* 当函数重载时，会优先调用类型匹配的函数实现， 否则才会进行类型转换
* auto \*x = new auto(3); //x是int\*
* 初始化列表的成员是按照声明的顺序初始化的，而不是按照出现在初始化列表中的顺序
* 在构造函数的初始化列表中，还可以调用其他构造函数，称为“委派构造函数”
* 就地初始化（成员变量的默认值）
* 基类构造🡪成员构造🡪自身构造
* A a[3] = {A(1, 2), A(3, 5), A(0, 7)};
* 全局对象的构造顺序不能完全确定，所以全局对象之间不能有依赖关系，否则会出现问题
* 哑元：后缀
* 友元的声明与当前所在域是否为private或public无关
* 友元函数可以在类内定义
* 静态数据成员应该在.h文件中声明，在.cpp文件中定义。
* 常量数据成员不允许在构造函数的函数体中通过赋值来设置，可以就地初始化或用初始化列表来初始化
* 不存在静态常量函数
* 右值引用不能绑定左值
* 所有引用本身都是左值
* 不发生返回值优化时：将局部对象传给“返回值”，析构局部对象；将“返回值”传给调用出待赋值对象，析构“返回值”
* 返回值优化：返回局部对象（左值）时：完全不发生移动或拷贝
* 转换运算符（无需定义返回值，参数列表为空）或构造函数都可以实现类型转换（不能同时使用）可使用expli禁止自动类型转换
* 构造函数，析构函数，赋值运算符不会被继承
* Using：继承构造函数；打开对私有继承中基类成员的权限；恢复被重写隐藏的基类成员函数
* 虚函数表的构建在编译期发生；虚指针在构造函数中创建
* 构造函数、析构函数调用虚函数时会调用“本地版本”
* 重写覆盖必须函数名和参数列表都相同；重写覆盖也会隐藏基类中其他同名函数
* 重写隐藏不会覆盖虚函数表（可以理解为向上类型转换后还可调用原版本，且可以用using恢复）
* Final：对于虚函数，禁止重写；对于类，禁止继承
* 纯虚函数也可以实现，并在子类中被显式调用；纯虚析构函数必须被实现
* 模版参数必须在编译期确定
* 静态：作用域仅限于文件
* Vector中使用erase删除元素：被删元素及其后元素迭代器失效
* Map和set会对其中元素排序
* 函数模版不能部分特化（可用重载函数来达到此效果）
* 优先匹配特化版本，前提是被特化的对应基础函数模板被匹配到。（<T>,<T\*>,<char\*>）;即全特化版本优先级可能低于某些特殊基础模版
* 类模版可以半特化
* 读入空格：getline函数；指定分隔符则还可以读入换行符
* Stringstream的str函数会返回其中所有内容，而不是未读取部分
* 对函数指针和函数对象的统一：<functional>中的function<returnTypr(arg list)>模版
* 不能使多个智能指针指向同一个裸指针；不能直接用智能指针维护数组
* <memory> shared\_ptr,weak\_ptr(有expired函数),unique\_ptr
* 迭代器模式：基于继承的设计（算法中需要使用迭代器基类的指针）和基于模版的设计
* 适配器模式：利用组合：对象适配器模式；利用继承（多重继承）：类适配器模式

正则表达式：

原生字符串：R”(str)”

元字符：. \w \s \d ^ $ \b(单词的开始或结束)

重复模式：\* + ？ { , }

分组：（）(信息储存到smatch类型对象中)(sm有suffix（）.str()函数)

regex\_match

regex\_search

regex\_replace($&,$1,$2……)