指针和数组

数组

- 概念：相同数据类型的变量集合
  - 数组分配在栈上面
  - 数组元素连续存储
    - 数组元素的地址 an = a0+ n*sizeof(type)
- 定义：数据类型 数组名[数组的大小]
  - 初始化： = {初始化列表}
    - 自动推导大小：忽略数组的大小，数组的实际大小根据初始化列表自动推断
- 调用数组： 数组名[数组下标]
  - 数组下标的范围：0 ~ 数组大小-1
  - []的原理：p[i] 等价于 *(p+i)
- 数组和指针的关系
  - 数组名的数值是？类型是？
    - 首个元素的地址，也是整个数组的地址
    - 类型是以元素类型为基类型的指针
      - 不能修改指向，可以修改指向位置的数值：const pointer 指针常量
- 二维数组
  - 定义：数据类型 数组名[行][列]
    - 可以按照一维数组的方式，进行初始化
    - 也可以在初始化列表里面嵌套列表，进行初始化
  - 储存方式：按行存储
    - 同一行元素连续存储
    - 行与行之间连续存储
    - 等价成为一个一维数组
  - 地址计算问题 int arr[M][N]
    - 数组名是一个指针，基类型是一个数组 int[N]
      - arr+1 地址值增加了 1*sizeof(int)*N
      - *arr+1 地址值增加 1*sizeof(int)
  - 调用：数组名[行][列]
  - 二维数组的本质
    - 数组的数组
  - 数组名的含义
    - int arr[M][N] arr的含义是一个指针，它的基类型是int[N]
- 字符数组
  - 字符串的特点：以'\0'结尾的字符数组
    - 不会检查数组的大小
  - 字符串的使用问题
    - char str[20] = "hello world"
    - 使用函数读取的时候，请务必使用memset将空间里面的所有位置都置为0
  - 一些函数
    - strlen strcpy strcat strcmp
  - 大小问题
    - 什么时候可以使用sizeof

指针

- 概念：存放了已分配空间的地址的一个变量
- 定义：基类型 * 指针名
- 指针的数据类型由其基类型决定
  - 解引用、间接访问的空间大小和解释方式
  - 指针偏移的脚步
- 指针的偏移（加上或者是减去一个整数）：根据基类型的大小，进行地址的增加或减少
  - 注意理解指针的偏移和数组之间的联系
- 数组指针
  - 二维数组可以看成元素为一维数组的数组
  - 一维数组的数组名在进行偏移或者是解引用的时候，看成是以元素类型为基类型的指针
- 动态数组
  - 分配在堆上
  - 申请：p = （基类型 *）malloc（数组大小*sizeof(基类型)）
    - 不要越界 0 ~ 数组大小-1
  - 释放：free(p)
    - 注意这个p的指向不能偏移
    - 注意将p=NULL 避免野指针
- 指针的传递
  - 如何在被调函数里面修改主调函数的变量
    - 主调函数将变量的地址传入被调函数
    - 被调函数使用解引用（间接访问）的方式修改指向变量的值
      - 间接访问 * [] ->
- 二级指针
  - 定义：基类型为指针类型的指针
  - 二级指针的传递
    - 改变一级指针的指向
  - 专题：如何排序二维数组
    - 数组指针
      - strcpy/memcpy
    - 二级指针
      - 交换指针
  - 心中要有一幅指针指向的内存图
- 函数指针
  - 概念比较复杂，但是使用比较简单