**C++ https://zh-google-styleguide.readthedocs.io/en/latest/google-cpp-styleguide/contents/风格指南 - 内容目录**

1. 每个C++程序必须包含且只能一个main()函数，函数的**定义**由函数返回值，函数名，参数列表，函数体构成。Main 函数的返回值必须是int 。
2. 标准库的iostream库，保护istream, ostream两个**基础类型，只能引用不能拷贝。同时定义了4个io对象。**

**cout, cin, cerr, clog。程序所运行的窗口与这些对象相关联，cin与程序正在运行的窗口，cout,cerr,clog属于同一个窗口。cin属于istream对象，cout属于ostream对象。**

1. **表达式由以一个或多个运算对象和一个或多个运算符构成，产生一个结果，最小的计算单元。cout << “jin” << endl;表明首先jin写入cout ，返回cout， endl写入cout，返回cout。endl是一个特殊值，含义是结束当前行，并将与设备关联的buffer中类容刷新到设备中。**
2. **注释类型：//， 当前行右侧内容被编译器忽视，包含任何文本。/\* \*/，多行内容，但是里面不能再有/\* \*/字符。**
3. 标准库头文件<>, 非标准库头文件””, 文件重定向。程序 <infile >outfile，将从infile件中读取输入，程序输出到outfile中。例如：ifconfig >> test.log 2>&1，将错误和输出均输出到test.log中，追加写。
4. **赋值，抹去当前对象的值，新值取代。类是C++基本特性，一种用于定义自己的数据结构及其相关操作的机制。类类型就是类名。内置类型就是语言定义的类型。{}表明程序块。C++标准库类型均在命名空间std中。未初始化的变量不可使用。()也是运算符，.也是运算符，左侧运算对象必须为类类型对象，右侧运算对象必须为其成员名字。**
5. **基本内置类型：字符型，整型，浮点型。内置类型的机器实现：可寻址的最小内存块为字节，存储的基本单位为字，字由4字节或者8字节构成=32或64位系统。数据类型决定了数据所占的bit数以及如何解释这些bit。字面值，每个字面值常量都对应一种数据类型。对象是具有某种数据类型的内存空间。数组类型的实现也与硬件密切相关。**
6. **对象初始化和赋值不一样。声明和定义区别，定义申请存储空间并赋予变量一个值。变量只能定义一次，声明多次。C++是一种静态类型语言，在编译阶段检查类型。对象的类型决定了对象所能参与的运算。变量命名规范，小写字母，自定义的类大写字母开头。变量是对象的具名。**

**如果定义变量时没有指定初值，则会被默认初始化。默认值由变量类型决定。如果内置类型的变量未被显示初始化，其值由定义的位置决定。定义于任何函数体之外的变量被初始化为0，定义在函数体内部的内置变量将不被初始化。每个类决定其初始化对象的方式也就是构造函数。string类规定如果没有指定初始值则生成一个空串。**

1. **变量作用域：外层作用域和内层作用域，{}分割作用域。内层作用域可以访问或覆盖外层作用域变量。**
2. **复合类型：基于内置类类定义的类型，引用和指针，引用是一种类型，不是对象。必须被初始化，初始化值必须是一个对象，不能是字面值，对象类型与引用类型必须严格匹配。指针是一个对象，也是一种复合类型。存放指向对象的地址。空指针，无效指针，含有有效值。void\*可存放任意类型对象的地址。int\* p, int \*p,两种写法含义一样。对象一定是一段内存，但一段内存不一定是对象，有构造函数和析构函数。复合类型可以是void 类型，也可以是内置类型。**
3. **const类型修饰符，const对象必须初始化。该变量的值不能改变。const对象仅在文件内有效。除非加extern关键字。一般引用的类型必须于对象的类型一致，但是const引用例外，其初始化值可以是非常量对象，字面值，以及表达式。const 引用无法改变其引用对象的值，但是如果该对象不是常量，可直接通过修改该对象的值。**
4. **const指针，类似const 引用， 必须初始化。因为指针是一个对象，所以可以有2种含义。指针指向对象是一个常量，指针的值可以改变，指针不是常量。无法通过指针去改变指向对象的值。指针本身是一个常量对象，其值不可改变，如果其指对象不是常量，则可修改其指向对象的值。例如：**

**int i=0;**

**int \*const p1= &i; p1是一个常量指针，其p1值不可改变。**

**const int ci = 41; c1的值不可改变，c1值不可改变**

**const int \*p2 = &c1；p2的值可改变。**

**const int \*const p3 = p2; p3的值不可改变**

**const int &r =ci; r不可改变ci的值。**

**const默认修饰左边的部分，如果左边没有，则修饰右边的部分。**

**p1是一个常量指针， 指向int型。**

**ci是一个常量int型**

**p2是一个指针，指向一个常量int**

**p3是一个常量指针，指向一个常量int**

**r是一个引用，引用的是一个常量int。**

1. **直接初始化：不使用赋值号=的初始化形式，拷贝初始化使用=的形式。a->b等价于(\*a).b**
2. **重载运算符时，运算对象的个数，运算符的优先级以及结合律均无法改变。C++左值和右值的区别：能否使用&取对象的地址，以及是否是临时对象还是持久化对象。**
3. **空语句 ; 复合语句=块，一个块也就是一个作用域, {}. 空块等价于空语句。**
4. **当程序的某部分检测到一个它无法处理的问题时，需要用到异常处理。一般少用，如果真的异常，程序直接退出即可。**
5. **函数执行第一步是定义并初始化其参数列表， 参数类型以及个数需要与定义函数时一致。函数的返回类型可以是指针，不能是函数类型。函数参数及其内部定义的变量均为局部变量。局部变量定义本身不含初始值，则执行默认初始化，此时内置类型的未初始化局部变量将产生未定义的值。**
6. **局部静态变量，直到程序终止时，才被销毁。函数可以声明多次，只能定义一次。每次调用函数时均会创建其形参，并用实参对形参初始化。形参是引用时，则是引用传递，若是拷贝实参的值给形参，则是值传递。引用形参避免拷贝，不改变形参的值，则使用const。使用引用形参返回额外信息。实参初始化形参时会忽略掉形参的顶层const。因为初始化const 类型时，初始化对象可以是const对象也可以不是。使用引用而非常量引用也会极大的限制函数所能接受的实参类型。**
7. **无返回值的return语句只能用在返回类型是void的函数中。返回的值用于初始化调用点的一个临时变量。函数重载：同一作用域内，函数名相同，但是形参列表不同。主要指形参个数以及形参类型区别。函数匹配：选择重载函数集，与实参匹配的可行函数，形参与实参数量相同，形参类型与实参类型相同，最佳匹配，需要实参类型转换来判断。二义性调用是指找不到唯一的最佳匹配。**
8. **当编译器调用函数时，会在当前作用域寻找函数的声明，然后检测函数是否可调用。函数的默认实参，调用函数时可以包含该实参，也可以省略。一旦某个形参被赋予了默认值，后面的所有形参都必须要有默认值。**
9. **函数指针，函数不是对象，函数指针指向是函数，**

**bool (\*pf) (int a, int b); 返回值是一个bool**

**pf = length; pf指向函数名未length的函数**

**bool b2= (\*pf) (1,2); 调用该函数。**

1. **类的成员函数，当调用成员函数时，实际上是替某个对象调用它，将total的地址传递给****隐式形参this。const成员函数，const的作用是修改隐式形参this指针的类型，默认情况下，this是指向类类型非常量的常量指针，即this的值不能改变，但是其指向对象的值可以改变。如果我们不允许改变对象，需要将this设置为指向常量的指针，C++允许将const关键字放在成员函数的参数列表后，表示this指向的是一个常量对象。因此常量对象，以及常量对象的引用或指针都只能调用常量成员函数。**
2. **当定义的函数类似于某个类似内置运算符时，应该该函数的行为模仿该运算符，左值返回，则返回类型应该是一个引用。与类相关的非成员函数，属于类的接口，但不属于类本身，与类声明定义在同一个头文件中。**
3. **构造函数无法声明为const，因为只有构造函数完成初始化过程，对象才能具有常量属性。默认构造函数：存在类内的初始值，则用其来初始化成员，否则默认初始化。内置类型或复合类型默认初始化为未定义。尽量定义构造函数。没有出现在构造函数初始值列表中的成员将通过类内初始值或者执行默认初始化。**

**Sales\_data() = default;**

**Sales\_data() { }; 不是pod类型，编译器无法对其进行优化。**

**以上两个都是默认构造函数。**

**除了初始化构造函数外，还有拷贝，赋值和析构函数，编译器也会自动合成。管理动态内存的类通常无法依赖以上合成函数。可变数据成员mutable，即使是const对象的成员，也可以被改变。类的编译，是首先编译成员的声明，才会编译类函数体。**

1. 类的静态成员，与类本身直接相关，而与类的各个对象无关。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数据成员相关的数据，静态成员不与任何对象绑定一起，不包含this指针，静态成员函数不能是const。必须在类的外部定义和初始化类的静态数据成员。存在于整个程序的声明周期中。
2. IO库，分为从流读写数据，输入流，输出流，错误流。从文件读写数据，以及从string读写数据。iostream, fstream, sstream,。ifstream 从给定文件读取数据，ofstream从给定文件写入数据，fstream可以读写文件。
3. 容器是特定类型对象的集合。顺序容器：vector, deque, list, forward\_list, array, string。

对于string, vector，将元素保存在连续的内存空间中，当添加一个元素时可能需要分配额外的内存空间，此时每个元素均需要移动到新的内存空间中。vector重点推荐.容器有赋值，swap，删除增加元素，遍历元素。容器相等比较，容器的元素是copy，不是对象本身。

1. 迭代器是一种检查容器内元素并遍历容器元素的数据类型，每种容器类型都定义了自己的迭代器类型。

vector<int>::iterator iter;

vector<int>::const\_iterator iter; 常量迭代器，无法修改对象内容。

使用\*符来访问对象。vector支持顺序遍历，也支持反向遍历，

v.rbegin(); v.rend(); 反向迭代器，迭代器支持++, --。

容器操作可能导致迭代器失效，向容器中添加或删除元素。所以在添加或删除元素时，注意更新迭代器。不要缓存end()返回的迭代器。

1. 泛型算法，通用算法，可作用于不同类型的容器和不同类型的元素。在algorithm或者numertic中，这些算法并不直接操作容器，而是遍历由两个迭代器指定的一个元素范围来进行操作。算法可能改变容器中元素的值，或者在容器内移动元素，但永远不会直接添加或删除元素。
2. 标准库提供了超过100个算法，分为只读算法，改变元素或者重排元素顺序。

只读算法：find, accumulate, equal,

写容器元素算法：fill,

重排：sort，unique，sort(words.begin(), words.end(), isShorter)后面是函数名，这个函数可以通过lamba表达式来代替：

[] () {}: 捕获列表，参数列表，函数体。捕获列表是局部变量的列表。

size\_t v1 =42;

auto f = [v1] { return v1;}

auto j = f()；j = 42

1. 关联容器，map, set, multimap, multiset, unordered\_map,。每个关联容器都自定义了一个默认构造函数，创建一个指定类型的空容器。关联容器对关键字的类型有一些限制，关键字的类型：具有<运算符。字典序升序排列，关联容器不使用泛型算法，关联容器的元素类型为pair，无序关联容器，使用哈希函数来组织元素。将元素映射为一个桶，将具有相同哈希值的元素放在同一个桶中。

c.bucket\_count(): 正在使用的桶的数目

c.max\_bucket\_count(): 最多桶的数目

c.bucket\_size(n):第n个桶中元素个数

1. 动态内存：全局对象，程序启动时分配，程序结束时销毁。对于局部自动对象，进入块时创建，离开块时销毁。局部静态对象第一次使用时分配，程序结束时销毁。静态内存和栈内存。堆用于分配动态分配内存。new delete, 运算符管理动态对象。

shared\_ptr<string> p1 = make\_shared<string>(“jin”); 允许多个指针指向同一个对象。

unique\_ptr<string> p1; 独占所指向的对象

智能指针默认初始化为空指针，当指向一个对象的最后一个shared\_ptr被销毁时，其会自动销毁此对象。

使用动态内存的原因：

程序不知道自己需要多少对象：容器类

程序不知道所需对象的类型：多态

程序需要在多个对象间共享数据：原对象以及其拷贝都引用相同的底层元素，此时某个对象被销毁时，不能单方面的销毁底层数据。

默认情况下：动态分配的对象是默认初始化的。delete的指针必须是由new运算符分配的指针。

内存泄漏：忘记释放内存，delete

使用已经释放的对象

同一块内存释放2次