C++基础“

1. 对于函数而言，think()为函数的调用，而think为该函数的地址。

函数指针 int (\*pm)(int a);

Int pam(int a);

Pm = pam;

必须保证 函数指针 与 函数的返回值类型相同

1. 指向常量的指针，指针常量，指向常量的指针常量 区分。 左定值，右定向

Const int \*ptr = &a;

Int\* const ptr = &a;

1. 指针 数组 指针算数

数组名称为指针 数组[i]为元素 &数组[i]为地址

Int num[3] = {1,2,3}; int\* ptr=num; int \*ptr2=&(num+1);

\*(num+1) 输出为 2

1. C++ define 宏定义 为编程过程中提供一定的方便

执行过程中没有拷贝 预处理阶段展开 没有类型 不分配内存

1. &代表位运算 创建引用变量 &&并 与 短路功能

1&2=0 1(01) 2(10) 位运算 0

1. \*pf(int) 是返回指针的函数，(\*pf)(int)是函数指针

Int\* ok[20] 指针数组：一个有20个指向int类型得指针的数组

Int (\*ok)[20] 数组指针：ok是指向用20个int值得数组的指针

Int (\*(\*func)[5])

1. 双引号与单引号的区别在于：字符串常量和字符常量的区别。

字符常量可以用于整数运算 ‘a’=1

字符串常量 赋值方法 const char\* s = 字符串常量；

1. 内联函数为什么执行速度快：执行过程中，不存在内存上的跳跃。

内联函数在编译时 将内涵函数的代码插入到调用该函数的地方

1. 内联函数的声明：在函数声明与函数定义处添加 inline关键字。
2. 区分引用变量与指针之间的区别：引用在声明时必须初始化。

Python中不区分传值和传地址的操作，根据变量类型判断 是传入值还是传入地址。

Int& a = name;

Int\* a = &name;

1. 什么叫做函数重载（函数多态），允许多个同名的函数存在。

类的多态性： 允许声明为基类的指针 指向 派生类

1. 函数模板template，允许函数声明过程中不指定数据类型，需要提前创建模板。
2. 运算符优先级：！优于 关系运算符 优于 && || 因此if（！（x>5））
3. ?:运算符 s=x>3?10:20
4. Continue 跳过循环中余下的代码 Break 跳出循环
5. 作用域解析符号 ：： 用于类的方法定义 还可以用定义派生类

Class supervip::public vip{}

1. 间接引用符 . 用于类的方法的调用（引用，）
2. 执行结构体成员的指针 -> 用于结构体的成员调用（指针）
3. 构造函数与析构函数的区别

由于类无法像 结构 式的命名规则，因为类的数据项是私有的（数据隐藏）

注意：类和结构都可以使用 结构初始化方法

Stuct \*TreeNode{

Int val;

TreeNode \*left;

TreeNode \*right;

TreeNode(0):val(0),left(nullptr),right(nullptr){}

}

构造函数：没有返回类型，不声明为void，声明为与类同名函数Stock：：Stock

派生类的构造函数 首先调用 基类的构造函数

Thing amabob={“wodegt”,-23} 结构初始化 亦可能是 构造函数初始化 匹配

Stock gupiao（“furrt”，23，25）利用构造函数初始化

构造函数不能被 对象所调用

析构函数：跟踪对象到过期 释放内存空间 当构造函数利用new 析构函数 用delete

~Stock（）没有参数

1. const Stock huangjin（）仅能调用函数声明时 赋予const关键字的方法

void name() const; 可以调用类中这一类方法

1. 在 C++ 中，每一个对象都能通过 this 指针来访问自己的地址。this 指针是所有成员函数的隐含参数。因此，在成员函数内部，它可以用来指向调用对象。
2. Stack 栈

Class stack(){

Enum{MAX=10};

ITEM item[MAX];

Int top;

Publish:

Stack();

Void pop(const Item& item);

Void push(Item& item);

Bool isempty() const;

Bool isfull() const;

}

1. 友元函数是定义在类外部,函数头位于类中 friend开头，但有权访问类的所有私有（private）成员和保护（protected）成员

通常与重载运算符相互搭配。

成员函数运算重载符：Time1+time2 time1.operator+(time2)

友元函数 ：Time1+time2 operator+(time1,time2)

子类可以访问 父类保护成员 不能访问父类的私有成员

1. volatile提醒编译器它后面所定义的变量随时都有可能改变 用于函数定义。
2. 函数定义不可以嵌套 但是函数调用可以嵌套
3. ++b 先执行+1，b++ 后执行+1
4. X=6；

X+=x-=x\*x;

最后x=-60，从后往前计算

1. 字符数组: char a[10]

指针数组：char\* a[10] 区分 int\* a a是一个指针

指向字符数组的指针：char (\*a)[10]

Typedef char T[10] 定义T为字符数组

T \*a //a为指向T这种数据类型的指针

A的类型 char (\*a)[10]

1. 在给指针赋值时，其赋予的值应该为一个地址

Int \* ptr=&(地址)

或者

Ptr=&(地址)

Ptr=new double[50]

必须申请内存空间

1. 命名规则：必须以字母和下划线为开头
2. 反斜杠\开头代表转义字符，默认为八进制。\72
3. 在 类声明 中定义的函数 将自动定义为内联函数
4. long long占8个字节

double 占八个字节

short 占两个字节

1. 函数重载的判断依据：参数个数、参数类型、Const，返回类型 不行
2. Static的作用

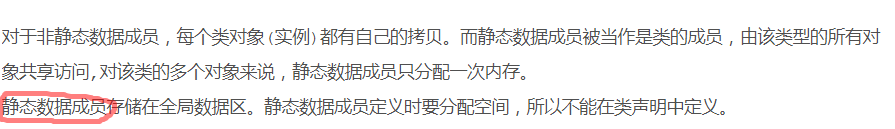
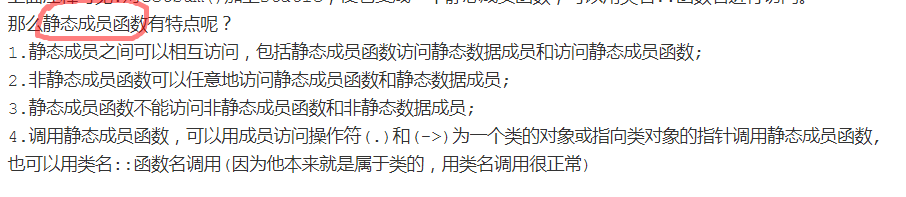
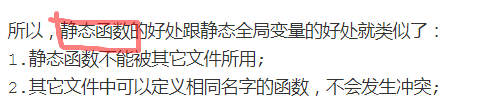
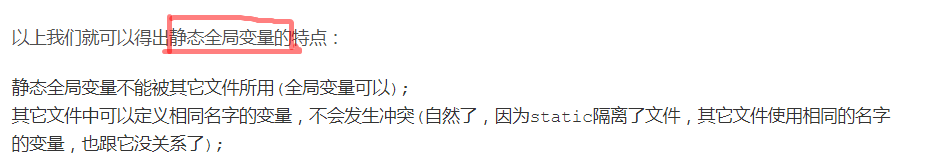
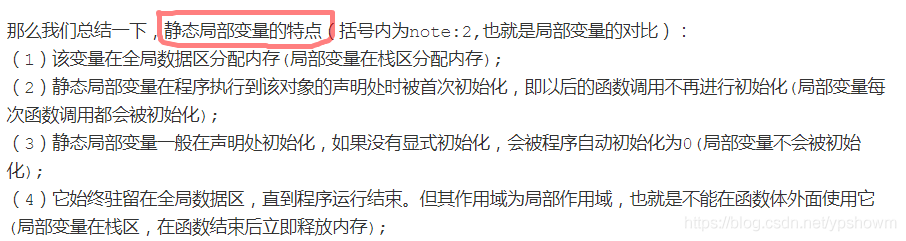
1）：修饰全局变量时，表明一个全局变量只对定义在同一文件中的函数可见。

2）：修饰局部变量时，表明该变量的值不会因为函数终止而丢失。

3）：修饰函数时，表明该函数只在同一文件中调用。

4）：修饰类的数据成员，表明对该类所有对象这个数据成员都只有一个实例。即该实例归 所有对象共有。

5）：用static修饰不访问非静态数据成员的类成员函数。这意味着一个静态成员函数只能访问它的参数、类的静态数据成员和全局变量



1. 特殊的成员函数

默认构造函数、默认析构函数、复制构造函数、赋值运算符、地址运算符

1. 字符串变量和字符串常量之间的区别

Char s[6] = “Hello” s为字符串 “Hello”字符串常量

Const char\* s = “Hello”;

String s = “Hello”;

可以看出 string 内部的构造函数 就是维护了一个const char\*

1. 初始化方法

Int \*a = &num;

Int &a = num;

注意区分 初始化 与 赋值之间的区别

1. 内区区域分配情况



代码区：存放程序的代码，即CPU执行的机器指令，并且是只读的。

常量区：存放常量(程序在运行的期间不能够被改变的量，例如: 10，字符串常量”abcde”， 数组的名字等)

静态区（全局区）：静态变量和全局变量的存储区域是一起的，一旦静态区的内存被分配, 静态区的内存直到程序全部结束之后才会被释放

堆区：由程序员调用malloc()函数来主动申请的，需使用free()函数来释放内存，若申请了堆区内存，之后忘记释放内存，很容易造成内存泄漏

栈区：存放函数内的局部变量，形参和函数返回值。栈区之中的数据的作用范围过了之后，系统就会回收自动管理栈区的内存(分配内存 , 回收内存),不需要开发人员来手动管理。栈区就像是一家客栈，里面有很多房间，客人来了之后自动分配房间，房间里的客人可以变动，是一种动态的数据变动。

1. 基类的析构函数 要定义为 虚函数

按 引用 或者 按 指针类型删除 基类指针指向派生类时 会调用基类的析构函数 派生类new申请的空间没有得到释放 导致内存泄漏

1. 虚函数 与 纯虚函数之间的区别

声明方式不同 纯虚函数在后面 =0

定义方式 纯虚函数不需要定义实现方法

有纯虚函数的类 成为 抽象类 不能生成对象