C++ Primer Plus （5-9章）阅读并总结关键语法点

1. 语句：

复合语句是指用花括号括起来的语句和声明的序列，复合语句也被称作块。

语句的作用域：定义在控制结构当中的变量只在相应语句的内部可见，一旦语句结束，变量就超出其作用范围。

条件语句:if 语句;switch 语句

迭代语句：while 语句； for 语句（for 语句中定义的对象只在for循环体内可见。）do while 语句

### 跳转语句：break（break只能出现在迭代语句或者switch语句内部。仅限于终止离它最近的语句，然后从这些语句之后的第一条语句开始执行）；Continue（continue语句终止最近的循环中的当前迭代并立即开始下一次迭代）goto（作用是从goto语句无条件跳转到同一函数内的另一条语句）

1. 函数:形参和实参：实参的类型必须与对应的形参类型匹配。函数的调用规定实参数量应与形参数量一致。

局部对象：形参和参数体内部定义的变量统称为局部变量，它们对函数而言是"局部"的，仅在函数的作用域内可见，同时局部变量还会隐藏外层作用域中同名的其他变量。

局部静态对象：在程序的执行路径第一次经过对象定义语句时候进行初始化，并且直到程序终止才会被销毁。

函数的三要素：（返回类型、函数名、形参类型）。

函数可被声明多次，但只能被定义一次。

分离式编译：允许把程序分割到几个文件中去，每个文件独立编译。

当形参是引用类型，这时它对应的实参被引用传递或者函数被传引用调用。当实参被拷贝给形参，这样的实参被值传递或者函数被传值调用。为函数传递一个数组时，实际上传递的是指向数组首元素的指针。

函数指针：函数指针指向的是函数而非对象。

第七章 类

类的基本思想是数据抽象和封装。抽象是一种依赖于接口和实现分离的编程技术。封装实现了类的接口和实现的分离。

（1）this

任何对类成员的直接访问都被看作this的隐式引用。

（2）在类的外部定义成员函数

类外部定义的成员的名字必须包含它所属的类名。

（3）构造函数

### Public：使用public定义的成员，在整个程序内可被访问，public成员定义类的接口。

Private：使用private定义的成员可以被类的成员函数访问，但是不能被使用该类的代码访问。

### 类的其他特性

（1）重载成员变量

（2）类数据成员的初始化

类内初始值必须使用=或者{}的初始化形式。

（3）基于const的重载

（3）类类型：对于一个类来说，在我们创建他的对象之前该类必须被定义过，而不能仅被声明。

（4）友元

如果一个类指定了友元类，则友元类的成员函数可以访问此类包括非公有成员在内的所有成员。

一个类就是一个作用域。

### 类的静态成员

### 声明静态成员：在成员的声明之前加上关键词static。类的静态成员存在于任何对象之外，对象中不包含任何与静态成员有关的数据。

类有两项基本能力：一是数据数据抽象，即定义数据成员和函数成员的能力；二是封装，即保护类的成员不被随意访问的能力。

第八章 IO库

C++语言不直接处理输入输出，而是通过一组定义在标准库中的类型来处理IO：iostream处理控制台IO；fstream处理命名文件IO；stringstream完成内存string的IO。

IO类：（1）IO对象无拷贝或复制。进行IO操作的函数通常以引用方式传递和返回流。（2）刷新输出缓冲区

flush刷新缓冲区，但不输出任何额外的字符；ends向缓冲区插入一个空字符，然后刷新缓冲区。

ifstream作用：从一个给定文件读取数据；

ofstream作用：从一个给定文件写入数据；

fstream作用：读写给定文件

### string流：

### Istringstream：从string读取数据；

Ostringstream：向string写入数据；

Stringstream：既可从string读数据也可以向string写数据

第九章 顺序容器

顺序容器为程序员提供了控制元素存储和访问顺序的能力。

Vector：可变数组大小。支持快速随机访问。在尾部之外的位置插入或删除元素可能很慢。

Deque：双端队列。支持快速随机访问。在头尾位置插入/删除速度很快。

List：双向链表。只支持双向顺序访问。在list中任何位置进行插入/删除操作速度都很快。

forward\_list：单向链表。只支持单向顺序访问。在链表任何位置进行插入/删除操作速度都很快。

Array：固定大小数组。支持快速随机访问。不能添加或删除元素。

string与vector相似的容器，但专门用于保存字符、随机访问快。在尾部插入/删除速度快。

迭代器

标准库的迭代器允许我们访问容器中的元素，所有迭代器都是通过解引用运算符来实现这个操作。

一个迭代器返回由一对迭代器表示，两个迭代器分别指向同一个容器中的元素或者是尾元素之后的位置。它们标记了容器中元素的一个范围。begin是容器中第一个元素的迭代器，end是容器尾元素之后位置的迭代器。当将一个容器初始化为另一个容器的拷贝时，两个容器的容器类型和元素类型都必须相同。

容器大小操作：size：返回容器中元素的数目；empty：当size为0返回布尔值true，否则返回false；max\_size：返回一个大于或等于该类型容器所能容纳的最大元素数的值

### 顺序容器操作：

（1）向顺序容器添加元素：使用push\_back:追加到容器尾部；使用push\_front:插入到容器头部；在容器中的特定位置添加元素:使用insert；插入范围内元素:使用insert

回下标为n的元素的引用。如果下标越界，则抛出out\_of\_range异常 |

（2）删除元素

c.pop\_back()：删除c中尾元素。若c为空，则函数行为未定义。返回void；

c.pop\_front()：删除c中首元素。若c为空，则函数行为未定义。返回void

c.erase(p)：删除迭代器p所指定的元素，返回一个指向被删除元素之后元素的迭代器，如p指向尾元素，则返回尾后(off-the-end)迭代器。若p是尾后迭代器，则函数行为未定义。

c.erase(b, e)：删除迭代器b和e所指定范围内的元素。返回一个指向最后一个被删除元素之后元素的迭代器。若e本身就是尾后迭代器，则函数也返回尾后迭代器

c.claer()：删除c中的所有元素。返回void。

（4）改变容器大小

reseize用于扩大或者缩小容器；resize操作接受一个可选的元素值参数，用来初始化添加到容器内的元素；如果容器保存的是类类型元素，且resize向容器中添加新元素，则必须提供初始值，或者元素类型必须提供一个默认构造函数。

改变string的其他方法

assign 替换赋值，总是替换string中的所有内容；insert 插入；append 末尾插入，总是将新字符追加到string末尾

replace 删除再插入

string搜索操作

s.find(args)：查找s中args第一次出现的位置；

s.rfind(args)：查找s中args最后一次出现的位置

s.find\_first\_of(args)：在s中查找args中任何一个字符第一次出现的位置

s.find\_last\_of(args)：在s中查找args中任何一个字符最后一次出现的位置

s.find\_first\_not\_of(args)：在s中查找第一个不在args中的字符

s.find\_last\_not\_of(args)：在s中查找最后一个不在args中的字符

### 容器适配器

顺序容器适配器：

stack; queue; priority\_queue;

适配器是一种机制，能使某种事物看起来像另外一种事物。