【要点】:

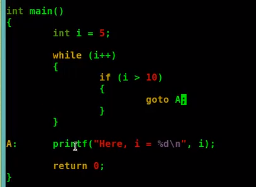
1、二进制的n个1 = 十进制的2n-1

2、++i：先加在用值，i++：先用值再加

如：i = 1; j = ++i🡺j为2，i为2;

i = 1; j = i++🡺j为1，i为2;

3、goto语句：（避免使用）



4、这个可以：char a[] = {1,2,3,4,5}; 会自动定义为5个字节的数组

C99特性：①部分赋值：char a[5] = {[0] = 1, [3] = 7}；未赋值自动的为0

②定义数组a[n]，n一定是确定的数，不可是变量。、

一维数组名表示首地址。

int a[3][4] = {{1}, {5}, {9}}；是将每行第一个元素赋为1、5、9（即第一列）

int a[][4] = {1,2,3,4,5,6,7,8}；（列必须定义）

5、int a = 1; printf(“%d”, a)，scanf(“%d”,&a) （a为普通变量）

char str[] = “abcdefg”; printf(“%s”, str); （打印字符串，给地址）

【二维数组】：array[4][5]

二维数组名array代表首行一维数组的首地址。array+1第二行的首地址

\*(array+1)+3（+优先级>\*）表示：第二行第四个元素 == array[1][3]

【指针】：

1、char \*p = &a; （\*p叫做指针变量，p叫做指针/地址，此处a为一个普通变量）

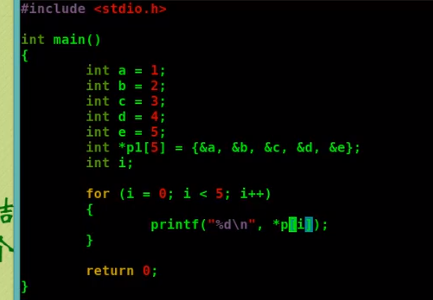
等价于：char \*p; p = &a;

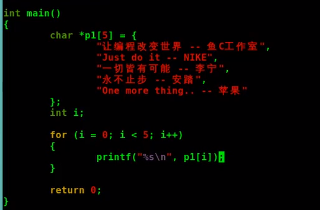
2、\*p与p[]，区别：p是一个可以改变的地址，如\*(p++)；而p为数组确定的地址，不可进行++运算，因为++运算的结果是赋给自身。

3、（[]运算符的优先级高于\*，先与谁结合本质就是谁）

指针数组：char \*p[2] = {&a, &b}; \*p[0]为一个char型指针变量，初始化给地址，看1点，

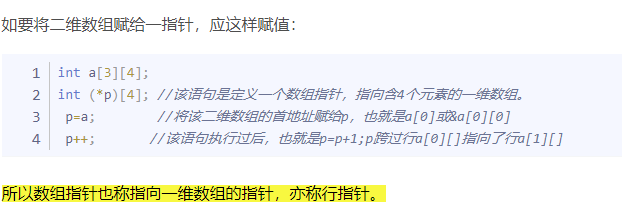
单独p无任何意义

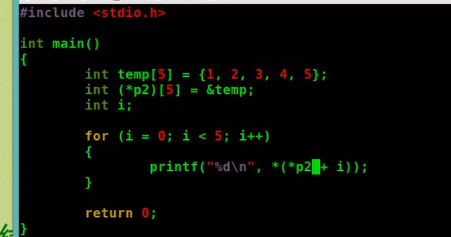




（打印字符串，给地址）

数组指针：char (\*p)[2]; 单独的p表示地址



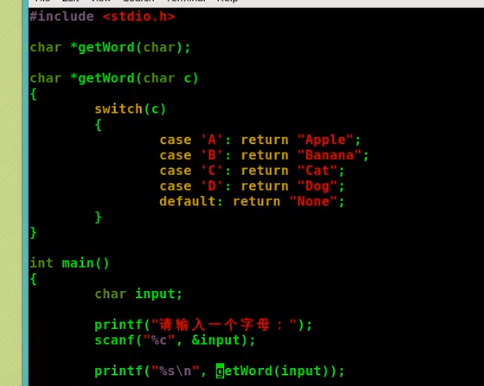


4、void \*p; 指向任何类型的指针

5、指针函数：int \*p() （()的优先级高于\*）

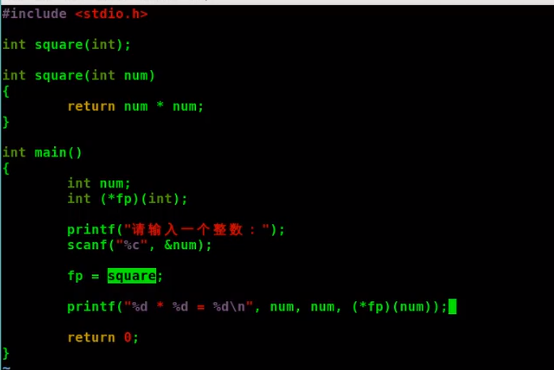
{ }

（不可以返回局部地址，直接使用字符串是可以的，有特殊区域存储字符串）

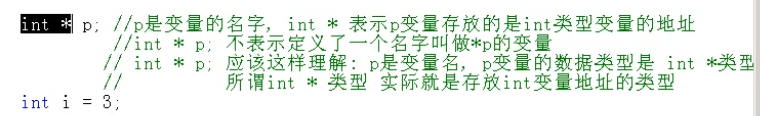


函数指针：int (\*p)()（函数名是函数入口地址）

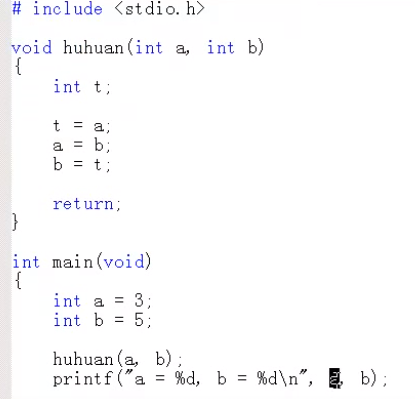
{ }

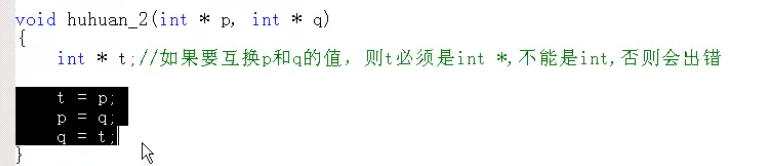


//郝斌C指针：

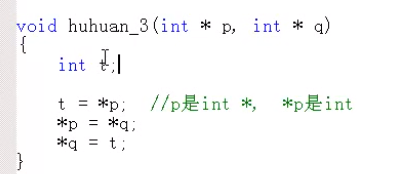


变量名仍然叫p，\*只是标注；p是int \*型，\*p是 int型，传递时，是将值传递给p而不是\*p，int \*p只是标注这个p变量只能存储地址。

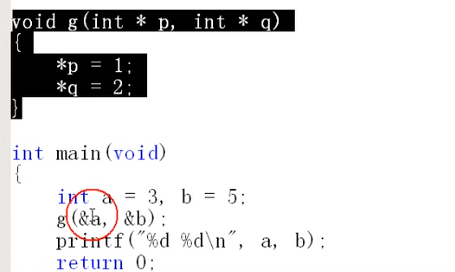


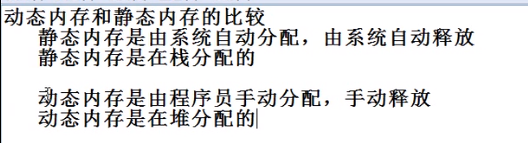
并未交换a，b，形参a，b不是主函数a，b,形参在传入时再分配空间,函数结束时就释放了

也不行。只是交换了pq，并不改变ab（a为\*p，b为\*q），ab的地址在定义时就确定了



可。\*p=a,\*q=b,交换\*p和\*q就交换了a,b





【共用体union】：（所有成员共用一个地址，同一进程只能使用一个成员，多次使用会进行覆盖）



【枚举类型enum】：（是整型常量，默认从0开始，依次+1）

enum Color

{

a=10，

b，

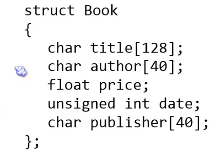
c，

d

};（若首元素赋值，后面依次+1；若中间元素赋值，后面依次+1，前面从0开始+1）

【结构体struct】：

1、（先声明再定义）



struct Book a； a.price（定义及使用）

struct Book \*a； a-> price

2、（声明同时定义，只需在结构体后面加上变量名）



3、（typedef）

typedef struct

{

int a；

char b；

float c；

}smart;

smart sb;（直接用smart定义结构体变量）

【define和typedef】：

typedef unsigned char u8； （重定义a为b，目的使用b）

#define pi 3.14 （将pi定义为3.14，目的使用3.14）