# Inline

可以解决一些频繁调用的函数大量消耗栈空间（栈内存）的问题。特别的引入了**inline**修饰符，表示为[内联函数](https://so.csdn.net/so/search?q=%E5%86%85%E8%81%94%E5%87%BD%E6%95%B0&spm=1001.2101.3001.7020)。**inline**只适合函数体内代码简单的函数使用。

最后能否真正内联，看编译器的意思，它如果认为函数不复杂，能在调用点展开，就会真正内联。

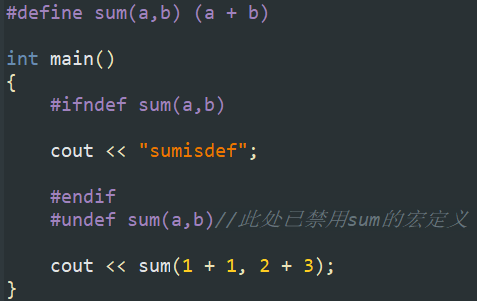
# pragma

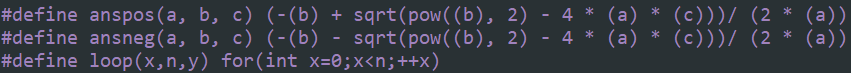
为了避免同一个文件被include多次

#pragma once

#ifndef #define #endif

# 宏定义





# 静态存储方式

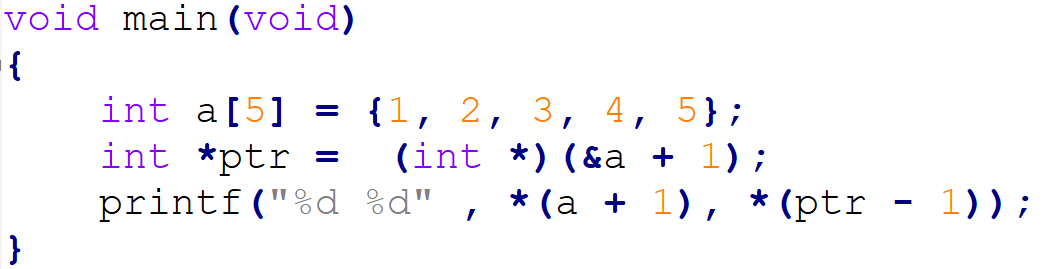
该存储方式通常是在变量定义时就分定存储单元并一直保持不变， 直至整个程序结束。**全局变量，静态变量**等就属于此类存储方式。

# 数组指针，指针数组，常量指针，指针常量

const int\* a int \* const a

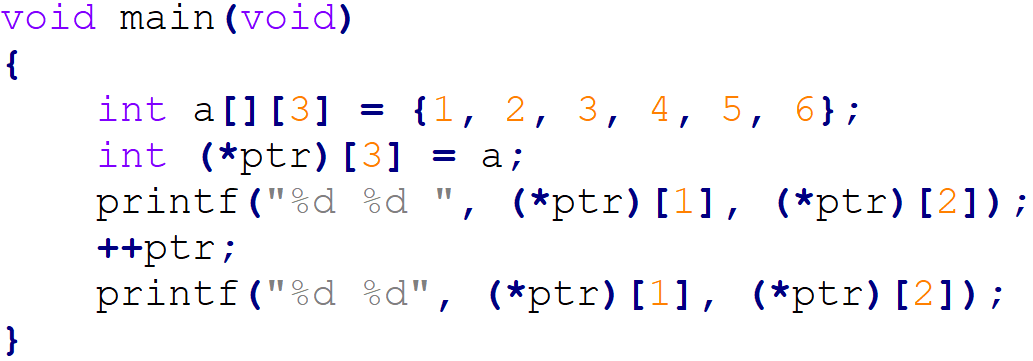
int (\*p)[n] int \*p[n] int \*p() int (\*p)()

# 考查指针

**这段程序的输出是:2 5**

&a取址后以a的整个数组为步进，a以a[0]也就是int为步进

# 考查指针数组

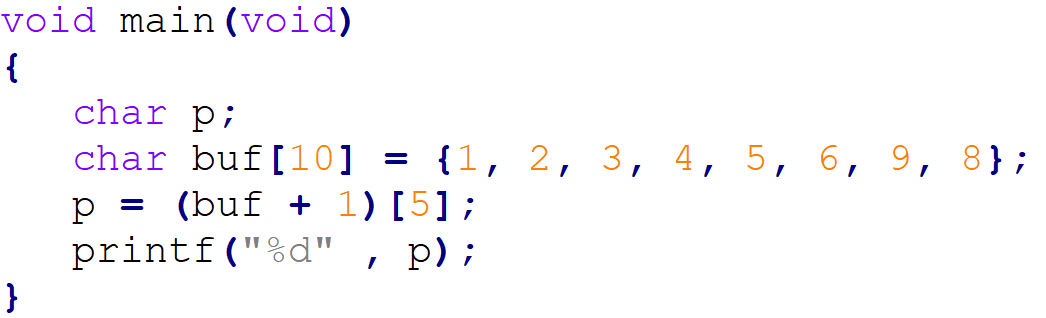


**这段程序的输出是: 2 3 5 6**

指针数组ptr为123，ptr+1为456，ptr+2为非法的不确定

ptr是一个数组的指针，该数组有3个int成员

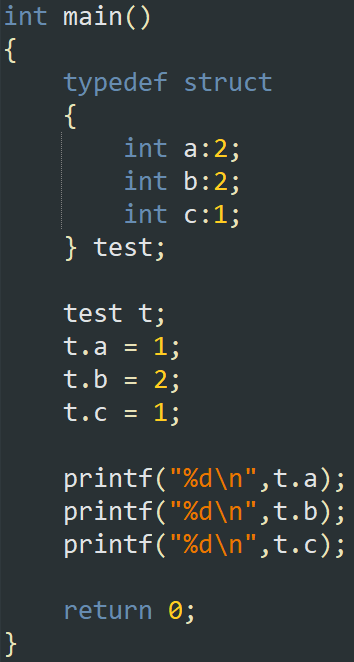
# 数组与指针的区别与联系



# Free之后记得置NULL

别看 free 和delete 的名字恶狠狠的（尤其是delete），它们只是把指针所指的内存给释放掉，但并没有把指针本身干掉。

# 原码反码补码

1，-2，-1

1000 0000 是-128 100 0000是-64

3因为是有符号的且占两位，就是11，那就是-1

2因为是有符号的且占两位，就是10，那就是-2，同1000 0000 -128

1因为是有符号的且占两位，就是01，那就是1

1因为是有符号的且占一位，就是1，那就是-1，同1000 0000

# memset、memcpy、strcpy

memset用来对一段内存空间全部设置为某个字符，一般用在对定义的字符串进行初始化为''或''；例:chara[100];memset(a,'',sizeof(a));

memcpy用来做内存拷贝，你可以拿它拷贝任何数据类型的对象，可以指定拷贝的数据长度；例：char a[100],b[50];memcpy(b,a,sizeof(b));注意如用sizeof(a)，会造成b的内存地址溢出。

strcpy就只能拷贝字符串了，它遇到'\0'就结束拷贝；例：chara[100],b[50];strcpy(a,b);如用strcpy(b,a)，要注意a中的字符串长度（第一个'\0'之前）是否超过50位，如超过，则会造成b的内存地址溢出。

# 判断bit有几位为1

