C语言 07指针

why

表示一些复杂的数据结构

快速的传递数据

使函数返回一个以上的值

能直接访问硬件

能够方便的处理字符串

是理解面向对象语言的引用的基础

——》指针是C语言的灵魂

定义

指针——》是地址，即从零开始的非负整数的内存单元的编号

指针就是地址，地址就是指针

指针变量

指针变量是存放指针，即存放地址的变量，与指针有本质的区别

int * p; // int * 是一个变量类型，叫指针变量，这个表示存放的是 int 类型变量的地址的变量；p 是变量的名字

int i = 3;

p = &i; // 由于 p 是指针变量，因此不能直接把 i（整数变量）赋给它，而是把 i 的地址（&i）赋给 p

*p == i; // *p 是以 p 的内容为地址的变量。在此例中，*p 就是 i；i 就是 *p

分类

1. 基本类型指针

   int * p;

   如何通过被调函数修改主调函数中普通变量的值

   i. 实参必须为该普通变量的地址

   ii. 形参必须为指针变量

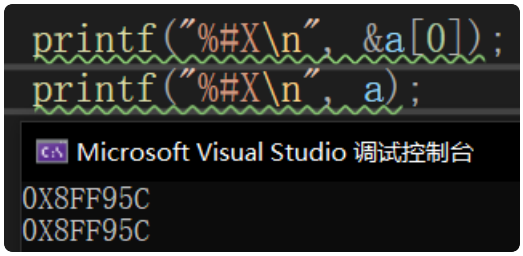
   iii. 在被调函数中通过

   *形参名 = 。。。。。，来修改普通变量的值

1. 指针和数组

   指针和一维数组

   数组名——》一维数组名是一个指针常量，存放着数组第一个元素的地址



   确定一个一维数组必须的参数——》首元素地址，数组长度

   下标和指针的关系

   a[i] 等价于 *(a + i)

   指针变量的运算

   不能相加，相除，相乘

   只能相减——》如果两个指针变量指向的是同一块连续空间中的不同存储单元

   一个指针变量占几个字节——》无论指向什么类型大小的变量，指针变量都为4个字节大小

专题：动态内存分配

传统数组的缺点（为什么需要动态分配内存）

a. 数组长度必须提前确定，且只能是常整数，不能是变量

b. 内存程序员无法手动释放传统数组的内存，除非程序结束

c. 数组的长度不能在运行过程中动态扩充或缩小

d. 一个函数内定义的数组，该函数运行期间，其他函数可以使用该数组

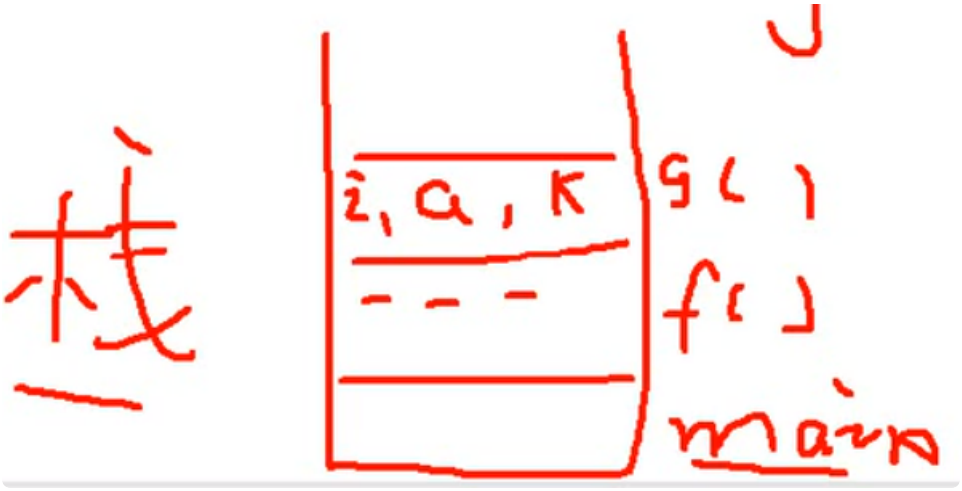
一旦该函数停止运行，该数组被释放，就不能在其他函数内使用

动态内存分配举例_动态数组的构造

```c
/*
    malloc 是 memory（内存） allocate（分配）的结合缩写


*/

#include <stdio.h>
#include <malloc.h>

int main(void)
{
    int i = 5;
    int* p = (int*)malloc(4);///12行
            /*
                1.要使用 malloc 函数，必须加入 malloc.h 库
                2.malloc【函数名】（所要分配的内存字节长度）
                3.malloc函数只有一个形参，且形参必须是整型
                4.malloc只返回首地址，类似于指针——》得指明数据类型，
                  才能获取完整的数据——》(int*)malloc(4)
                5.在12行，系统分配了8个字节的内存。4个静态分配给指针p
                    4个动态分配给p所指向的内容
                6. 只有malloc分配的内存是动态分配，变量类型+变量名的
                    是静态分配的内存——》指针p的内存是静态的，所指向的
                    内容的内存是动态的


            */

    * p = 5; // *p 就是被动态分配内存的int 变量，和 i 一样，只是分配方式不同
    free(p);/*  1. 静态内存只有在程序结束后被系统释放，动态内存可以在程序
                    运行中通过 free(被指指针)；随时释放
                2. 释放的是malloc分配的4个字节的动态内存，而不是指针p本身
                    静态分配的4个内存
            */
    return 0;
}
```

静态内存和动态内存的比较

静态内存——》栈 结构



动态内存——》堆 结构

跨函数使用内存的问题

跨函数使用静态内存（错误

```c
#include <stdio.h>



//通过函数改变 指针p的值
void f(int** q)
{
    int i = 5;
    //把 i 的值赋给p所指
    // *p = i;
    // 把 *p 用 q 表示
    // p == *q, p == &i;
    *q = &i;

}



int main(void)
{
    int* p;
    f(&p);

    printf("%d",*p);//该句语法没问题，但逻辑有问题
                    // 本句访问了不属于main 函数的内存，int i是 f 函数的
                    // p 可以保存静态变量 i 的地址，但不能访问

    return 0;
}
```

跨函数使用动态内存（正确

```c
#include <stdio.h>
#include <malloc.h>

void f(int** q)
{

    *q = (int*)malloc(sizeof(int)); //size（数据类型）表示返回值是该数据类型所占的字节
                                    /*
                                    * 由于不同的机器，int所占字节不同。因此sizeof（数据类型）
                                    * 比 4 更灵活。告诉电脑“占你这里int类型的大小”
                                    *
                                    */
    **q = 5;

}


int main(void)
{
    int* p;
    f(&p);
    printf("%d\n", *p);
    free(p);
    printf("%d\n", *p);

    return 0;
}
```