数定义时 适当使用指针活引用来进行优化

