1. 一维数组名称
   1. 本质并不是一个指针
   2. 有两种特殊情况
      1. 对数组名称进行sizeof
      2. 对数组名称 取地址 ，获取的指针步长是整个数组长度
   3. 除了两种特殊情况外，都是指向数组中首元素的地址 的指针
   4. 数组名--- 指针常量，指针的指向不可以修改
   5. 如果将数组名传入到函数参数中，为了提高可读性通常写为
      1. int arr[]
   6. 访问数组元素时候，下标可以为负数
2. 如何定义数组的指针
   1. 先定义出数组的类型，再通过类型创建数组指针
      1. typedef int(ARRAY\_TYPE)[5];
   2. 先定义数组指针的类型，再创建数组指针变量
      1. typedef int(\*ARRAY\_TYPE)[5];
   3. 直接创建数组指针变量
      1. int(\*pArr)[5] = &arr;
3. 二维数组名称
   1. 除了两种特殊情况外，都是指向第一个一维数组的指针
   2. 两种特殊情况
      1. sizeof 统计整个二维数组长度
      2. 对数组名称取地址 int(\*p2)[3][3] = &arr;
   3. 二维数组做函数参数传递方式
      1. void printArray( int p[][3] , int row, int col)
      2. void printArray(int p[3][3], int row, int col) 可读性高
      3. void printArray( int(\*p)[3] , int row ,int col)
   4. 数组指针 指针数组 区别
      1. 数组指针 是一个指向数组的指针 int (\*p)[10];
      2. 指针数组 是一个存放指针的数组 int \* p[10];
4. 指针数组的排序
   1. 选择排序
      1. 假设排序规则为从小到大
      2. 先认定一个最小值下标为i，通过j = i+1找的真实最小值下标
      3. 判断计算出的真实最小值下标 和开始认定的i是否相等，如果不相等，交换i和min下标的两个元素
   2. 对指针数据进行从大到小排序
5. 结构体基本使用
   1. 如果有typedef 定义结构体，那么后面跟着的单词是类型的别名
   2. 没有typedef，定义结构体，后面跟着的单词是一个结构体变量
   3. 结构体数组
      1. 在栈上开辟
      2. 在堆区开辟
6. 结构体赋值问题以及解决
   1. 系统提供的赋值操作是简单的值拷贝，逐字节拷贝---- 浅拷贝
   2. 如果属性中有指向堆区的内容，在释放期间会导致堆区重复释放，并且还有内存泄露
   3. 解决方案：利用深拷贝，手动赋值
7. 结构体嵌套一级指针
   1. 设计结构体 struct Person { char \* name,int age }
   2. 在堆区创建 结构体指针数组 malloc(sizeof(struct Person \*) \* 3);
   3. 给每个结构体也分配到堆区
   4. 给每个结构体的姓名分配到堆区
   5. 打印数组中所有人的信息
   6. 释放堆区数据