# 指针变量的声明和使用

## 指针的概念

为了正确地访问这些数据，必须为**每个字节都编上号码**，就像门牌号、身份证号一样，每个字节的编号是唯一的，根据编号可以准确地找到某个字节。

下图是 4G 内存中每个字节的编号（以十六进制表示）：



我们将内存中字节的编号称为**地址（Address）或指针（Pointer）**

#include<stdio.h>

#include<stdlib.h>

int main(void)

{

int a = 100;

char str[20] = "c.biancheng.net";

printf("%#X, %#X\n", &a, str);

system("pause");

return 0;

}

上述代码中，%#X表示以十六进制形式输出，并附带前缀0X。a 是一个变量，用来存放整数，需要在前面加&来获得它的地址；str 本身就表示字符串的首地址，不需要加&。

## 指针变量的声明

### 指针变量的定义

数据在内存中的地址也称为**指针**，如果一个**变量存储了一份数据的指针，我们就称它为指针变量。**

在C语言中，允许用一个变量来存放指针，这种变量称为**指针变量**。**指针变量的值就是某份数据的地址，这样的一份数据可以是数组、字符串、函数，也可以是另外的一个普通变量或指针变量。**

**（1）**int \*p1;// p1是一个指向 int 类型数据的指针变量，至于 p1 究竟指向哪一份数据，应该由赋予它的值决定。

**（2）**int a = 100;int \*p\_a = &a;// **在定义指针变量 p\_a 的同时对它进行初始化**，并将变量 a 的地址赋予它，此时 p\_a 就指向了 a。值得注意的是，p\_a 需要的一个地址，a 前面必须要加取地址符&，否则是不对的。

### P和\*p的注意事项

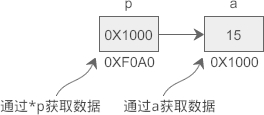
1. //定义普通变量
2. float a = 99.5, b = 10.6;
3. char c = '@', d = '#';
4. //定义指针变量
5. float \*p1 = &a;
6. char \*p2 = &c;
7. //修改指针变量的值
8. p1 = &b;
9. p2 = &d;

**（1）\*是一个特殊符号，表明一个变量是指针变量，定义 p1、p2 时必须带\*。给指针变量赋值时不能带\*。**（p是一个指针变量，而\*p代表着这是一个指针，p的值就是所存储数据的地址！）

**（2**）需要强调的是，**p1、p2 的类型分别是float\*和char\***，而不是float和char，它们是完全不同的数据类型，需要引起注意。

1. #include <stdio.h>
2. int main(){
3. int a = 15, b = 99, c = 222;
4. int \*p = &a; //定义指针变量
5. \*p = b; //通过指针变量修改内存上的数据
6. c = \*p; //通过指针变量获取内存上的数据
7. printf("%d, %d, %d, %d\n", a, b, c, \*p);
8. **return** 0;
9. }

假设变量 a、p 的地址分别为 0X1000、0XF0A0，它们的指向关系如下图所示：



程序被编译和链接后，**a、p 被替换成相应的地址。使用 \*p 的话，要先通过地址 0XF0A0 取得变量 p 本身的值，这个值是变量 a 的地址，然后再通过这个值取得变量 a 的数据，前后共有两次运算**（**可以间接的改变a 的值**）；**而使用 a 的话，可以通过地址 0X1000 直接取得它的数据，只需要一步运算。**

## 关于 \* 和 & 的谜题

假设有一个 int 类型的变量 a，pa 是指向它的指针，那么\*&a和&\*pa分别是什么意思呢？

\*&a可以理解为\*(&a)，&a表示取变量 a 的地址（等价于 pa），\*(&a)表示取这个地址上的数据（等价于 \*pa），绕来绕去，又回到了原点，**\*&a仍然等价于 a。**&\*pa可以理解为&(\*pa)，\*pa表示取得 pa 指向的数据（等价于 a），&(\*pa)表示数据的地址（等价于 &a），**所以&\*pa等价于 pa**。

## 指针变量的使用

**注：C语言中所有的变量都是要占据内存的，并且其占据内存大小是由变量类型所决定的。**

**注：所有的指针变量都占据相同大小的内存，在32位电脑上，指针变量占4个字节。（指针是有类型的，指针变量虽然都占4个字节，它的类型含义：该变量只能存放它所对应的类型去掉一个\*后的那个类型的变量或常量的地址，例如：int \* p，则p只能存放int的变量或常量的地址。）**

**）**

1. **例 设int x = 3; int \* p = &x;，请分析下述表达式的值：x、&x、p、&p、\*p。（**对于%p一般以[十六进制](https://www.baidu.com/s?wd=%E5%8D%81%E5%85%AD%E8%BF%9B%E5%88%B6&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dWryf4mWKhnWcdnjnLmvm0IAYqnWm3PW64rj0d0AP8IA3qPjfsn1bkrjKxmLKz0ZNzUjdCIZwsrBtEXh9GuA7EQhF9pywdQhPEUiqkIyN1IA-EUBtknWRYPjRvrHf)整数方式输出指针的值）

#include<stdio.h>

#include<stdlib.h>

int main(void)

{

int x=3;

int \*p=&x;

printf("%d\n",x);

printf("%p\n",&x);

printf("%p\n",p);

printf("%p\n",&p);

printf("%d\n",\*p);

system("pause");

return 0;

}

**（2）注意：当指针类型与赋值变量的类型不同时会发生内存溢出或者内存截断的问题.**

#include<stdio.h>

#include<stdlib.h>

int main(void)

{

int x=0x89898;

double \*q=(double\*)&x;

printf("%lf\n",\*q);

printf("%d\n",\*q);

system("pause");

return 0;

}

# 指针变量的运算

指针的运算关系：

（1）表示该指针（操作数）所指向的内存空间（\*）

（2）取该“变量”（操作数）地址（&）

（3）指针的赋值（=）赋值运算包括两种情况：

**将变量地址赋给指针----指向变量**

**将指针赋给指针----与被赋值的指针指向同一个变量**

（4）指针的加减运算（+、-）

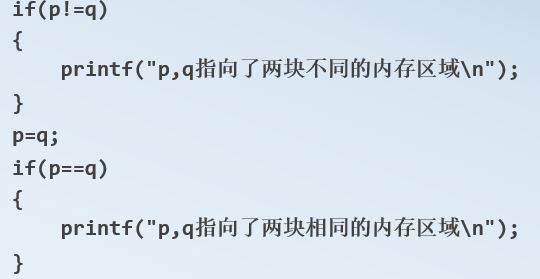
**指针的加减运算中是以指针的类型的长度为单位的。指针并不是都可以进行加减**

**和比较运算的，只有指向数组的指针加减运算才有意义。指针的加减运算是以其所指类型的字节长度为单位的**

**p+1 <==> p 移动 sizoef(所指类型）字节**

（5）指针的关系运算（<、>、==）

**设有int x = 3, y = 4; int \* p = &x, \* q = &y; 请依次观察下面的语句。**

****

# 多重指针的声明和使用

指针可以指向一份普通类型的数据，例如 int、double、char 等，也可以**指向一份指针类型的数据**，例如 int \*、double \*、char \* 等。

**若指针的存储内容为另一个指针的地址。则成称这个指针为多重指针。**

假设有一个 int 类型的变量 a，p1是指向 a 的指针变量，p2 又是指向 p1 的指针变量，它们的关系如下图所示：  
http://c.biancheng.net/cpp/uploads/allimg/160714/1-160G41Q12O23.jpg

#include<stdio.h>

#include<stdlib.h>

int main(void)

{

int x=3;

int \*p=&x;

int \*\*q=&p;

printf("%d\n",&x);

printf("%d\n",\*\*q);

system("pause");

return 0;

}

# 指针和数组

## 数组指针

### 数组指针的声明

**数组指针（也称行指针）  
定义 int (\*p)[n];**

**Int （\*p）[4]=&a;//此时a也表示一个数组**( )优先级高，首先说明p是一个指针，指向一个整型的一维数组，这个一维数组的长度是n，也可以说是p的步长。也就是说执行p+1时，p要跨过n个整型数据的长度。

**如要将二维数组指针进行声明，应这样赋值：  
int （\*a）[3][4];**

**所以数组指针也称指向一维数组的指针，亦称行指针。**

### 数组指针的运算

**int \*p, a[10];**

**p** **=** **a;**

**p++;**

**（1）p** **=** **a;在C语言中，a代表的由两个含义，一方面代表着这是一整个数组，另一方面代表着这是&a[0],表示数组的第一个元素a[0]的地址。**

**（2）指针的加减运算是以其所指类型的字节长度为单位的**

**p+1 <==> p 移动 sizoef(所指类型）字节**

**条件**：p,q是指向同一数组。可以进行指针和整数的加减运算；同类型指针之间的减法运算

|  |  |
| --- | --- |
| 运算方式 | 说明 |
| **p+n** | p之后第n个元素的地址 |
| **p-n** | p之前第n个元素的地址 |
| **p++** | p作为当前操作数，然后后移一个元素 |
| **++p** | p后移一个元素，然后作为当前操作数 |
| **p--** | p作为当前操作数，然后前移一个元素 |
| **--p** | p前移一个元素，然后作为当前操作数 |
| **p-q** | 表示p和q两者之间的元素个数 |

**（3）条件：指向同一数组的两个指针才能进行关系运算**

p<q、p<=q、p==q、p!=q、p>=q、p>q

**p<q:判断p所指元素是否在q所指元素之前**

其他运算的含义与上述类似

**（4）不能与非指针类型变量进行比较,但可与NULL(即0值)进行等或不等的关系运算**

判断p是否为空指针

P == NULL p != NULL

### 数组元素的几种等价形式

**a[ i ]** **\*(p+i)**  **\*(a+i) p[i]**

**表示数组的两种方法：**

**（一）下标法：**

**#include** **<stdio.h>**

**int** **main(void)**

**{**

**int** **a[4]** **=** **{1,2,3,4};**

**int** **\*** **pa** **=** **a,** **i;**

**for(i=0;i<4;i++)**

**{**

**printf("a[%d]:%d\n",i,a[i]);**

**}**

**for(pa** **=** **a,i** **=** **0;** **i** **<** **4;i++)**

**{**

**printf("pa[%d]:%d\n",i,pa[i]);**

**}**

**return** **0;**

**}**

**（二）指针法**

**#include <stdio.h>**

**int main(void)**

**{**

**int a[4] = {1,2,3,4};**

**int \*pa, i;**

**for(i=0;i<4;i++)**

**{**

**printf("\*(a+%d):%d\n",i,\*(a+i));**

**}**

**for(pa = a;pa < a + 4;pa++)**

**{**

**printf("\*pa:%d\n",\*pa);**

**}**

**return 0;**

**}**