一、C++入门

1、C++中关键字----C++98-----63个关键字

2、命名空间

>>什么是命名空间：一个作用域　一个部门----->一个公司

>>命名空间的作用：防止命名冲突

>>如何定义一个命名空间：namespace N1{变量、函数}

>>命名空间中的成员如何使用

a、N1::成员名字

b、using N1::b;

c、using namespace N1;

二、缺省参数----备胎

1、概念：如果在定义函数时，给函数的参数带上一个默认值，在使用时可以传参，也可以不传参，如果传参，使用用户传递的实参，如果用户没有传递，则使用默认值

2、分类：半缺省参数和全缺省参数

半缺省参数：部分参数带有默认值，注意：只能从右往左一次给出

```
void TestFunc(int a,int b=2,int c=1){}
TestFunc(1,2);
```

全缺省参数：所有参数带有默认值

```
void TestFunc(int a=1,int b=2,int c=3){}
```

3、注意

1>默认参数设置规则：必须从右往左一次给出

2>不能在声明和定义时同时给出-----男女朋友关系：1对1　1对多肯定不行

3>默认参数提供方式----常量、全局变量

4>C语言不支持

三、函数重载（重要程度****）

1、概念：在同一个作用域，如果存在多个相同名称的函数，并且参数列表不同（参数个数、参数类型、类型次序）与返回值类型是否相同无关

生活中：一词多义

```
int Add(int a,int b);          double Add(double a,double b);
```

2、调用原理

Add(1,2);//编译阶段，编译器会推演函数实参的类型，int，int，根据推演的结果选择合适的函数进行　调用---如果没有合适的函数||有多个函数可供选择，编译器会报错

Add(1.0,2.0);

Add(1,2,3);//int,int,int 编译报错

Add(1,2.0);//int double----->尝试进行类型转化：a、int，int　b、double，double 此时编译器就不知道到底应该选择哪一个而引起二义性

3、C+中为什么可以支持函数重载，而C语言不可以

关键：C++和C语言关于名字修饰的规则----名字修饰规则

C语言名字修饰规则:编译器只是在函数名字前增加_

C++名字修饰规则：函数名字_参数类型1_参数类型2_.....

4、extern　“C" 的作用：将函数按照C语言的方式进行编译

四、引用

1、概念：引用是一个别名，与其引用的实体共用同一块内存空间，编译器不会给引用变量重新开辟一块新的内存空间

2、特性:引用在定义时必须初始化

一个实体可以有多个引用

一个引用一生只能与一个实体进行结合:引用一旦引用了一个实体，就不能再去引用其他实体

3、常引用-----const类型的引用

const int a=10;　int& ra=a;　//报错　const int& ra=a;

const int& b=10;

double d=12.34;　int&rd=d;　//报错　const int&rd =d;

临时变量：用户既不知道临时变量的名字，也不知道改块空间的地址，就无法修改该空间中的内容，认为临时变量具有常性---将其看成一个常量

4、作用---参数、返回值

C++中总共有三种传参方式：

a、传值

优点：对函数形参的改变不会影响外部实参

缺点：1、不能通过形参改变外部实参 2、浪费空间，减低代码的运行效率

b、传地址

优点：1、可以通过形参改变外部实参 2、传参的效率

缺点：1、函数的副作用会影响外部的实参 2、指针不安全 3、降低代码的可读性

c、传引用

优点：>>传参效率比较高(在传参期间就不会生成实参的一份拷贝)也比较节省空间，因为不会创建临时变量

>>可以通过修改形参来改变实参（形参就是实参的别名）

一般在C++中：传参是能用引用尽量用引用，如果不需要用形参改变实参，可以用const来进行修饰

引用作为函数返回值：注意----不能返回函数栈上的空间

5、传值、引用效率

传参效率：传引用比传值效率高

传引用和传指针------效果基本相同

6、引用和指针的区别（*****）

1、从概念：别名----与其引用的实体共用同一块内存空间

2、从底层实现：编译器在底层实际是将引用转换为指针类进行处理

因此引用在底层实际也有自己的内存空间

从底层实现上看：引用和指针是没有区别的

从应用或概念层面：

>>引用在定义期间必须初始化，指针没要求

>>引用在初始化时引用一个实体后，不能在引用其他实体

>>在sizeof中含义不同：引用结果为引用类型的大小，但指针始终是在地址空间所占字节个数(4个字节)

>>引用自加即引用的实体增加1，指针自加即指针向后偏移一个类型的大小

>>有多级指针，但无多级引用

>>访问实体方式不同，指针需要显示解引用，引用编译器自己处理

>>引用比指针使用起来相对更安全

>>没有NULL引用，但有NULL指针

五、内联函数和宏函数的区别（*****）

C语言中宏：

宏常量：

>>优点：能够提高代码的可读性，一改全改

>>缺陷：宏常量没有类型，在编译阶段就不会进行类型检测

C++中为了解决宏常量的缺陷：在C++中使用const修改某个实体，该实体不仅仅是一个常量，而且在编译阶段也可以进行替换

宏函数：在C语言中，为了提高代码的运行效率，会将一些比较短小的代码利用宏函数来替代

在预处理阶段会用宏体来替代宏函数

>>优点：可以提高代码的运行效率

>>缺陷：1、对于宏函数的参数没有加括号时，可能会造成错误

2、因为宏函数在预处理阶段已经被替换，因此不能进行调式

3、宏函数的参数没有类型,也不会进行参数类型检测，错误会延迟到运行时，增加错误的代价

4、副作用

5、引起代码膨胀

内联函数----解决宏函数缺陷

1、概念--在C++中用inline修饰的函数，编译器会将其作为内联函数进行处理

2、作用:编译器在编译阶段,会用内联函数替换函数的调用,少了函数调用的开销,来提到代码的运行效率

注意：inline是一个建议性的关键字，建议编译器将函数当成内联函数来处理，但实际不一定：函数中最好不要有循环、递归、函数不易太长

如果一个函数在类中进行定义，编译器也会将函数当成内联函数来处理

>>优点：

1、参数不需要导出加括号

2、内联函数是一个函数，会参与编译，在编译节点会进行参数类型检测

3、在debug版本下是可以进行调式

4、也不会有什么副作用

>>缺陷：可能会引起代码膨胀

在C++中，构造函数、析构函数、static修饰成员函数、虚函数都可作为内联函数

六、C++三大特性：封装、继承、多态

类和对象------封装特性

1、什么是面向对象，面向对象和面向过程的区别？*****

2、类的定义：

将类的声明和定义全部放在类中

将类的声明和定义分开，头文件中放声明，源文件中放定义--->如果成员函数在类外进行定义，必须在函数名字前增加类名::

3、访问权限：public、protect、private

4、什么是封装？C++中是如何进行封装？

封装概念：将一个事物的内部实现细节包装起来，只需要对外提供操作对象的接口

C++如何实现封装:通过类的方式对对象进行封装,通过访问权限控制那些接口可以提供给外部来进行使用

5、一个类就是一个作用域；

6、类的实例化：用类类型创建对象的过程

类：类是用来描述一个对象的，对象都有那些属性，对象都有那些方法

对象：用类创建出来的一个实体

7、如何求一个类的大小---类对象的模型

a、通过编译来猜测：对象 列出对象所对应类的所有成员 结论1：对象中既包含成员变量，也包含成员函数

b、解决1进行分析-----如果每个对象中都包含成员函数，比较浪费空间 结论2：将成员变量和成员函数分开存放，对象中只存放成员变量+指针(指向所有成员函数存放位置的起始位置)

c、对情况1和情况2进行验证：

最终结论：对象中只包含类中的非静态的成员变量

求类对象大小的方式：只需要将对象中成员变量加起来，注意内存对齐