关键字

static 静态的

延长生命周期

修饰全局变量:只能在自己所在源文件内部使.全局变量外部连接属性,static修饰变成内部属性

修饰函数,实得函数只能在自己源文件内部使用(同上)

#define定义常亮和宏

1.define预处理命令

定义宏

#define ADD(x,y) x+y                 替换

#define ADD(x,y) ((x)+(y))

指针\*

[www.cplusplus.com](http://www.cplusplus.com/" \t "https://mail.qq.com/cgi-bin/_blank)

函数，自定义函数：类型＋函数名＋（传参）默认int 型

函数数组传参传的是首元素的地址（需要个数函数外求好传入）－－－－－－二分查找

函数的链式访问，把一个函数的返回值作为另一个函数的参数

函数递归，函数直接或间接的调用自身

存在限制条件，并且不断接近

有跳出条件，递归层数不能太深（栈溢出）                                                                                  stack overflow.com

for (row=0; row<100; row++) {

// 低效率：长循环在最外层

for ( col=0; col<5; col++ ) {

sum = sum + a[row][col];

}

}

for (col=0; col<5; col++ ) {

// 高效率：长循环在最内层

for (row=0; row<100; row++) {

sum = sum + a[row][col];

}

}

折半查找

汉诺塔

青蛙跳台阶

求一到一百带九的数字个数

1/1-1/2+1/3.............

求个整数的最大值

99乘法表

函数

返回值类型 函数名(参数类型 形式参数1，参数类型 形式参数2，…) {

函数体;

返回值;

}

函数名称不能相同

形参实参类型不一致, 会自动转换为形参类型

函数返回值的类型和return实际返回的值类型应保持一致。如果两者不一致,则以返回值类型为准,自动进行类型转换

负数的原码、反码和补码

二进制的最高位我们称之为符号位, 最高位是0代表是一个正数, 最高位是1代表是一个负数

一个负数的原码, 是将该负数的二进制最高位变为1

一个负数的反码, 是将该数的原码除了符号位以外的其它位取反

一个负数的补码, 就是它的反码 + 1

0000 0000 0000 0000 0000 0000 0000 1100 // 12二进制

1000 0000 0000 0000 0000 0000 0000 1100 // -12原码

1111 1111 1111 1111 1111 1111 1111 0011 // -12反码

1111 1111 1111 1111 1111 1111 1111 0100 // -12补码

判断闰年

交换两个整数的内容

九九乘法表

递归

递归函数构成条件

自己搞自己

存在一个条件能够让递归结束

问题的规模能够缩小

打印数的每一位

字符串逆序abcdef （循环+递归）reverse\_string

计算 一个数各个位的和

数组

元素类型 数组名[元素个数];

// int 元素类型

// ages 数组名称

// [10] 元素个数

int ages[10];

定义初始化可以同时也可以分开，只能在定义的同时初始化多个值, 不能先定义再初始化多个值

指定元素个数,部分初始化

int nums[5] = {[4] = 3,[1] = 2};

二维数组

初始化，行可以省略列不可以

内存中也是连续存储的

数组作为函数参数

冒泡

数组传参传的是首元素的地址

//数组名是什么?

//数组名是数组首元素的地址//但是有2个例外

l/1. sizeof(数组名)–数组名表示整个数组–计算的是整个数组的大小单位是字节

//2．&数组名–数组名表示整个数组–取出的是整个数组的地址

三子棋

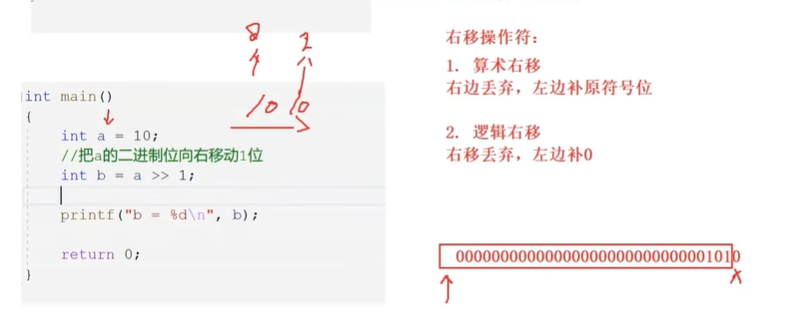
扫雷

操作符

算数操作符 +，-\*，/，%

移位操作符

<<, >>二进制左移/右移



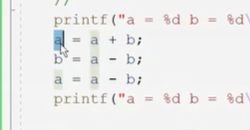
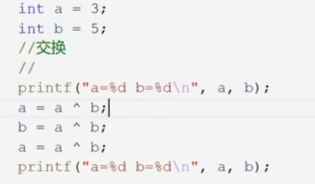
负数移动的是补码 ，内存中存的是补码

&按位与

|按位或

^按位异或（二进制位相同为0不同为1）

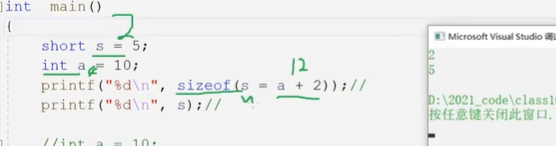
不使用的三个变量交换两个数的值

1. 
2. 

？？？求一个整数内存中存储二进制数1的个数

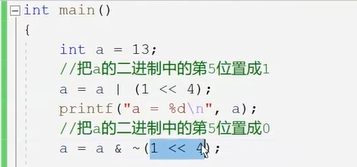
单目

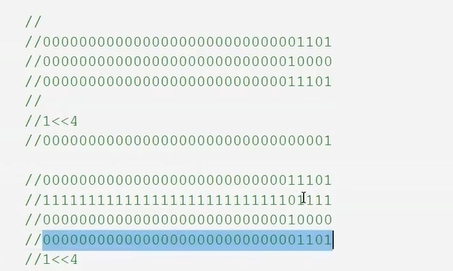
sizeof

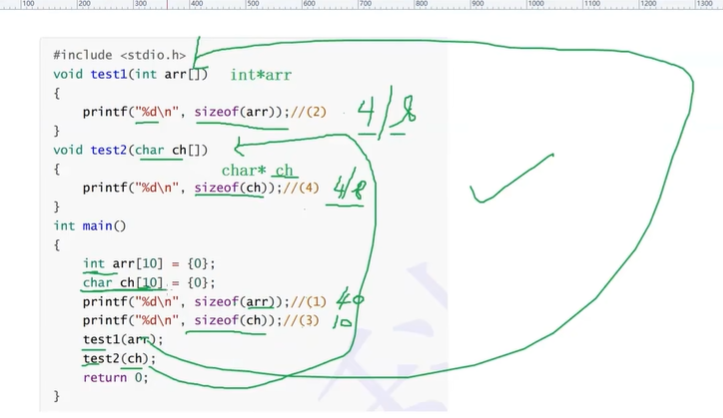


sizeof括号中的表达式不参与运算

~ 按位取反



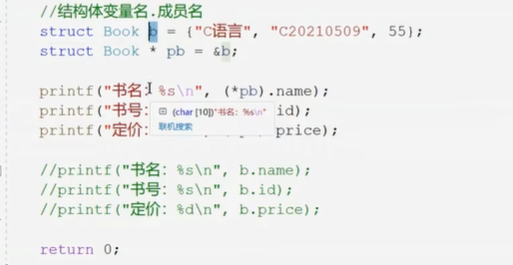


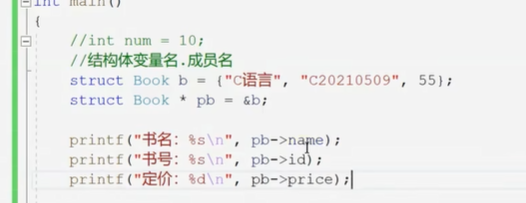


逻辑与，逻辑非，控制求职顺序

三木操作符（判断）

逗号表达式，依次计算取最后一个的结果。

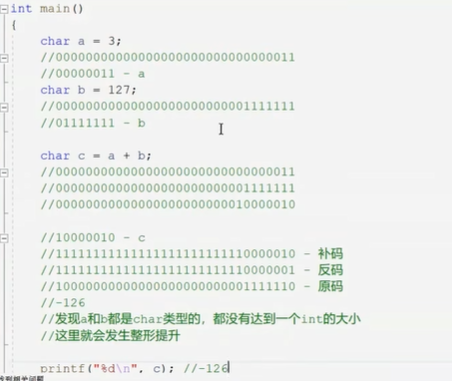




整形提升

自身类型达不到整形大小

整形提升看原符号位



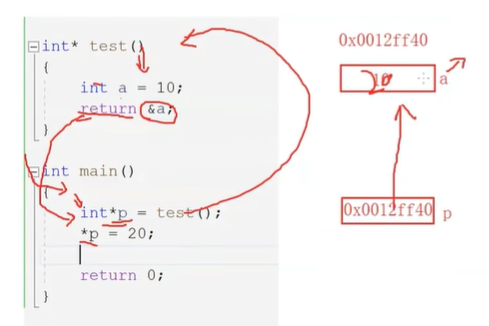
指针-编号-地址

指针类型决定，指针解引用的权限

指针类型决定了指针走一步走多远

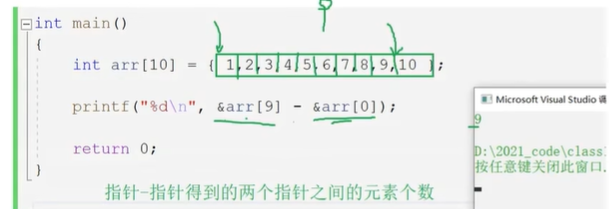
野指针

未初始化，指针越界

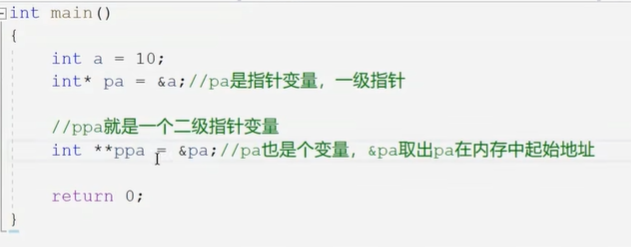
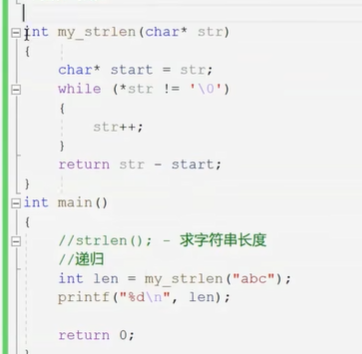


函数结束a的地址还给操作系统，此时\*p为野指针.

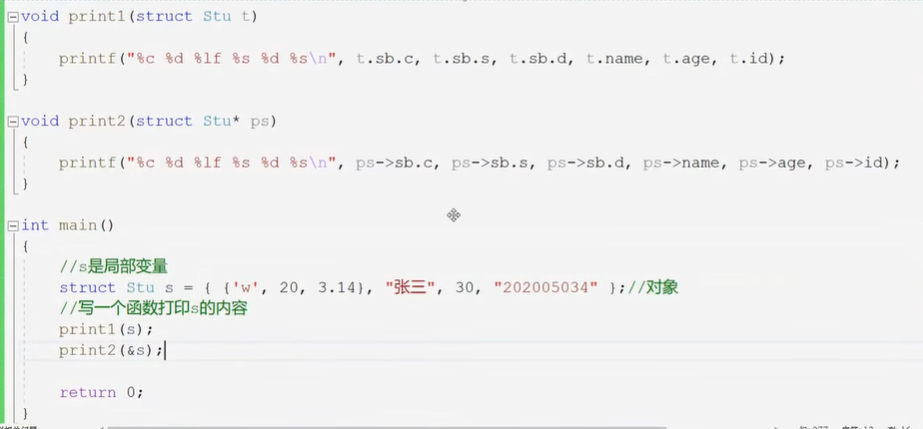
避免野指针：初始化（指针空指针），防止越界，释放空间，使用之前检查有效性（if(p !=NULL))



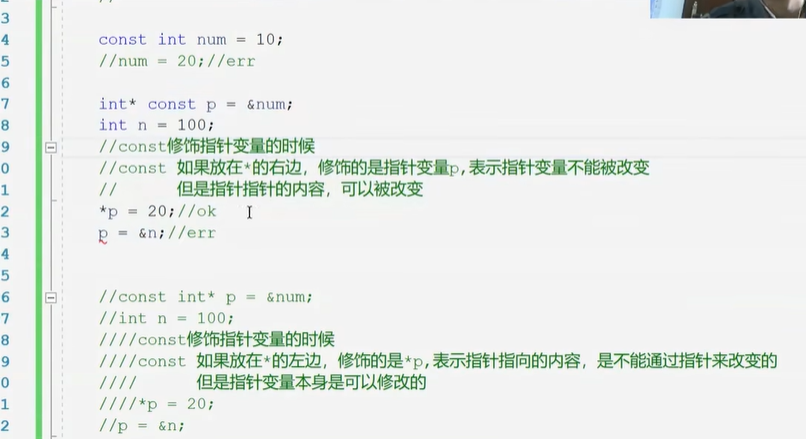
指针相减的前提，两个指针指向同一块空间



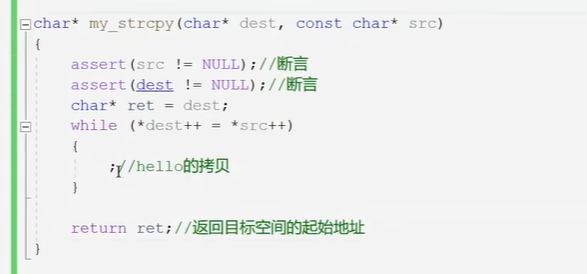
结构体



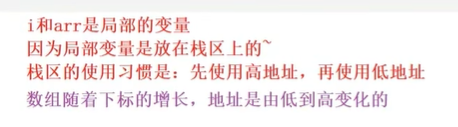
const修饰指针



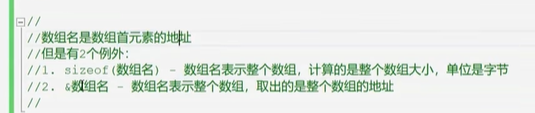
模拟strcpy函数

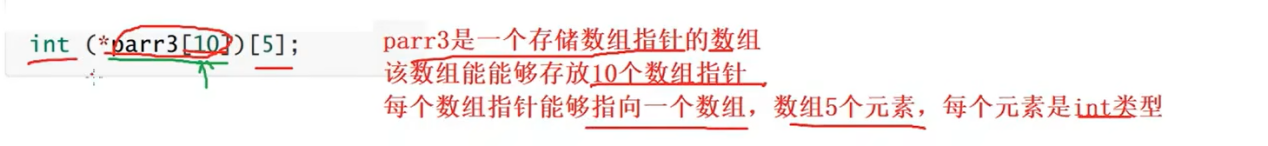


模拟strlen



指针进阶





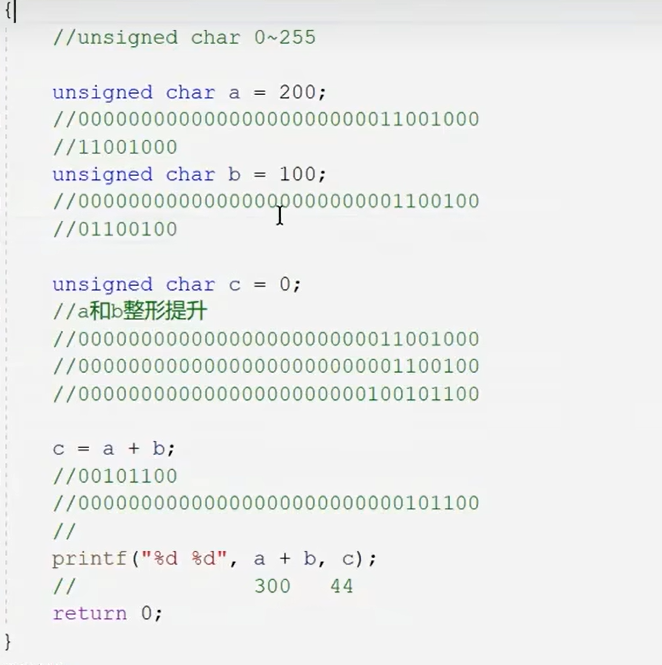
求s=a+aa+aaa

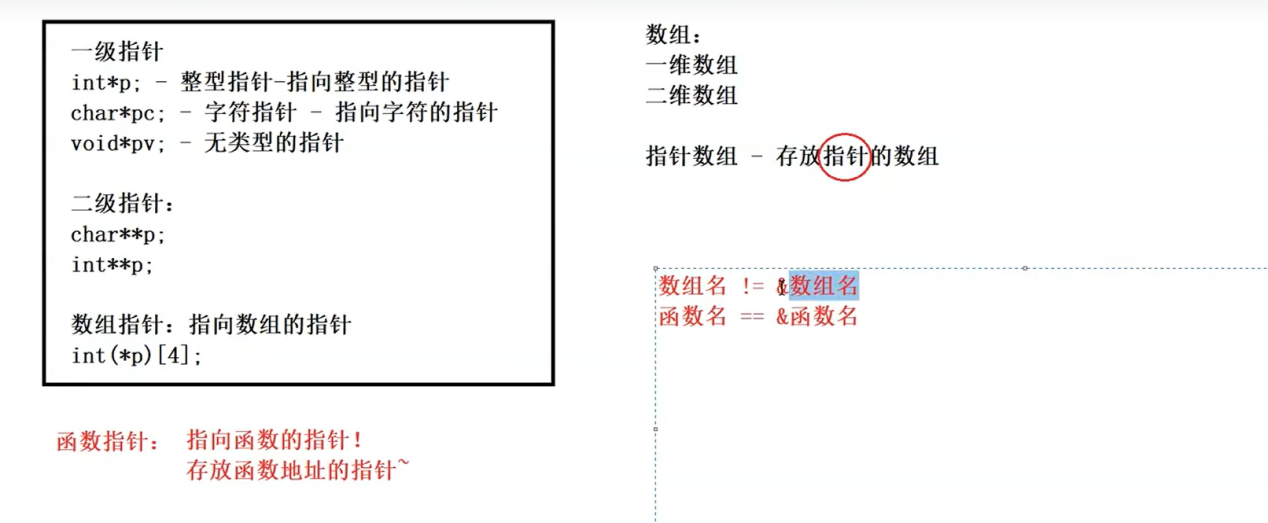
使用指针打印数组内容

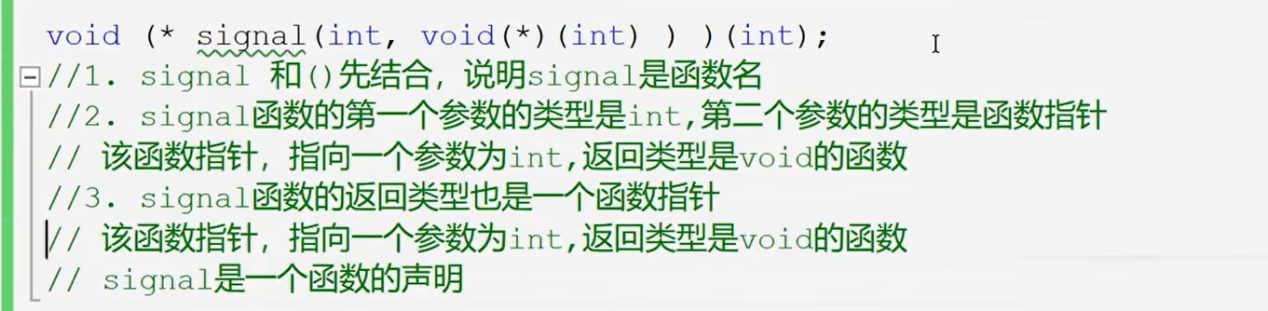
打印水仙花数1~100000

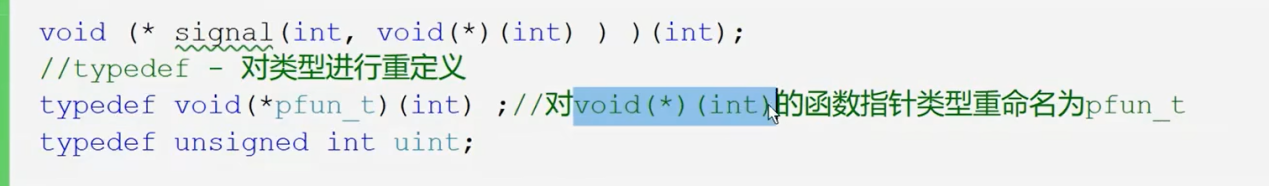
计算位数

每一位的n次方

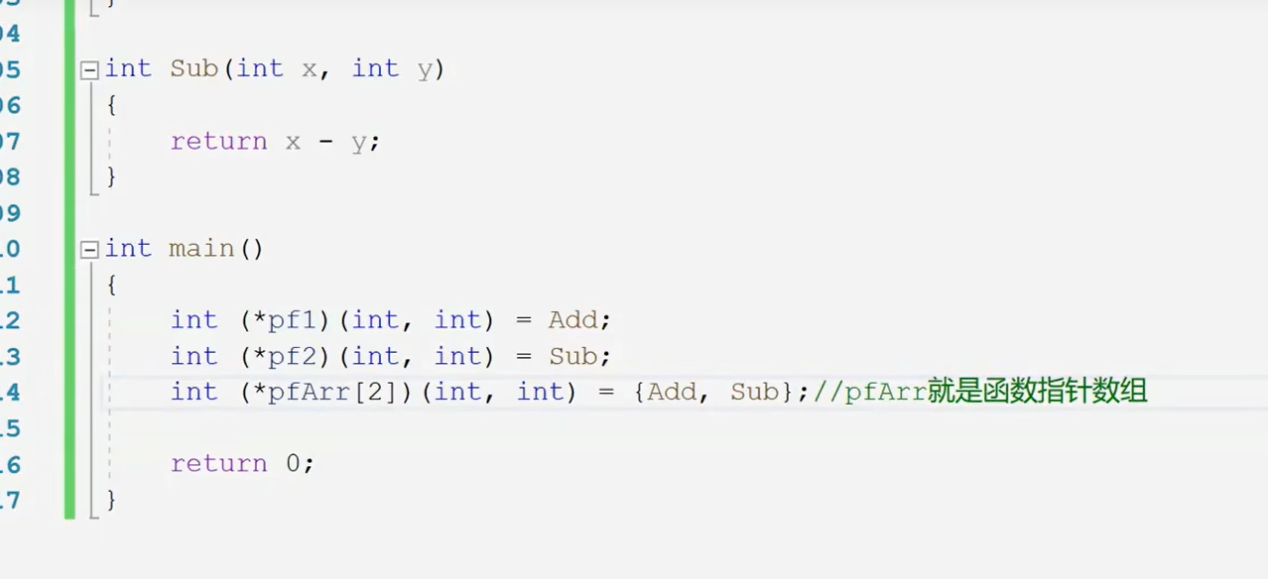






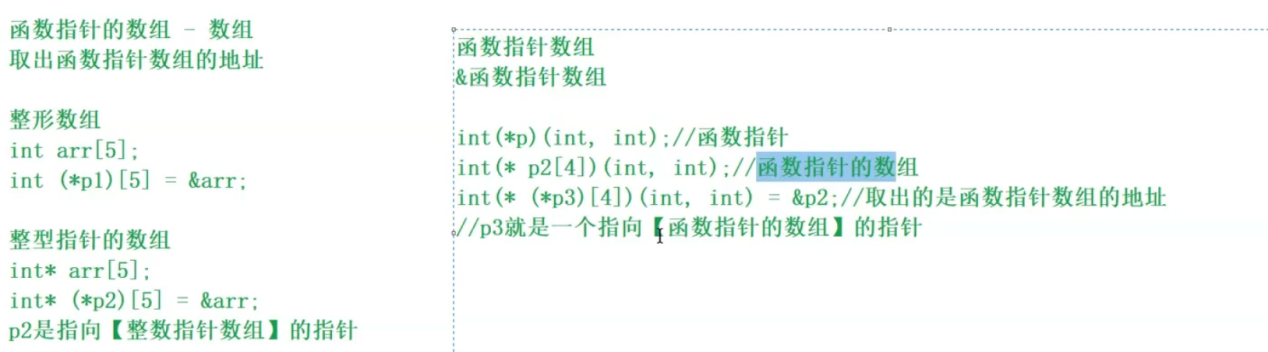


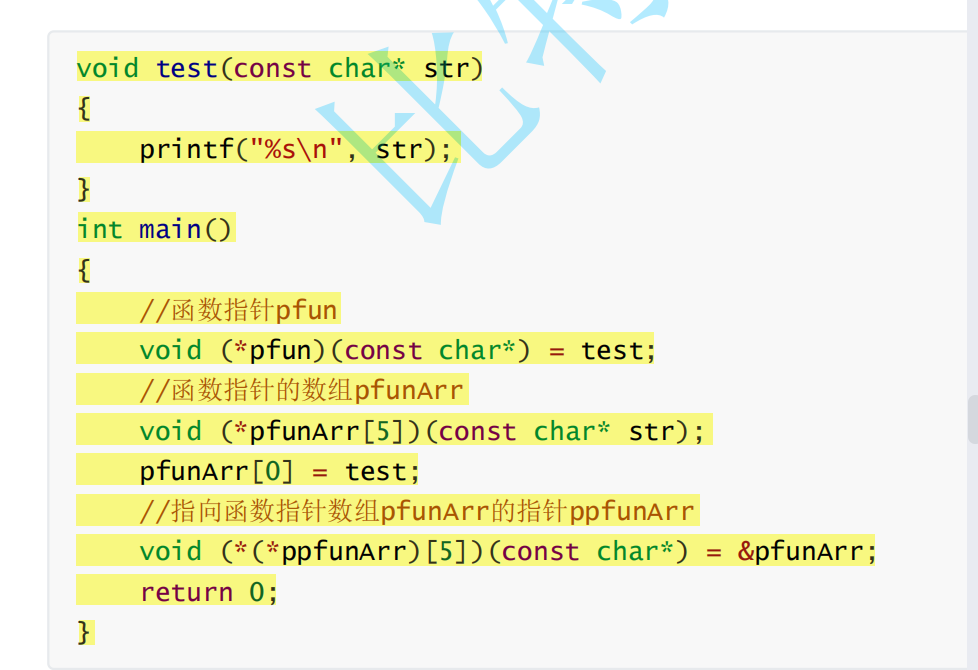
函数指针数组



函数指针数组使用

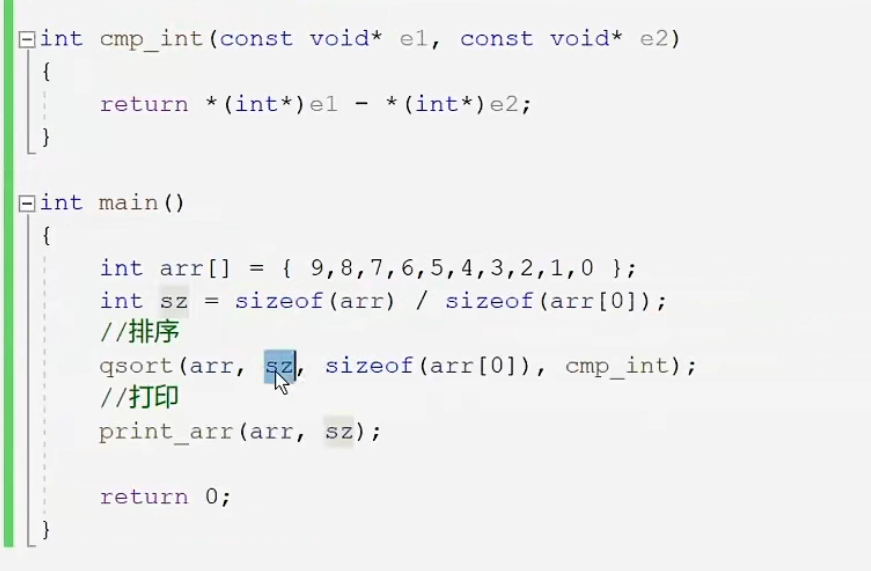


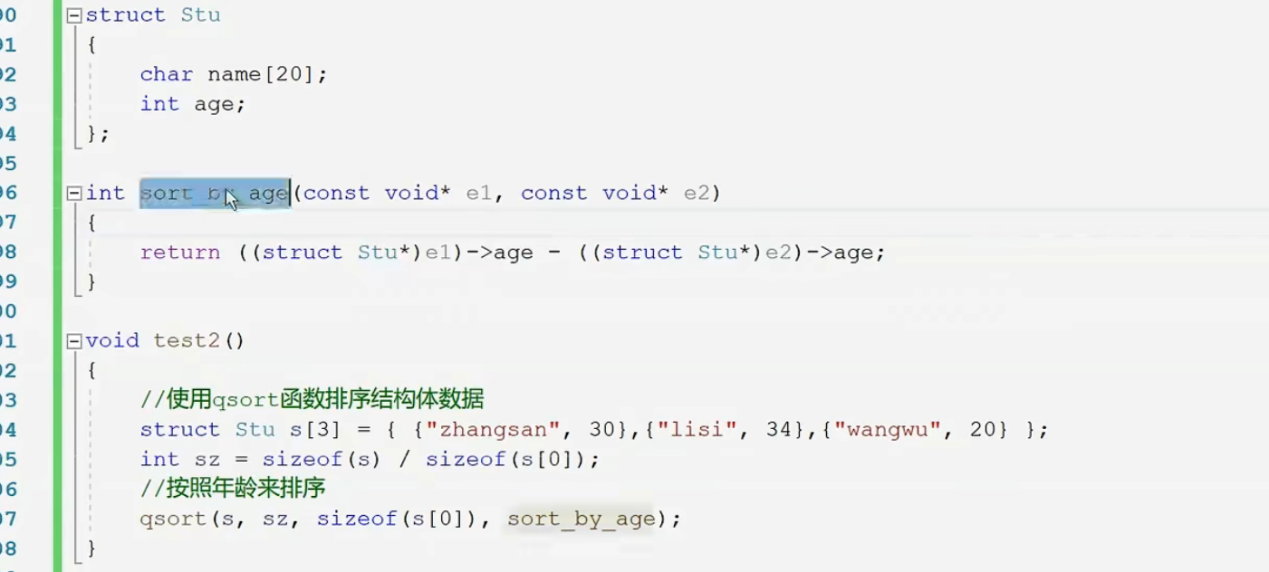


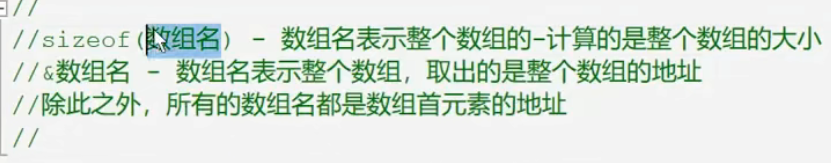


回调函数

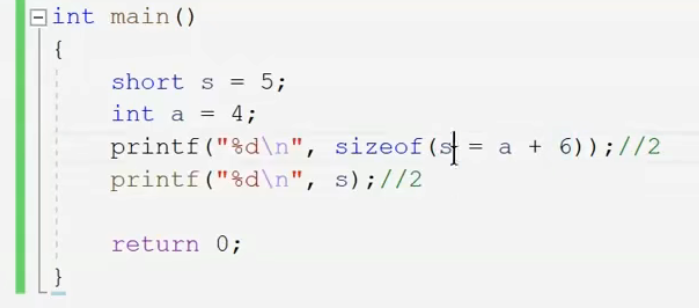




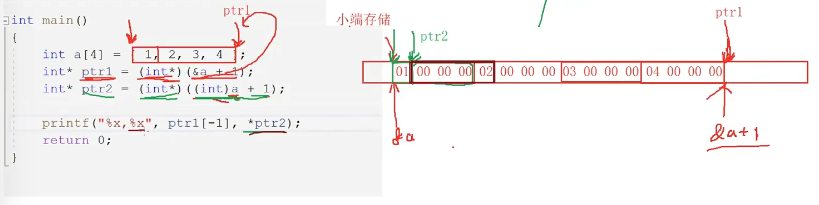


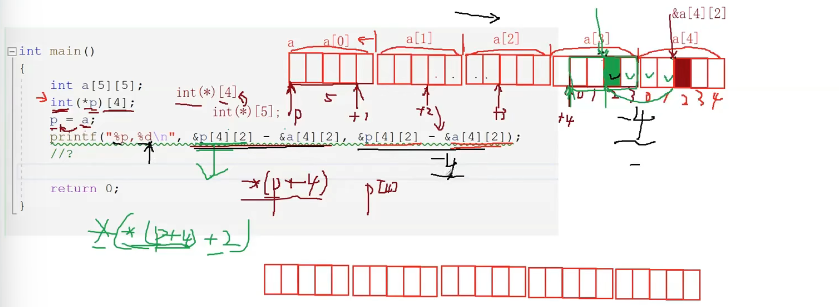


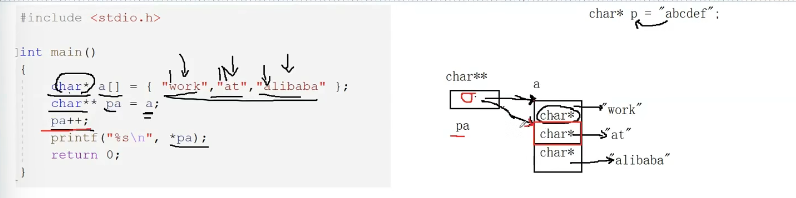




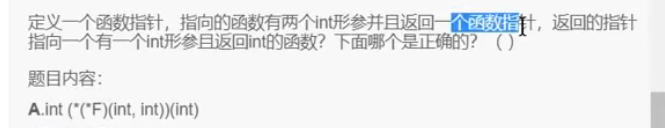
Sizeof内部表达式不参与运算

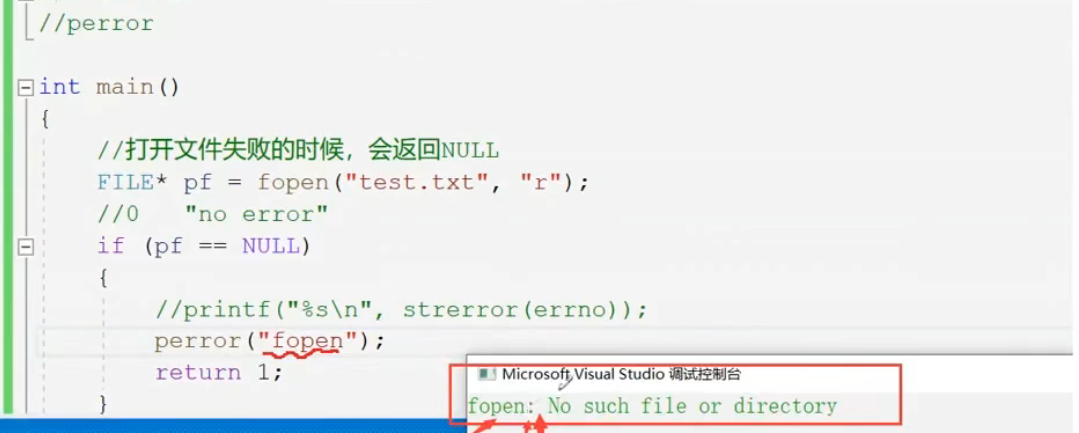






at





动态内存分配



文件操作

#include <stdio.h>

int main ()

{

FILE \* pFile;

//打开文件

pFile = fopen ("myfile.txt","w");

//文件操作

if (pFile!=NULL)

{

fputs ("fopen example",pFile);

//关闭文件

fclose (pFile);

}

return 0;

}



**文件的顺序读写**



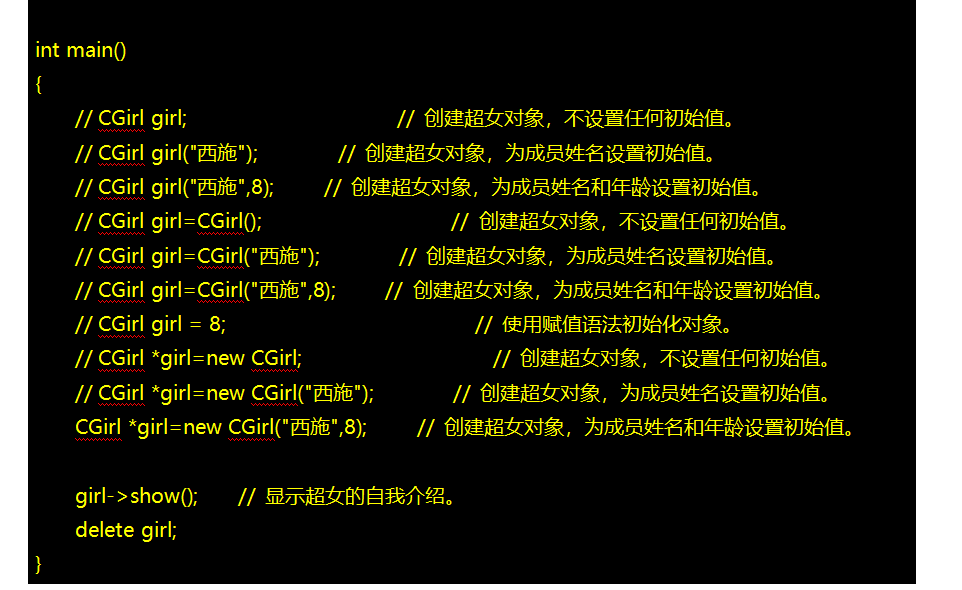
fgets读

Fputs写入



C++

类



CGirl g1(8); // 常规的写法。

CGirl g1 = CGirl(8); // 显式转换。

CGirl g1 = 8; // 隐式转换。

CGirl g1; // 创建对象。

g1 = 8; // 隐式转换，用CGirl(8)创建临时对象，再赋值给g。

将构造函数用作自动类型转换函数似乎是一项不错的特性，但有时候会导致意外的类型转换。explicit关键字用于关闭这种自动特性，但仍允许显式转换。

explicit CGirl(int bh);

CGirl g=8; // 错误。

CGirl g=CGirl(8); // 显式转换，可以。

CGirl g=(CGirl)8; // 显式转换，可以。

