C++ 笔记

Kang

2020 年 9 月 3 日

1 复合类型

1.1 字符串

字符串输入：cin读取字符串以空白（(空格、制表符和换行符)来确定字符串的结束位置，所以cin在获取字符数组输入时只能读取一个单词。将其储存在数组中时，会自动添加空白字符'\0'。每次从输入中读取一行使用getline()，通过回车键输入的换行符来确定输入的结尾。输入的参数有两种，为:

cin.getline(name, length)

上述语句可以读取19个字符。此外，还可以通过cin.get()来读取，该函数的参数和getline()类似，但是读取到换行符之后不会自动读取并删除换行符，单独调用无参数的cin.get()则会自动从输入队列中读取一个字符，可以消除掉换行字符。但是C++常使用指针来处理字符串。

1.2 标准库类型string 类

使用string前要包含string库，string初始化方式可以为 string s4(10, 'c')，将s4初始化为十个'c'。string可以执行的操作有：注意.size()输出的是一个无符号数，和负数进行比较时，负数会自动的转化为$2^16 = 65536$(16位)的

表 3.2： string 的操作

os<<s	将 s 写到输出流 os 当中，返回 os
is>>s	从 is 中读取字符串赋给 s，字符串以空白分隔，返回 is
getline(is, s)	从 is 中读取一行赋给 s，返回 is
s.empty()	s 为空返回 true，否则返回 false
s.size()	返回 s 中字符的个数
s[n]	返回 s 中第 n 个字符的引用，位置 n 从 0 计起
s1+s2	返回 s1 和 s2 连接后的结果
s1=s2	用 s2 的副本代替 s1 中原来的字符
s1==s2	如果 s1 和 s2 中所含的字符完全一样，则它们相等；string 对象的相
s1!=s2	等性判断对字母的大小写敏感
<, <=, >, >=	利用字符在字典中的顺序进行比较，且对字母的大小写敏感

补码，例如$-32 -> 65536 - 32 = 65504$，然后和无符号数进行比较。在使用加法运算符进行字符串的拼接时，只需要每个加法运算符左右两侧有一个是string对象即可。

基于范围的for循环: for (declaration: expression){statement;}，declaration会创建一个变量储存expression中的元素(直接拷贝复制)，然后随着循环不断向后移动，类似于python 的 for i in...

.find(const string& a)函数可以找到子字符串a在对应的string中的第一个位置的索引，如果没有此字符，那么返回string::npos。 函数 .substr(size_type pos = 0, size_type len = npos)可以进行字符串的切割，从pos开始切割len个字符(包含pos在内)。

函数decltype(a) 可以返回a的类型，可以用于定义同类型的变量，例如上述类型中的size_type(可以看做是一个无符号整数)。

1.3 标准库类型类模板vector

使用vector之前要包含库vector。定义变量格式为: vector<T> name，定义T类型的数组name。vector定义不进行初始值的指定时，库会创建一个"值初始化"的元素初值，例如int的会初始化为0，如果元素是某种类类型，则元素由类默认初始化。vector可以调用函数push_back(a)，将元素a添加到数组的末尾。此外，vector也可以和string类似，调用size和empty函数进行判断。数组长度返回值的类型为 vector<T> size_type 类型。

1.4 迭代器

使用迭代器时需要一个容器，对容器v使用迭代器时，定义:auto b = v.begin(), e = v.end();end()函数返回的迭代器称作尾后迭代器，指向的是容器中本不存在的"尾后元素"，并不是最后一个元素，是一个虚拟的位置。b和e都是一个迭代器，其类型由容器定义，为容器元素类型加iterator，常量元素则为const_iterator，例如vector<int>::iterator和string::iterator。

1.5 结构体简介

结构体初始化时，也可以像数组一样初始化 {...}。结构体之间可以使用=直接赋值

1.6 枚举类型

enum spectrum red, orange, yellow. 则由spectrum 所定义的变量只能在red, orange, yellow这三个符号常量中取值。如果想定义符号常量，可以将类型值spectrum省略。

1.7 引用

int &变量名 = 另一个变量名(必须是对象，不能是表达式或者数值)，引用类型定义时必须进行初始化。

1.8 指针与自由储存空间

地址运算符& 间接值或者解除引用运算符*，分别代表地址和地址中储存的变量值。int* 表示指向int类型的指针，是指针不是int类型。

动态分配内存的语句 typeName* pointer_name = new typeName; 其中typeName可以是对象，此时指针指向的是一个"数据对象"(非面向对象的对象，只是某个内存地址中的信息)。动态数组 int* p = new int [10];释放内存对应delete [] p，数组的长度可以是变量。

1.9 指针、数组和指针算术

C++将数组的名字视作首元素的地址，指针变量加一后，其数值增加一个指向的类型的所占用的字节数(不一定是一个字节)。指针是变量，但是数组名是常量，定义指针的语句将指针变量的名字去掉即为其类型。在定义指针的语句中，*和&互为逆运算，要看清两边的数据类型再进行赋值。例如:

int (*b)[3]说明b是int(*)[3]类型的指针，指向一个int的3个元素的数组。数组名字取&是整个数组的地址，视作一个整体。例如int a[3] = 1,2,3,则&a的类型是int(*)[3]的。取地址后自动增加一个*，数组名字的类型也会增加一个*。

C++在解释符号数组或指针名[偏移量i] 时，将统一转化为*(数组或指针名字 + 偏移数i)，所以指针也可以像数组名字一样访问数组。

指针与字符串cout在打印一个字符串时，并不会直接将整个字符串的元素传递，而是传递第一个字符的地址，继续打印知道遇到 \0停止打印。因此直接传递给cout一个字符串的首地址，会自动打印这个字符串。**用引号引起的字符串，也被解释为一个数组名字，即为第一个元素的首地址**。实际上，在诸多C++表达式中，char数组名、char指针以及用双引号括起的字符串都被解释为第一个元素的地址。

使用指针指向一个结构体时，访问成员应该使用-¿，不能使用.运算符。但是可以使用(*指针名字).访问。

2 循环语句

{}代表一个代码块，在代码块中定义的变量只是局部变量，没有代码块默认第一个语句为执行的代码块(不包括后续的语句)，编译器会自动地省略缩进。

C++的二维数组引用:[列][行]。