1. 函数调用流程
   1. 宏函数
      1. 在一定程度上会比普通函数效率高，普通函数会有入栈和出栈的时间开销
      2. 将比较频繁短小的函数 写为宏函数，直接跑源码
      3. 优点： 以空间换时间
   2. 调用惯例
      1. 主调函数和被调函数都必须有一致的约定，才可以正确的调用函数，这个约定我们称为调用惯例
      2. 调用惯例包含的内容： 出栈方、参数的传入顺序、函数名称的修饰
      3. c和c++下默认的调用惯例为 cdecl
2. 栈的生成方向以及内存存储方式
   1. 生长方向
      1. 栈底 --- 高地址
      2. 栈顶 --- 低地址
   2. 内存存储方式
      1. 高位字节数据 --- 高地址
      2. 低位字节数据 --- 低地址
      3. 小端对齐
3. 空指针和野指针
   1. 空指针 ---不允许向NULL和非法地址拷贝内存
   2. 野指针
      1. 未初始化指针
      2. malloc后也free了，但是指针没有置空
      3. 指针操作超越变量作用域
   3. 空指针可以释放 ，但是野指针不可以释放
4. 指针的步长
   1. 指针变量+1 之后 跳跃的字节数量
   2. 解引用的时候，取的字节数
   3. 对自定义数据类型进行练习
      1. 如果获取自定义数据类型中属性的偏移
      2. offsetof( 结构体 ， 属性 )
      3. 头文件 #include<stddef.h>
5. 指针的间接赋值
   1. 满足条件
      1. 一个普通变量和一个指针变量（或者一个实参一个形参）
      2. 建立关系
      3. 通过 \* 进行赋值
6. 指针做函数参数的输入输出特性
   1. 输入特性：
      1. 在主调函数中分配内存，被调函数使用
   2. 输出特性：
      1. 被调函数中分配内存，主调函数使用
7. 字符串强化训练
   1. 字符串是有结束标志 \0
   2. 利用三种方式对字符串进行拷贝
      1. 利用[]
      2. 利用指针
      3. 利用while (\*dest++ = \*source++){}
   3. 利用两种方式对字符串进行反转
      1. 利用[]
      2. 利用指针
8. 格式化字符串
   1. 可以利用sprintf对字符串进行格式化
   2. sprintf(目标字符串, "格式",占位参数…);