

# static关键字作用

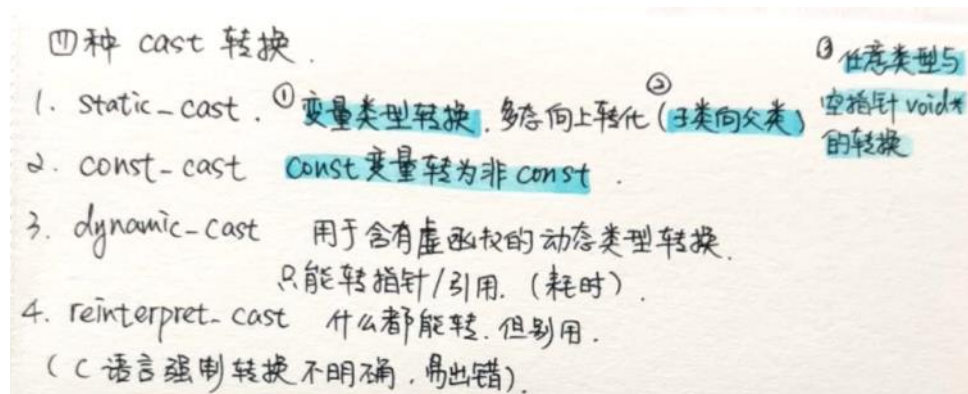
2019年7月8日 星期一 13:53

Static 关键字作用 (隐藏) (只在本文件中可见)

1. 静态变量  $\left\{ \begin{array}{l} \text{局部} \\ \text{全局} \end{array} \right.$  , 静态储存区  $\rightarrow$  反复调用多次也只初始化一次
2. 静态函数  $\rightarrow$  只能这样调用
3. 类的静态成员及静态函数  $\langle \text{类名} \rangle :: \langle \text{静态成员名} \rangle$  (this 不可使用)  
类的所有对象通用, 属于类不属于对象

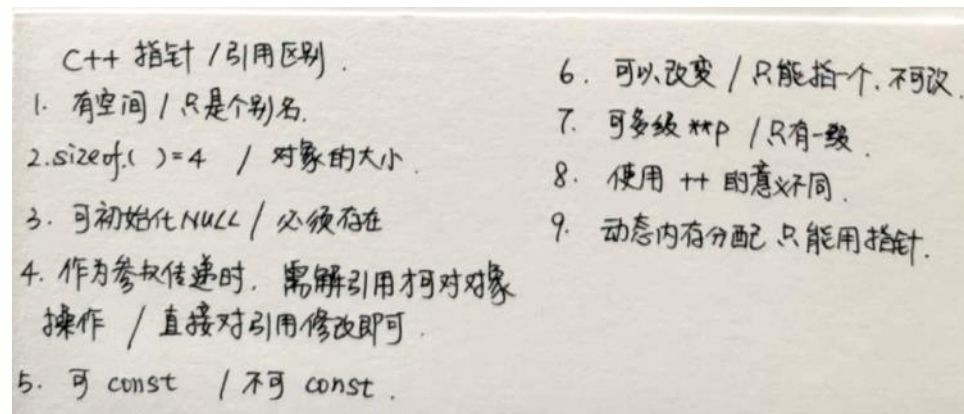
# 四种cast转换

2019年7月8日 星期一 13:55



# 指针的区别

2019年7月8日 星期一 13:55

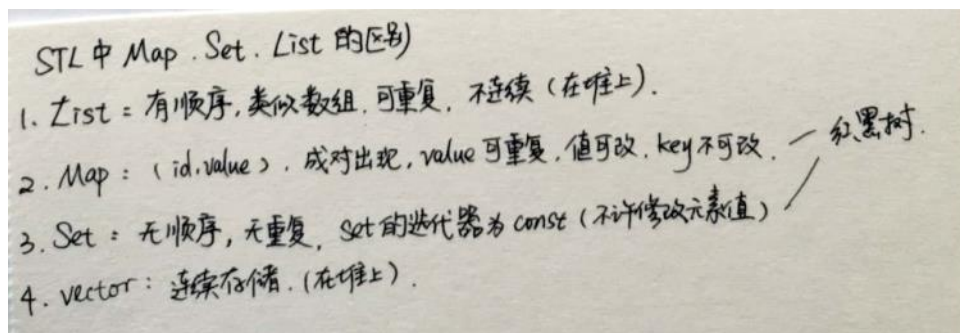


# STL相关

2019年7月8日 星期一 13:57

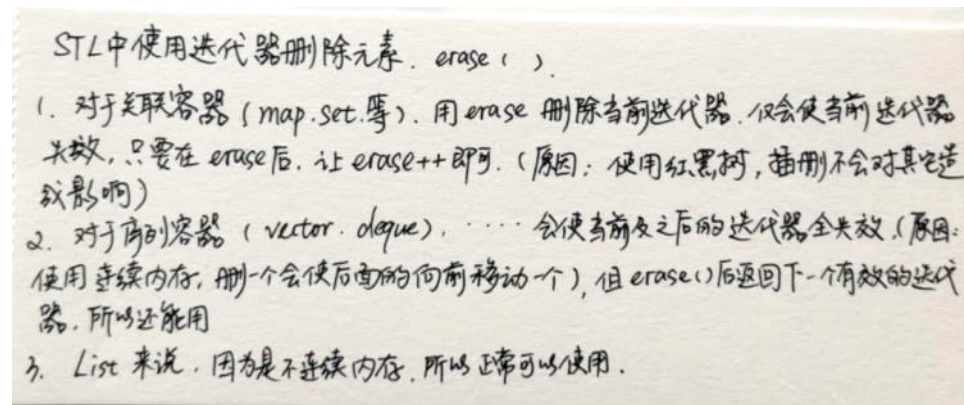
# Map Set List区别

2019年7月8日 星期一 14:00



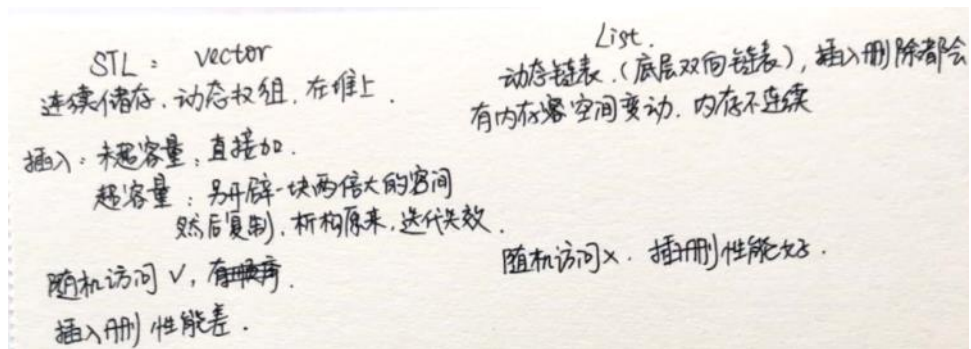
# STL中使用迭代器删除元素

2019年7月8日 星期一 13:55



# STL中vector和List区别

2019年7月8日 星期一 13:56



# STL中map, unordered\_map区别

2019年7月8日 星期一 13:57

STL中map和unordered\_map.

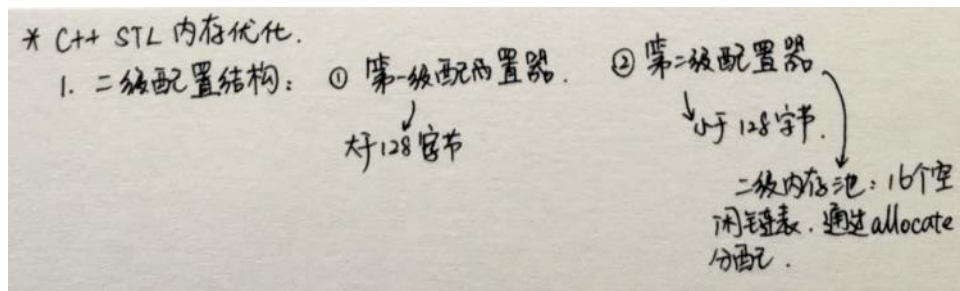
map: 用红黑树实现.  
按从小到大排序 (中序遍历).  
~~插入, 删除效率高.  $O(\lg n)$ .~~  $O(\lg n)$ .  
缺点: 每个结点, 占用内存较大 (父子, 指针).

unordered\_map: 用哈希表实现.  
无序.  
查找效率高.  $O(1)$ .  
建立哈希表耗时.



# STL内存优化

2019年7月8日 星期一 14:06



# 迭代器，指针的区别

2019年7月8日 星期一 13:56

迭代器与指针的区别。

✓ 类模板，不是指针但像。提供一种方法顺序访问聚合对象中的各个元素，同时不暴露内部。重载了指针的操作符，相当于一种智能指针。根据类型不同操作不同。

✓ 存在原因。迭代器返回对象引用而不是其本身。

# resize和reserve区别

2019年7月8日 星期一 23:36

resize 和 reserve 区别.  
✓ 改变当前容器内元素数量. 多出来的初始化为0, 后面 push-back 会从 new+1 开始算.  
... 的最大容量, 开辟块新空间, copy 来, 然后销毁以前份.

# 判断点是否在三角形内或边上

2019年7月8日 星期一 13:58

Q: 给定一点, 是否在三角形内/边上.  
A:  $S_{ABC} = S_{ABP} + S_{ACP} + S_{BCP}$ .

# 四种智能指针

2019年7月8日 星期一 13:58

Q: 给定一点, 是否在三角形内/边上.

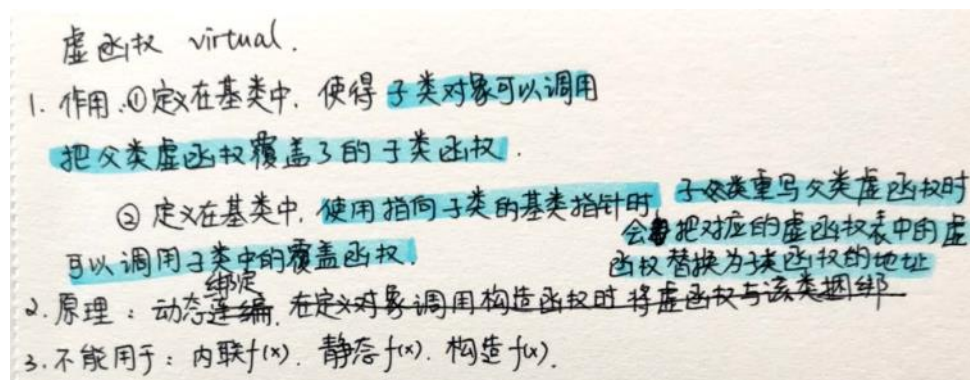
A:  $S_{ABC} = S_{ABP} + S_{ACP} + S_{BCP}$ . → 在堆上分配的内存.

**智能指针:** 就是一个类, 会在作用域外自动析构, 不会有内存泄漏.

1. **auto\_ptr**. (C++11 已不支持).  
    ↑   ↑  
    p2 = p1; // 之后访问 p1 会出错, 所以有崩溃问题. 临时右值.
2. **unique\_ptr** (代替 auto\_ptr). 但如果 p2 = new unique\_ptr(new ...);  
    ↑   ↑  
    p2 = p1; // 非法, 所以没有崩溃问题. 就可以通过.
3. **share\_ptr** (多个智能指针指向相同对象, 并计数)  
    成员: use-count, get 等.
4. **weak\_ptr** (为了解决两个 share\_ptr 相互引用造成的死锁问题).  
    弱引用. 只能用 share\_ptr, weak\_ptr 来赋值. 不算作计数.

# 虚函数

2019年7月8日 星期一 13:58



# 求一个数中1的个数

2019年7月8日 星期一 13:59

求一个数中1的个数。

1. 逐位除10取余，计数。

# 为什么虚构造函数是虚函数

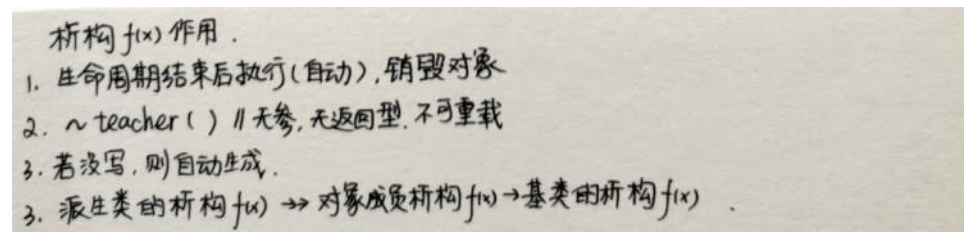
2019年7月8日 星期一 13:59

Why 析构 f(x) 为虚 f(x)?  
将父类析构 f(x) 设为虚, 则当 new 子类时, 用基类指针指向时, 释放该指针可以  
释放子类空间, 防止内存泄露。  
Why 默认析构 f(x) 不为虚?  
虚 f(x) 需要虚函数表和虚表指针会占用额外内存。



# 析构函数作用

2019年7月8日 星期一 13:59



# strlen和strcpy的区别

2019年7月8日 星期一 23:35

strlen & strcpy  
1. strcpy // 从第2个参数拷贝至第1个参数, 不指定长度, 易越界. strcpy 更安全.  
2. strlen // 返回①到"\\0"的字符串个数.

# 多态

2019年7月8日 星期一 13:59

## 多态

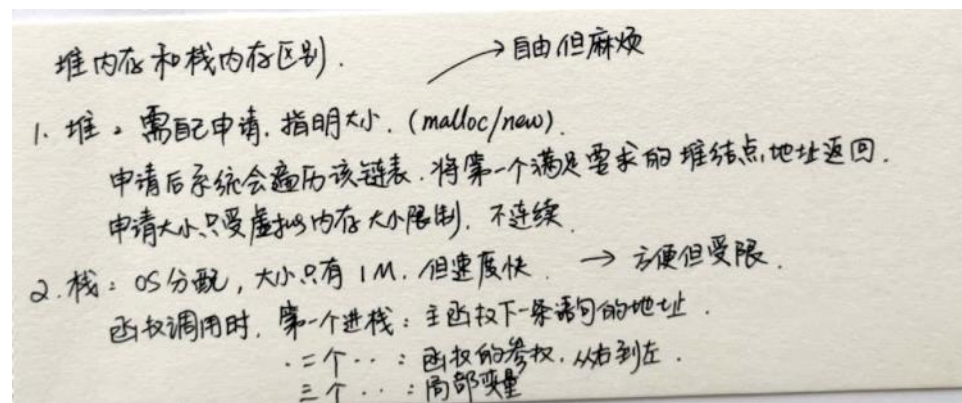
1. 静态多态：重载，在编译的时候就已确定。
2. 动态多态：虚函数。

## 常量

1. 顶层 const + 类型，必须初始化。
2. 局部对象：常量存在栈区，  
全局对象：常量存在全局/静态存储区。

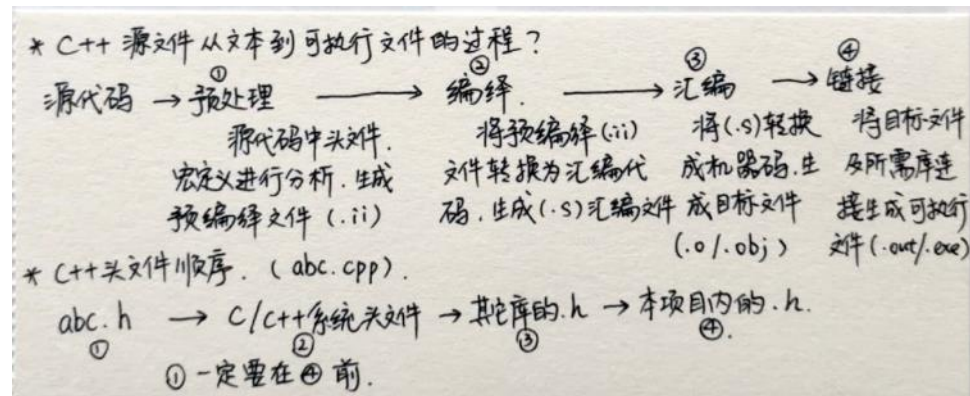
# 堆内存和栈内存的区别

2019年7月8日 星期一 14:00



# C++源文件从文本到可执行文件的过程

2019年7月8日 星期一 14:04



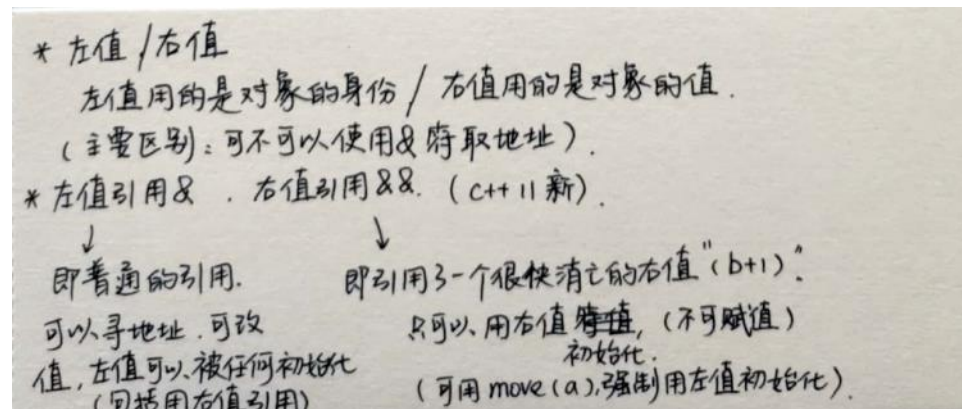
# C++头文件处理顺序

2019年7月8日 星期一 14:04

\* C++头文件顺序. (abc.cpp).  
abc.h ① → C/C++系统头文件 ② → 其它库的.h ③ → 本项目内的.h. ④.  
①一定要在④前.

# 左值引用和右值引用

2019年7月8日 星期一 14:05

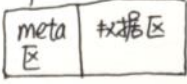


# malloc原理

2019年7月8日 星期一 14:05

## \* malloc() 原理

用于动态内存分配。

从堆区中分配内存。将堆区分为多个块。每个块  的大小根据物理内存大小不同而不同。

当块被分配出去时，将当前块的堆结点，从空闲结点链表中删除。  
使用完后需要 free()。



# new和malloc区别

2019年7月8日 星期一 14:05

\* new和malloc区别.

1. 申请的内存位置不同: new从自由存储区分配, malloc从堆上分配.  
是c++基于new的一个抽象概念, 凡是用new申请的内存都叫这个名字. ↓ 是操作系统所维护的一块特殊内存, 不是一个概念.
2. 属性不同: new是c++关键字, malloc是库函数, 需头文件.
3. 返回类型: new分配成功返回对象类型的指针, 与对象匹配故安全.  
malloc成功返回 `void*`, 需强制转换成所需类型.

4. 分配失败: new抛出 `bad_alloc` 异常, malloc返回 `NULL`.

5. 是否调用构造/析构函数: ① new先调用 `operator new` 函数, 申请内存.  
② 编译器运行相对应类型的构造函数, 初始化. ③ 返回指向对象的指针  
④ delete时先调用析构函数, 后调用 `operator delete` 函数释放空间.  
① 而malloc/free是库函数, 没法调用自定义类型的构/析函数, 只负责分配内存.
6. 重载: new/delete可以重载, malloc不可.
7. 参数: new无需指定内存块大小, 编译器配算.  
malloc必需.