C++ Primer笔记

1. C++11新增语法

---》目录后附有C++11语法出现的页数

1、列表初始化（list initialization）

（1）现在无论是构造形式初始化，还是赋值形式，都可以使用花括号

---》例如：int a{10}, b = {10};

1. 使用列表初始化且初始值存在丢失信息（如精度）的风险，则编译器将报错

---》例如：double转int

（3）列表初始化返回值（见p203）

---》函数可以返回列表初始化返回值（初始化返回的临时变量）

---》如果函数返回单个对象而非数组等，列表元素不超过1个

2、constexpr和常量表达式（p59）

（1）常量表达式：不会改变且在编译过程就能得到计算结果的表达式

（2）constexpr变量与const变量的区别：

---》const：可以接收非constexpr函数的返回值，但会在运行时计算

---》constexpr：只能接收字面值常量，或者constexpr函数返回值（编译时确定）

---》声明为constexpr类型，可以由编译器检查变量是否是一个常量表达式（比const更严格）

（3）constexpr指针

---》constexpr指针的初始值必须是nullptr或者0，或者是存储于某个固定地址中的对象（全局变量）

---》constexpr修饰指针：只有指针常量（顶层，不可修改指向），没有常量指针（底层）

3、using与typedef

---》typedef int size；

---》using size = int；

4、auto类型推算（不用深究，会用就好）（p61）

（1）使用auto在一条语句中声明多个变量（只能有一个基本数据类型）

---》auto a = 0, \*p = &a; // 正确：int, int\*

---》auto a = 0, b = 3.14; // 错误：int, double

（2）auto一般会忽略掉顶层const（去常const）

（3）auto的引用保留顶层const

5、decltype类型推断（不用深究，会用就好）（p63）

（1）decltype()内表达式是变量：则返回该变量的类型（包括顶层const和引用在内）

---》注意：引用从来都作为其所指对象的同义词出现，只有用在decltype处是一个例外

（2）如果变量表达式的结果是左值，编译器会推断为其引用类型

---》decltype(\*p) => p指向对象的引用类型，解引用操作结果是左值

---》decltype((a)) => a的引用，编译器理解为左值表达式

---》decltype(a) => a本身是引用，才是引用

（3）decltype()内表达式不是变量：则返回表达式结果对应的类型

6、类内初始值(in-class initializer)

---》类似函数缺省参数，指定默认初始值

7、范围for（range for）（详见p168）

（1）每次迭代，左边赋值(=)为右边的下一个元素值

---》for（auto c：str） // 值传递

---》for（auto &c：str）// 引用，可修改右边内容

（2）使用范围for处理多维数组时，除了最内层循环外，其他所有循环的控制变量都应该是引用类型

---》避免数组被自动转化成指针，参见p114

8、容器迭代器begin和end

---》对象是常量，返回只读迭代器；否则，可写迭代器

---》新增：cbegin和cend，不论对象，只返回只读迭代器

9、C++11新增std::begin和std::end函数（了解）

---》用于返回数组首尾地址，用法类似容器迭代器

---》例如：std::begin(int\_arr)、std::end(int\_arr)

10、函数变长形参表的解决方法：

（1）所有实参类型相同：用一个initializer\_list类型的形参（见p198）

---》initializer\_list类型类似于vector

---》区别：列表中元素都是const；同类型对象拷贝时，副本共享原列表元素（只读）

（2）实参类型不同：可变参数模板（见p618）

（3）仅用于C和C++通用的类型：省略符形参“...”（见p199）

11、尾置返回类型

---》auto foo(int) -> int(\*)[10];

---》函数返回数组指针

12、constexpr函数（了解，见p214）

（1）定义规则

---》函数返回值结果和实参都得是字面值

---》有且只有一条return语句

（2）用处：主要用于初始化constexpr变量，确保编译时就可以计算出结果

13、=default 和 =delete

---》含义：使用或删除(删除)类的默认成员函数

---》类的默认函数包括：缺省构造、缺省析构、拷贝构造、拷贝赋值

---》单例模式中会涉及删除，类似构造析构私有化

14、委托构造函数（了解，见p261）

15、overide、final关键字

16、lambda表达式

---》一种仿函数对象

---》mutable: 在lambda中可以修改按值捕获的外部变量。这样的lambda表达式，其参数表不可省略。

17、泛型元组

---》tuple可以理解为是对老版本pair类模板的扩展，其中的元素个数不再限于两个，且其功能更加丰富。

18、右值引用

（1）左值和右值

---》可以取地址的值就是左值，左值通常具名；不可取地址的值就是右值，右值通常匿名。

（2）左值引用、右值引用、常左值引用

---》左值引用只能引用左值，

---》右值引用只能引用右值

---》常左值引用既可以引用左值，也可以引用右值，万能引用

---》任何临时变量只具语句级生命期，(右值)引用可将其寿命延长至和该引用的生命期一样。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

补充：

---》变量名可以理解为虚拟内存地址(指针)，引用变量也可以这么理解

---》所以，非const的右值引用也是左值，也可以被修改，他只是延长了临时变量的生命周期

---》左值引用的目标是左值，其本身也是左值，既可取地址，亦可被修改

---》右值引用的目标是右值，但其本身是左值，既可取地址，亦可被修改，因此右值引用可以被左值引用引用

---》因此左值引用不能被右值引用引用，但是通过move可以将左值转换为右值，被右值引用引用

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

（3）通用引用

---》在函数模板隐式推断过程中，若实参为左值，则T&&被推断为左值引用，若实参为右值，则T&&被推断为右值引用，这样的引用被称为通用引用。

---》以下情况不会产生通用引用：

### 只有T&&才是通用引用，加了const就不会产生该语法。

### 不做隐式推断，不会产生通用引用的语法。

---》类似的规则也同样适用于基于auto关键字的类型推导。

（4）引用折叠

---》C++98/03标准不允许声明引用引用的引用，但C++11允许这么声明，并根据引用折叠的规则将其处理为简单引用的形式

---》规则：只有右值引用的右值引用才是右值引用，其它情况下的引用折叠一律被视为左值引用。

（5）move和forward（左右值转化）

---》std::move：无论左右值，都化为右值

---》右值引用也是左值，当传参给右值引用的形参时，还是要调用move转化右值引用为右值

---》std::forward：左值返回左值，右值返回右值，完美转发

---》在函数模板中，通过forward可将其接收到的参数以完全相同的类型传递给另一个函数，谓之完美转发

---》无论左值引用还是右值引用，其引用本身都是左值！！！

19、移动拷贝、移动赋值、(左值和右值)引用函数

---》跳过：p473~485，了解

20、标准库中定义的函数对象

---》用于替换算法中的加减乘除、关系、逻辑运算符，p510

21、可调用对象与function（参见p511）

---》包括：函数、函数指针、lambda表达式、bind创建的对象、重载了函数调用运算符的类

---》标准库定义的function类型模板，用来存储可调用对象，其调用形式应该与模板参数类型相同

============================================

二、warning（重要点）

1、程序员常常在调试时添加打印语句。这类语句应该保证“一直”刷新流（std::endl）。否则，如果程序崩溃，输出可能还留在缓冲区中，从而导致关于程序崩溃位置的错误推断

2、切勿混用带符号类型和无符号类型

---》当带符号类型为负值并与无符号类型运算时，会类型提升变成一个很大的正数，导致结果异常

3、一行定义同类型的变量及其指针

---》int \*a, b; // 可读性好，a=int\*，b=int

---》int\* a, b; // 合法但是容易产生误导

1. 变量的声明和定义（extern）

---》extern int a; // 声明，为程序所知

---》int b; // 声明并定义，创建与名字相关的实体

---》extern int c = 10;// 定义，一旦赋了值，就抵消了extern的作用

5、const对象仅在文件内有效

---》类似static关键字

---》多文件共享const：对于const变量，不管是定义还是声明都添加extern关键字，以告诉编译器

6、凡是使用了索引或迭代器的循环体，都不能改变所属序列或容器的大小

---》否则会造成逻辑错误或迭代器失效

7、求值顺序

（1）只有4种运算符明确规定了运算对象的求值顺序

---》逻辑与（&&）、逻辑或（||）、条件运算符（?：）、逗号表达式（,）

（2）未定义求值顺序的表达式，当访问公共对象或执行IO任务时，会出现未定义的行为

8、作用域与隐藏（见p210）

（1）内层作用域中声明的名字，将隐藏外层作用域中声明的同名实体（函数或变量名）

（2）情况举例

---》在不同的作用域中无法重载函数，只会隐藏

---》子类成员隐藏父类同名成员（函数或变量名）

9、类的前置声明

（1）形式：

---》如：class A; // 前置声明

---》class关键字向编译器引入了名字A，并指明A是一种类型

（2）不完整类型（见p250）

---》对于类型A来说，在声明之后定义之前是一个不完整类型（只知道A是一个类类型，但是不知道有什么成员）

---》只能在有限的情景下使用：定义指向A的指针或引用；声明(不是定义)以A为参数或返回值的函数

---》不能使用A实例化一个对象，编译器不知道对象需要多少内存空间

---》任何类不允许包含自身类型的成员变量，因为编译类成员时，类还不完整

---》静态成员可以是不完全类型（类可以包含自身类型的静态成员，饿汉模式）

10、建议使用初始化列表初始化类成员

（1）何时必须使用：

---》基类没有缺省构造；

---》成员是const、引用等必须初始化的变量

（2）初始化顺序：

---》与成员在类中定义的顺序一致

---》建议构造函数初始化顺序与成员定义顺序相同

============================================

三、性能方面

1、vector对象能高效增长

---》C++标准要求vector应该能在运行时高效快速地添加元素，vector对象能高效地增长

---》定义vector对象时，没必要设定其大小；这么做性能可能更差；只有一种情况例外，所有元素值一样。

2、++i比i++性能更好

---》后置递增会拷贝一个副本，不是必须不使用

---》这一点对于对象类型尤其重要

3、移动语义和右值引用(std::move)

============================================

四、重要概念

1、左值和右值

---》可以取地址的值就是左值，左值通常具名；不可取地址的值就是右值，右值通常匿名。

---》《C++ Primer》p121,

2、求值顺序、优先级、结合律

（1）求值顺序举例

---》int i = f1() \* f2()； // 无法知道哪个函数先调用

---》cout << i <<“，”<< ++i << endl; // 结果未定义

（2）运算对象的求值顺序与优先级和结合律无关

（3）例如：表达式f()+g()\*h()+j()中

---》优先级规定：g()的返回值和h()的返回值相乘

---》结合律规定：f()的返回值先与g()\*h()的结果相加，再与j()的返回值相加

---》求值顺序：没有明确规定这些函数哪个先调用

（4）注意：如果f、g、h、j是无关函数，它们既不会访问公共对象也不执行IO任务，那么函数的调用顺序不受限制；反之，会出现未定义的行为。

3、面向对象编程（OOP）

（1）特点 -- 抽象、封装、继承、多态

---》封装：避免恶意修改带来的安全问题；代码复用；减少程序的耦合度；任意的改变存储数据的方式，不用调整用户代码

---》继承：复用代码；结构层次清晰；为了实现多态

---》多态：通过重写；覆盖提升代码可扩展性

（2）与面向过程编程的区别

---》OOP强调的是在运行阶段（而不是编译阶段）进行决策。

---》更具灵活性，可以根据当时情况进行调整。

---》比如：vector比起数组可以动态调整空间，不用提前指定数组大小

---》比如：虚函数的动态绑定，运行时决定调用哪个子类虚函数

4、类

（1）类的基本思想是数据抽象和封装。

---》数据抽象：是一种依赖于接口和实现分离的编程技术和设计技术。

---》封装后的类隐藏了它的实现细节，只提供调用接口。

（2）成员定义与编译顺序

---》编译顺序：编译器首先编译成员变量，然后编译成员函数。

---》内部定义：成员函数是隐式的inline函数。（见p230）

---》外部定义：成员函数可以忽略编写顺序，而进行嵌套调用

---》函数声明和定义的分离的优点与之类似：函数可以忽略编写顺序进行嵌套调用

5、善于代码复用

（1）代码复用的好处

---》函数代码量可能很大，重复下大量代码不太现实

---》修改维护时，只要修改一处地方

（2）如何代码复用

---》基于常用功能的封装函数

---》基于常函数的重载，常版本可以对this指针去常转换，然后调用非常版本

==============================================

五、使用C++的建议

1、初始化不是赋值（有关于const和列表初始化的概念）

---》初始化：创建变量时赋予其一个初始值

---》赋值：擦除对象的当前值，用一个新值代替

2、建议使用C++版本的C标准库头文件

---》在名为cname的头文件中定义的名字从属于命名空间std，而定义在名为.h的头文件中的则不然

3、尽量使用vector代替数组，string代替C字符串

---》定义指针及数组时，容易出错；使用时也容易出错

4、在一条语句中混用解引用和递增运算符（建议使用简洁的写法）

---》熟悉这种写法：\*i++ （见p132）

---》等价于这两种写法：\*(i++)、先\*i; 后++i;

---》这种用法基于，后置递增运算符返回还未加1的值

5、避免强制类型转换，尽量使用其他方法实现相同目的（尤其是重解释）

6、建议用引用代替指针

7、尽量加上const关键字

---》包括：返回值、形参、this指针

---》目的：（1）防止意外修改；（2）可以同时兼容常量参数和非常参数

8、函数的封装（见p213）

（1）优点

---》通过函数名提高可读性

---》每次相关操作都可以保证按照同样的方式进行

---》代码复用性好，且便于修改

（2）缺点

---》调用函数一般比求等价表达式要慢一些

---》一些短的函数可以内联优化

（3）函数的调用过程

---》调用前要先保存寄存器，并在返回时恢复

---》可能需要拷贝实参

---》程序转向一个新的位置继续执行（跳转）

9、内联函数和constexpr函数应放在头文件中

10、

============================================

六、了解

1、标准库定义了4个IO对象，cin、cout、cerr、clog

---》cerr用来输出警告、错误信息

---》clog用来输出程序运行时的一般信息（日志）

2、一般来说，类型float和double分别有7和16个有效位

3、顶层const：修饰对象本身的const

---》const int；int \*const

4、string类的相关函数

---》重载了输入输出操作符

---》s.empty()：判断是否为空

---》s.size()：返回s中字符个数（string::size\_type类型，无符号整型）

---》使用getline读取一整行：p78

5、vector的5种初始化方式

std::vector<int> v1(3, 1); // 1,1,1

std::vector<int> v2(3); // 0,0,0

std::vector<int> v3{ 1,2,3 }; // 1,2,3

std::vector<int> v4 = { 1,2,3 };// 1,2,3

// 数组int\_arr中的一部分，初始化vector

int int\_arr[4] = { 1,2,3,4 };

// 使用std::begin，std::end计算数组首尾地址

std::vector<int> v5(std::begin(int\_arr) + 1, std::end(int\_arr));

6、s.c\_str()的浅拷贝

---》该函数只返回，字符串首地址；如要一直，最好深拷贝一份

7、位运算符

---》包括：~、&、^、|、<<、>>，详见p135

8、sizeof运算符的2种形式

---》sizeof（type）

---》sizeof expr

9、空语句（最简单的语句）

---》; // 空语句，一个分号

---》有些地方，语法上一条需要语句，但逻辑上不需要，此时应该使用空语句（见p154）

10、英文名词

---》主调函数：calling function

---》被调函数：called function

11、声明一个返回数组指针的函数的方法（见p205）

（1）类型别名：using、typedef

（2）int（\*foo(int)）[10];

---》类似于数组指针定义：int（\*p(int)）[10]

（3）尾置返回类型（C++11）

---》auto foo(int) -> int(\*)[10];

（4）使用decltype（C++11）

---》decltype(数组名) \*foo(int);

12、函数重载

（1）重载和const形参（见p208）

---》非指针和引用的const无法构成函数重载（顶层const）

---》常量指针和常量引用可以构成重载（底层const）

（2）重载匹配（见p217~220）

13、函数指针（见p221）

（1）注意

---》函数名作为值时，取地址与否是等价的

---》函数指针解引用与否是等价的

（2）函数指针形参与回调函数

（3）返回函数指针

---》四种方法（与上面11项类似）

14、友元声明值影响访问权限，并不是普通意义上的声明

---》即使友元函数定义在类里，外部也要函数声明

---》这条规定并不是所有编译器都强制的

15、名字查找过程（见p254）

（1）非成员名字查找

---》搜索当前作用域，只考虑在名字的使用之前出现的声明

---》没找到，继续查找外层作用域

---》如果最终没有找到匹配的声明，则程序报错

（2）类的成员函数，名字查找有所不同

---》因为类的定义分为2步处理：先编译所有成员变量，再编译成员函数

---》所以，可以使用类内定义的任何名字

（3）成员函数作用域中的名字查找

---》现在成员函数内查找该名字的声明，只考虑之前的

---》没找到，到类内继续查找，类中所有成员都可以被考虑

---》没找到，到成员函数定义之前的作用域内继续查找

16、静态成员

（1）定义静态成员变量（见p270）

---》静态成员不属于对象 => 不由构造函数初始化 => 必须在类外单独定义和初始化

---》定义const或constexpr修饰的静态成员，可以在类内初始化，但初始值必须是常量表达式

（2）静态成员能用于普通成员用不了的场景（见p271）

---》静态成员可以是不完全类型（类可以包含自身类型的静态成员，饿汉模式）

---》静态成员可以作为成员函数的缺省实参

17、explicit抑制构造函数定义的隐式转换（见p264）

18、聚合类和字面值常量类

---》聚合类：用户可以直接访问其成员，并具有特殊的初始化语法（见p266）

---》字面值常量类：不太懂有什么用（见p267）

19、默认初始化与值初始化（见p88和p408）

（1）默认情况下，动态分配的对象是默认初始化的

---》内置类型或组合类型的对象的值是未定义的

---》类类型对象将用默认构造函数进行初始化

（2）值初始化

---》容器元素初始化时，未提供初始化值，则由容器决定元素初始化的值

---》一般int为0，string为空字符串

============================================

七、表格（可查找）

1、用前缀或后缀指定字面值的类型（p37）

---》例：L’a’(wchar\_t)；3.14f(float)；125u(unsigned)

2、C++关键字（p43）

3、初始化String对象的方式（p76）

4、cctype头文件中的函数（p82）

---》包括一些判断及处理char型字符的函数

---》例如：isalnum、isdigit、isspace、toupper

5、运算符优先级表（p147）

---》从高到低

==============================================

备注;

1、重点：C++新增语法、warning、性能方面、重要概念、使用C++的建议

2、表格类知识点，不重要的可以只记录，用到时才查找

3、了解：适当看看，知道就可以

4、书中，不常用的部分或深入的部分可以简单看或跳过（记录与另一份笔记）