1. 数据类型
   1. 编译器指定出的数据类型，为了更好的管理内存
2. C语言标准
   1. ANSI 美国国家标准协会指定出的标准，在89年指定出第一套标准 C89标准
3. typedef使用
   1. 主要用途：给类型起别名
   2. 可以简化struct 关键字
   3. 可以区分数据类型
   4. 提高代码移植性
4. void的使用
   1. 无类型，不可以创建变量，无法分配内存
   2. 限定函数返回值
   3. 限定函数中的参数列表
   4. void \* 万能指针，可以不需要强制类型转换 给其他指针赋值
5. sizeof 的使用
   1. 本质： 不是函数，而是一个操作符
      1. 当统计类型占的内存空间时候，必须要加 小括号
      2. 当统计变量占内存空间时候，可以不加小括号
   2. 返回值类型是 无符号整型 unsigned int
   3. 可以统计数组的长度
      1. 数组名称如果在参数列表中，会退化为指针，指向数组的第一个元素
6. 变量的修改方式
   1. 直接修改
   2. 间接修改
      1. 通过指针对内存进行修改
      2. 对自定义数据类型进行了修改
7. 内存分区
   1. 运行前
      1. 代码区
         1. 共享的
         2. 只读的
      2. 数据区
         1. data 已初始化的全局变量、静态变量、常量
         2. bss 未初始化的全局变量、静态变量、常量
   2. 运行后
      1. 栈区
         1. 属于先进后出的数据结构
         2. 由编译器管理数据开辟和释放
         3. 变量的生命周期在该函数结束后自动释放掉
      2. 堆区
         1. 容量远远大于栈
         2. 没有先进后出这样的数据结构
         3. 由程序员管理开辟和管理释放
            1. malloc、free
         4. 记住手动开辟的要手动释放
8. 栈区
   1. 不要返回局部变量的地址，因为局部变量在函数执行之后就释放了，我们没有权限取操作释放后的内存
9. 堆区
   1. 在堆区开辟的数据，记得手动开辟，手动释放
   2. 注意事项
      1. 如果在主调函数中没有给指针分配内存，那么被调函数中需要利用高级指针给主调函数中指针分配内存
10. 数据区
    1. 放入是静态变量、全局变量、常量
    2. static 和 extern 区别
       1. static 静态变量：编译阶段分配内存，只能在当前文件内使用，只初始化一次
       2. extern 全局变量，C语言下默认的全局变量前都隐藏的加了该关键字
    3. const修饰的变量
       1. 全局变量
          1. 直接修改 失败
          2. 间接修改 失败 原因放在常量区，受到保护
       2. 局部变量
          1. 直接修改 失败
          2. 间接修改 成功 原因放在栈上
          3. 伪常量 不可以初始化数组
    4. 字符串常量
       1. 不同的编译器可能有不同的处理方式
       2. ANSI没有指定出标准
       3. 有些编译器可以修改字符串常量，有些不可以
       4. 有些编译器将相同的字符串常量看成同一个