1. 设计类 抽象类
   1. class 类名{
   2. public 公共权限
   3. 设置 成员属性
   4. 设置 成员函数
   5. }
   6. 使用类 创建对象 实例化对象
   7. 类名 对象名
   8. 通过对象 来设置属性 调用成员函数
   9. 类和对象 关系？？？
      1. 类是对对象的抽象
      2. 对象是对类的实例
2. 内联函数 解决宏缺陷问题
   1. 给编译器一个建议，加上关键字，编译器不一定按照内联处理
   2. 不加关键字，也许编译器还偷摸的给你加inline
   3. 成员函数 默认加上inline关键字
   4. 函数声明加了关键字，函数实现也要加inline关键字
   5. 如果函数实现时候，没有加inline关键字 ，那么这个函数依然不算内联函数
3. 函数默认参数
   1. 参数可以有默认值

|  |
| --- |
| void func( int a , int b = 10,int c = 1 )  {  cout << "a + b + c = " << a + b + c << endl;  } |

* 1. 如果有一个位置有了默认值，那么从这个位置开始，从左往右都必须有默认值
  2. 函数声明和实现 只能有一个有默认值

|  |
| --- |
| void myFunc(int a = 10, int b = 10);  void myFunc(int a , int b ){} |

1. 函数占位参数
   1. void func(int) 占位参数 调用时候必须要提供这个参数
   2. 占位参数也可以有默认值

|  |
| --- |
| void func2(int a , int = 1)  {  } |

* 1. c语言中没有默认参数 和占位参数

1. 函数重载的基本语法
   1. 函数名称相同 又在同一个作用域下
   2. 函数参数个数不同、类型不同、顺序不同都可以满足重载条件
   3. 函数的返回值可以作为函数重载条件吗？ 不可以
   4. 当函数重载碰到了函数默认参数 要注意避免二义性

|  |
| --- |
| //当函数重载 碰到了 默认参数时候，要注意避免二义性问题  void func2(int a,int b = 10)  {  }  void func2(int a)  {  }  void test02()  {  //func2(10);  } |

1. extern C浅析
   1. 解决了C++文件中调用C语言的代码
   2. ifdef \_\_cplusplus extern “C” {
   3. }

|  |
| --- |
| C语言文件：  #ifdef \_\_cplusplus //两个\_下划线  extern "C" {  #endif  #include <stdio.h>  void show();  #ifdef \_\_cplusplus //两个\_下划线  }  #endif |

1. C++语言的封装
   1. 将属性和行为作为一个整体，来表示生活中具体的事物
   2. 有访问权限
   3. class 和struct唯一区别 默认权限不同
      1. class默认是private
      2. struct 默认是public
   4. public 是类内类外都可以访问到
   5. protected 类内可以，类外不可以，(当前类的子类可以访问)
   6. private 类内可以，类外不可以，（子类不可以访问）
2. 建议将所有成员属性设置为私有
   1. 自己提供公共的对外接口来进行 set或者get方法访问