**程设作业第6章**

**附加1.**

#include <stdio.h>

#include <stdlib.h>

#define M 3

#define N 4

typedef int (\*P2D)[M];

void main\_v2() {

int i, j, a[M\*N], (\*p)[N];

for(i = 0; i < M \* N; i++) a[i] = i + 1;

p = (P2D)a;

for(i = 0; i < M; i++) {

for(j = 0; j < N; j++) {

printf("%4d", p[i][j]);

}

printf("\n");

}

}

void f() {

int i, a[N], \*p;

for(i = 0; i < N; i++) {

a[i] = i + 1;

}

p = a;

for(i = 0; i < N; i++) {

printf("%4d", p[i]);

}

}

void main() {

int i, j, a[M\*N], \*p[M];

for(i = 0; i < M \* N; i++) a[i] = i + 1;

for(i = 0; i < M ; i++) {

p[i] = &a[N \* i];

}

for(i = 0; i < M; i++) {

for(j = 0; j < N; j++) {

printf("%4d", p[i][j]);

}

printf("\n");

}

main\_v2();

}

**附加2：**

**数组指针（指向指针的数组） 又称行指针**

定义为  int (\*p)[2]   从形式来看 是一个指针变量，指向包含了两个元素的数组，每次指针加一时，都要横跨整个步长。似乎数组指针是C语言里专门用来指向二维数组的，它占有内存中一个指针的存储空间。

如要将二维数组赋给一指针，应这样赋值：

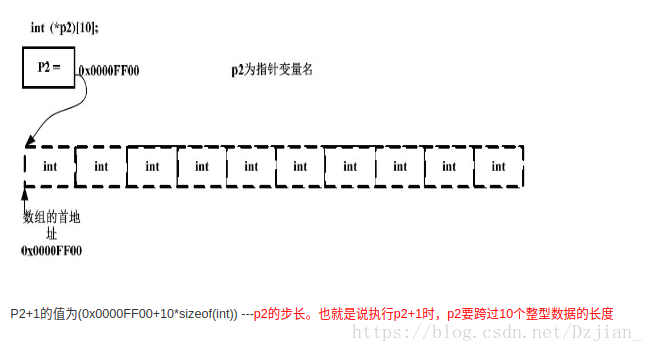
int a[3][4];

int (\*p)[4]; //该语句是定义一个数组指针，指向包含4个元素的一维数组。

 p=a;        //将该二维数组的首地址赋给p，也就是a[0]或&a[0][0]

 p++;       //该语句执行过后，也就是p=p+1;p跨过行a[0][]指向了行a[1][]

所以数组指针也称指向一维数组的指针，亦称行指针。



**指针数组**

顾名思义，这是一个存放指针类型的数组。如：int \*p[n] (二级指针 int \*\*p 本质上就是一个指针数组)，它有n个指针类型的数组元素。这里执行p+1是错误的，这样赋值也是错误的：p=a；因为p是个不可知的表示，只存在p[0]、p[1]、p[2]...p[n-1],而且它们分别是指针变量可以用来存放变量地址。但可以这样 \*p=a; 这里\*p表示指针数组第一个元素的值，a的首地址的值。

如要将二维数组赋给一指针数组:

int \*p[3];

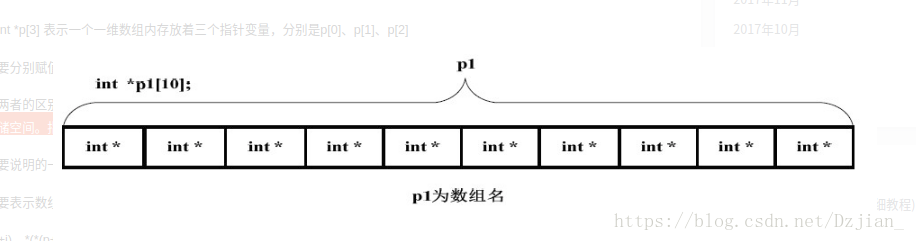
int a[3][4];

for(i=0;i<3;i++)

p[i]=a[i];

这里int \*p[3] 表示一个一维数组内存放着三个指针变量，分别是p[0]、p[1]、p[2]

所以要分别赋值。



---------------------

作者：Dzjian\_

来源：CSDN

原文：https://blog.csdn.net/Dzjian\_/article/details/79826236

版权声明：本文为博主原创文章，转载请附上博文链接！

**附加3：**

概念：创建变量（不包括函数指针）：

1. 创造一个变量
   * 1. 基本格式，\*\*\* a\*\*\* === Type a; (其中\*\*\* ()\*\*\* 表示为Type, 不仅限于基本类型，例int(\*a)[5] 中 Type 表示 int(\*)[5])
     2. 引申，\*\*\* (\*a)\*\*\* === Type (\*a);
        1. a类型推导，Type (\*a) -> (\*a)为Type类型 -> a的Type类型的指针
2. 创造一组变量
   * 1. \*\*\* a[N]\*\*\* === Type a[N]; 创建N个变量，变量类型为 Type, 但是a实际上是首元素的地址。
        1. Type a[N], 首先创建N个变量，其次将a[N]视为整体A, 即Type A, 数组元素类型与A相同

例子：

1. Int a[N][M]://创建N\*M个元素
   1. N 个 类型与int B[M]中B相同的元素（下同）
      1. M 个 int C
2. Int (\*p)[M]://创建一个元素
   1. 一个指针，指向类型 int A[M]
      1. M 个 int B
3. Int \*p[M]://创建M个元素
   1. M 个 int \*A
      1. 一个指针， 指向 int B
4. Int (\*p[N])[M];//创建N个元素
   1. N 个 int (\*A)[M];
      1. 一个指针，指向int B[M]
         1. M个int C

问题分析：

1. 附加1，为什么Main()中需要赋值M次，main\_v2()赋值1次？
2. 上面例子1, 2中a, p类型是什么关系？
3. 附加1中，p 与 &a[N] 类型关系。
4. Int a[N]与int\*p中，a,p类型，以及创建过程的区别。

**附加4：**

c语言中左值、右值定义：

例子：

a = b + 25;

1

左值：能够出现在赋值符号左侧的东西。例子中a,

右值：能够出现在赋值符号右侧的东西。例子中b + 25

上面的例子可以写成这样吗？

b + 25 = a;

1

答案是否定的。原因如下：

（1）a是个左值，因为它标识了一个可以存储结果值的地点；b + 25是个右值，因为它指定了一个值。

（2）a是个右值，因为它可以指定一个值；b + 25不能作为左值，因为它并未标识一个可以存储结果值的地点。因此，这条赋值语句是非法的。

讲述上面概念只是为了引出指针左值和右值的概念，若想详细了解左右值本身可参见《C和指针》5.3节。

指针的左值和右值

先看下列代码中的表达式：

int a = 112, b = 1;

int \*d = &a;

b = \*d;

1

2

3

问题来了，你能准确写出下列表达式的值吗？

a

b

d

\*d

a 和 b 的值分别为112，1。无须解释；

d的值分为两种情况：

d左值（当d出现在赋值符号左侧时）：表达式 d 表示的是存储d这个指针变量本身的内存地址。

d右值：表达式 d 表示的是存储变量a的内存地址。

\*d的值也分为两种情况：

\*d左值：表达式 \*d 表示的是存储变量a的内存地址。

\*d右值：表达式 \*d 表示的是存储变量a的内存地址中的所存的值。

综述：

- 左值表示的是可以存储结果值的内存地址。

- 右值表示的是结果值。

---------------------

作者：xiaoyu-Wang

来源：CSDN

原文：https://blog.csdn.net/u014656992/article/details/49863605

版权声明：本文为博主原创文章，转载请附上博文链接！