C语言程序设计教程

第8章　指针

返回

8.1 指针与指针变量

8.1.1 指针的概念

1. 内存与变量地址

- 内存地址：内存是计算机用于存储数据的存储器，以一种字节作为存储单元，为了便于访问，给每个字节单元一种唯一的编号，第一字节单元编号为0，后来各单元按顺序连续编号，这些单元编号称为内存单元的地址 。

- 变量地址：变量所分配存储空间的首字节单元地址（字节单元编号）。

2. 变量的三要素：名字、类型与值

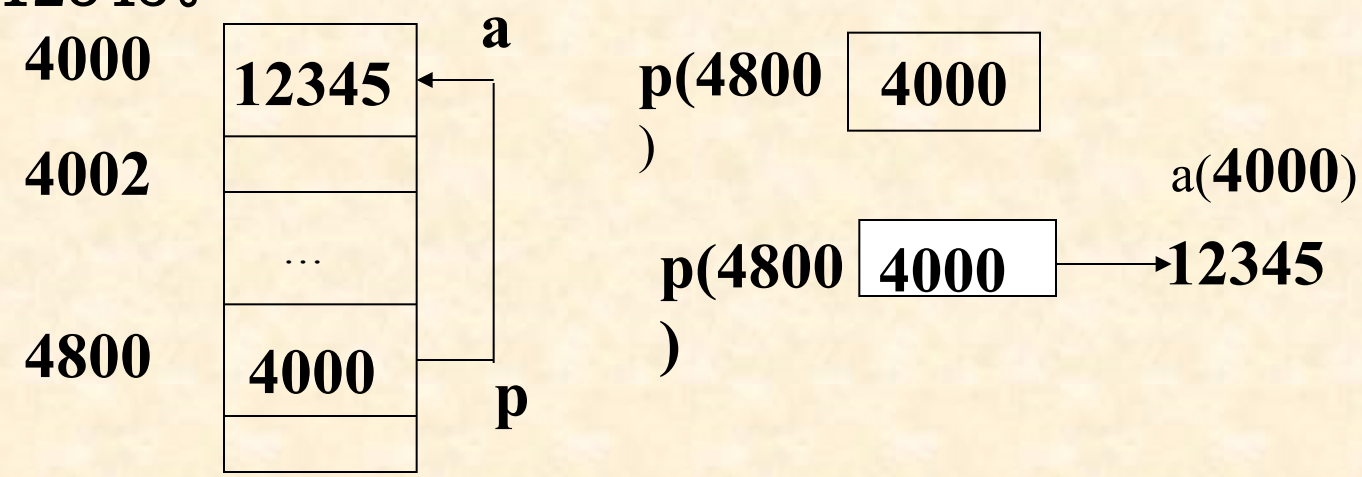
- 每个变量都经过变量名与相应的存储单元相连系，详细分配哪些单元给变量，由C编译系统完毕变量名到相应内存单元地址的变换。
- 变量分配存储空间的大小由类型决定。
- 变量的值则是指相应存储单元的内容。

3. 内存存取方式

- 直接存取：把直接按变量名或地址存取变量值的方式称为“直接存取”方式。
- 间接存取：经过定义一种特殊的变量专门存储内存或变量的地址，然后根据该地址值再去访问相应的存储单元。

系统为特殊变量p（用来存储地址的）分配的存储空间地址是4800，p中保存的是变量a的地址,即4000，当要读取a变量的值12345时，不是直接经过a变量,也不是直接经过保存12345的内存单元的地址4000去取值，而是先经过变量p得到p的值4000，即a的地址，再根据地址4000读取它所指向单元的值12345。

间接存取示意图右



这种间接的经过变量p得到变量a的地址，再存取变量a的值的方式即为“间接存取”。

一般称变量p指向变量a，变量a是变量p所指向的对象

4.指针的概念

- 在C语言中，用指针来表达一种变量指向另一种变量这么的指向关系。
- 所谓指针即地址。
- 一种变量的指针即该变量的地址，如4000就是指向变量a的指针。
- 指针变量：专门存储地址的变量，如p即是一种指针变量,它存储的是a的地址4000。

8.1.2 指针变量的定义与初始化

1. 指针变量的定义

类型标识符 *指针变量名;

例：float *p1; (定义p1为指向实型变量的指针变量)

char *p2; (定义p2为指向字符型变量的指针变量)

- 在指针变量定义中，*是一种阐明符，它表白其后的变量是指针变量，如p是指针变量,而不要以为“*p”是指针变量。
- 指针变量定义时指定的数据类型不是指针变量本身（变量存储的值）的数据类型，而是指针变量所指向的对象（或称目的）的数据类型
- 指针变量存储的是所指向的某个变量的地址值，而一般变量保存的是该变量本身的值
- 指针变量并不固定指向一种变量，可指向同类型的不同变量

2. 指针变量初始化

(1) 指针运算符与地址运算符

与指针引用有关的两个运算符：&与*。

&:取地址运算符

***：指针运算符，或称指向运算符、间接访问运算符。**

指针指向的对象的表达形式： *指针变量

此处*是访问指针所指对象的运算符,与指针定义时的*不同。

(2) 指针变量初始化

若有定义： int a,*p;

语句仅仅定义了指针变量p，但指针变量并未指向拟定的变量(或内存单元)。因为这些指针变量还没有赋给拟定的地址值，只有将某一详细变量的地址赋给指针变量之后，指针变量才指向拟定的变量（内存单元）。

指针变量初始化:在定义指针时同步给指针一种初始值

如： int a,*p=&a;

(3) 指针变量的引用

① *指针变量名——代表所指变量的值。

② 指针变量名——代表所指变量的地址。

有定义： int a,*p=&a;

用*p来表达p指向的对象a,*p与a是等价的。

*p能够象一般变量一样使用。 例如：

```
a=12;
*p=12;
scanf("%d",&*p);    scanf("%d",p);
printf("%d%d",*p,a);
```

注意：*与&具有相同的优先级，结合方向从右到左。这么，&*p即&(*p),是对变量*p取地址，它与&a等价；p与&(*p)等价，a与*(&a)等价。

8.1.3 指针运算

1. 指针的赋值运算

（1）将变量地址值赋给指针变量，使指针指向该变量。

设有如下定义：

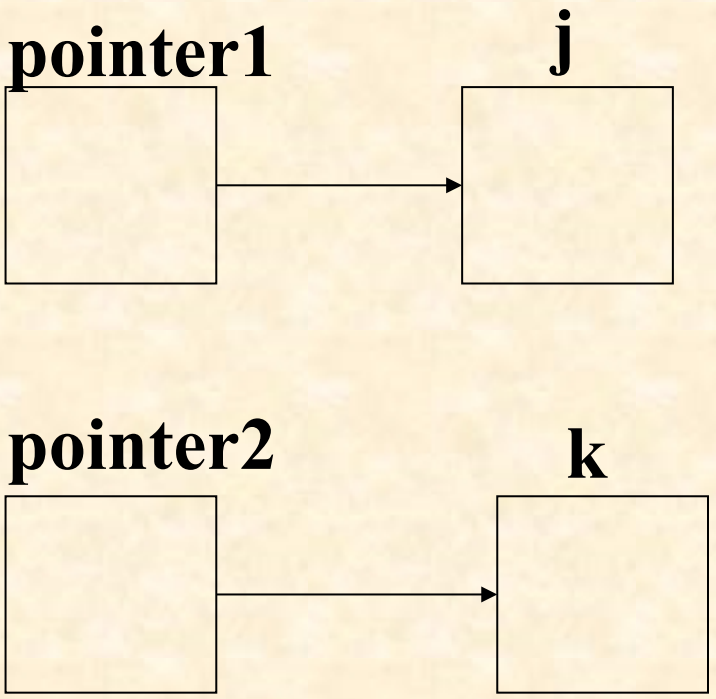
```
int a,b,*pa,*pb;
float *pf;
```

第一行定义了整型变量a,b及指针变量pa,pb。pa、pb还没有被赋值，所以pa、pb没有指向任何变量，下面语句完毕对pa,pb的赋值：

```
pa=&a;
pb=&b;
```

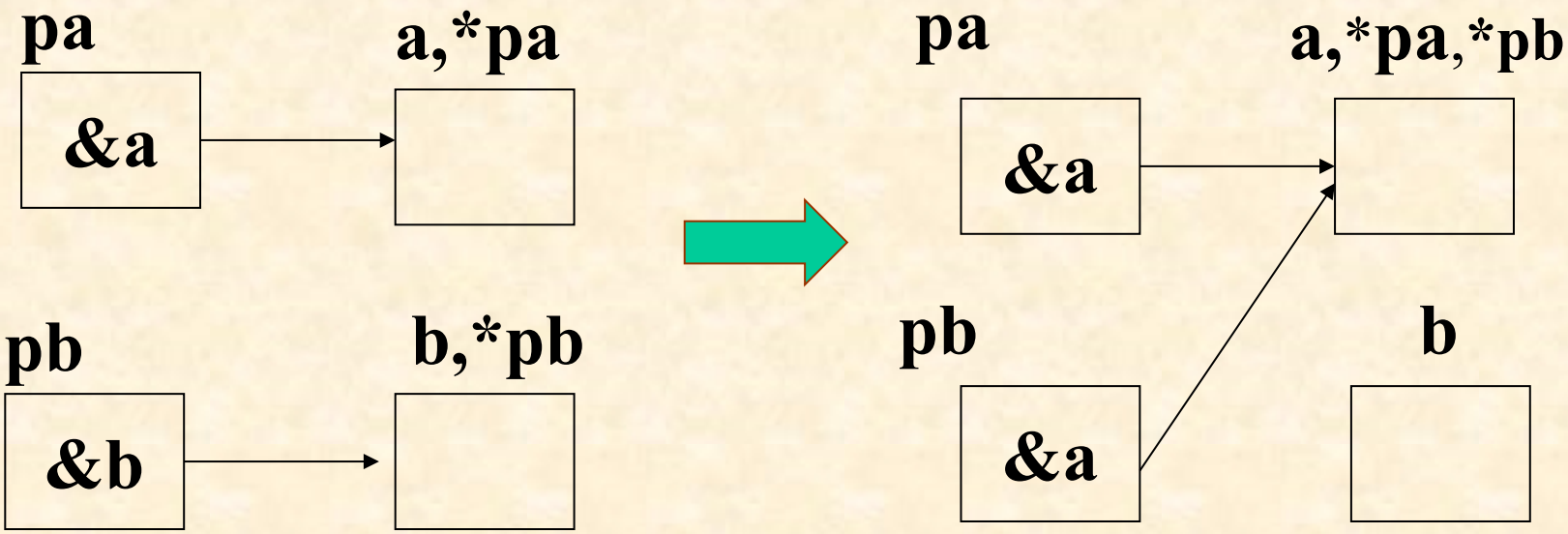
例如：

```
int j,k;
int *pointer1,*pointer2;
pointer1=&j;
pointer2=&k;
```



(2) 相同类型的指针变量间的赋值

pa与pb都是整型指针变量，它们间能够相互赋值，如：pb=pa; 即 pa,pb都指向变量a，此时a、*pa、*pb是等价的。指针指向变化如下图:



注意：只有相同类型的指针变量才干相互赋值，如pf=pa;是不允许的。因为pa是整型指针,pf是浮点型指针。

（3）给指针变量赋空值

给指针变量赋空值，阐明该指针不指向任何变量。

“空”指针值用NULL表达，NULL是在头文件stdio.h中预定义的常量，其值为0，在使用时应加上预定义行,如：

```
#include "stdio.h"
int  *pa=NULL;
```

亦能够用下面的语句给指针赋“空值”：

```
pa=0;  或:   pa='\0';
```

这里指针pa并非指向0地址单元，而是具有一种拟定的“空值”，表达pa不指向任何变量。

注意：指针虽然能够赋值0，但却不能把其他的常量地址赋给指针。例如：　pa=4000;　是非法的。

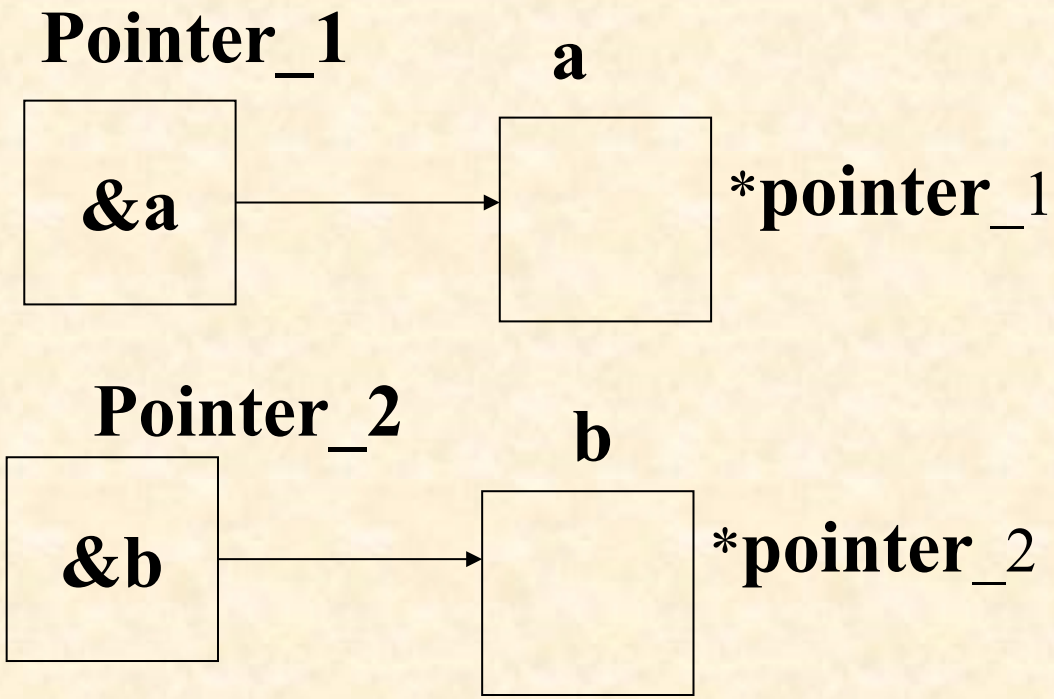
例 8.1 指针定义与初始化

```
main()
{int a,b;
int *pointer_1,*pointer_2;
a=100;b=10;
pointer_1=&a;
pointer_2=&b;
printf ("%d,%d\n",a,b);
printf("%d,%d\n",*pointer_1,*pointer_2);
}
```

程序运营成果：

100，10

100，10



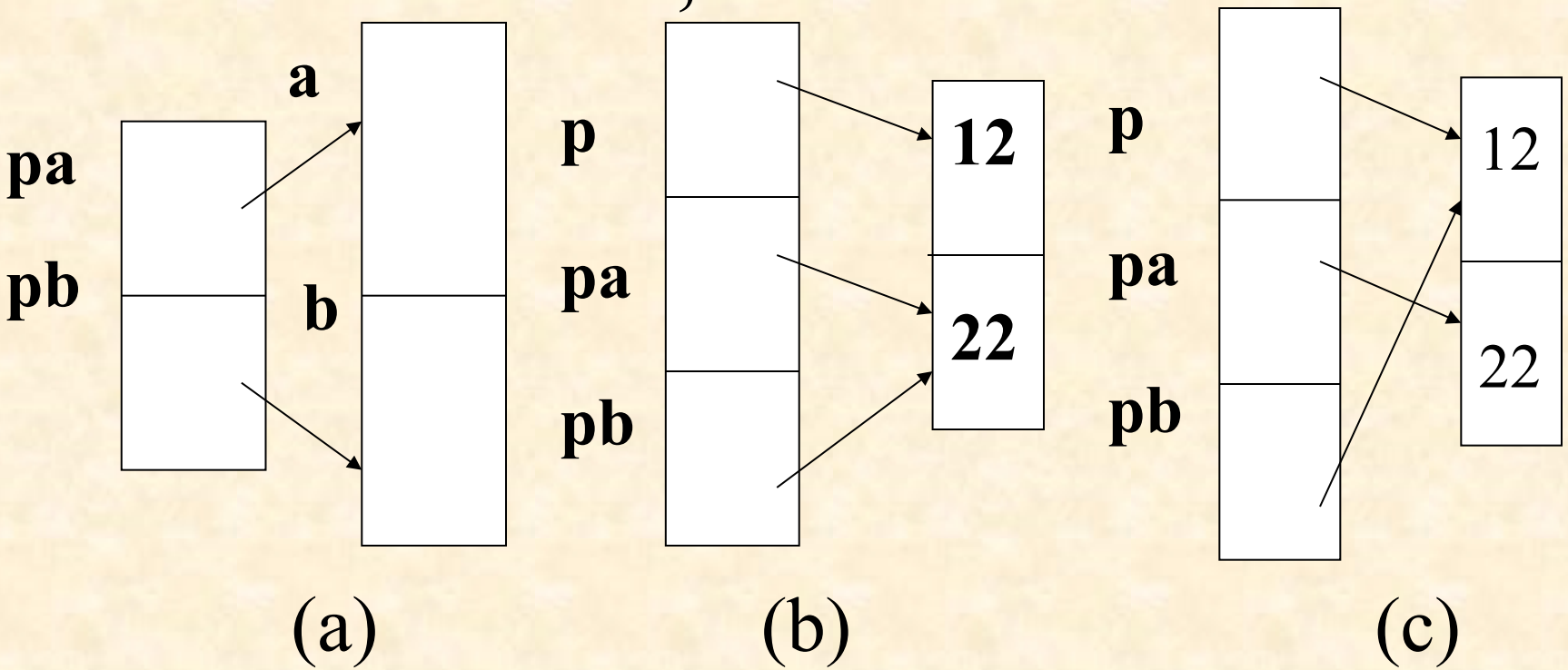
例8.2 从键盘上输入两个整数到a、b,按由大到小输出。

```
#include <stdio.h>
    main( )
 { int a,b,*pa=&a,*pb=&b,*p;/*定义指针变量pa、pb,如下页图a*/
    scanf("%d%d",&a,&b);
 if (*pa＜*pb) {  p=pa;  /*进行指针互换,如下页图b,c*/
                  pa=pb;
                   pb=p; }
      printf("\n a=%d,b=%d\n",a,b);
      printf("\n max=%d,min=%d",*pa,*pb);
                       /*  pa指向大数，pb指向小数*/
}
```

若输入： 12 22↙

输出成果：a=12,b=22

max=22,min=12



指针变化示意图

2. 指针的算术运算

(1) 加减运算: 一种指针能够加、减一种整数n，其成果与指针所指对象的数据类型有关。指针变量的值应增长或降低“n×sizeof(指针类型)”。

加减运算常用于数组的处理。对指向一般数据的指针，加减运算无实际意义。例如：

```
int  a[10], *p=a, *x;
x=p+3;  /*实际上是p加上3*2个字节赋给x，x指向数组的第三个分量*/
```

对于不同基类型的指针，指针变量“加上”或“减去”一种整数n所移动的字节数是不同的。例如：

```
float  a[10],  *p=a,  *x;
p=p+3;  /*实际上是p加上3*4个字节赋给x，x依然指向数组的第三个分量*/
```

(2) 自增自减运算

指针变量自增、自减运算具有上述运算的特点，但有前置后置、先用后用的考虑，务请小心。例如：

```
int  a[10],  *p=a,  *x;

x=p++/* x第一种元素分量，  p指向第二个元素*/
x=++p;  /* x、  p均指向数组的第二个分量*/
```

* p++相当于*（p+ +）。

*（p++）与（*p）++ 含义不同，前者表达地址自增，后者表达目前所指向的数据自增。

思索

1.若有定义 int a,*p; 执行了“p=&a”,则：
“&*p”的含意是什么？

(答：相当于&a)

2. *&a的含意是什么？

(答：a)

3. (*p)++相当于什么？

(答：a++)

3. 指针的关系运算

- 与基本类型变量一样，指针能够进行关系运算。

- 在关系体现式中允许对两个指针进行全部的关系运算。若 p,q是两个同类型的指针变量，则：p>q,p<q,p==q,p!=q，p>=q都是允许的。

- 指针的关系运算在指向数组的指针中广泛的利用，假设p、q是指向同一数组的两个指针，执行p＞q的运算，其含义为，若体现式成果为真（非 0 值），则阐明p所指元素在q所指元素之后。或者说q所指元素离数组第一种元素更近些。

注意：在指针进行关系运算之前，指针必须指向拟定的变量或存储区域，即指针有初始值；另外，只有相同类型的指针才干进行比较。

8.1.4 多级指针

把指向指针型数据的指针变量称为指向指针的指针，或称多级指针。

二级指针的定义形式如下：

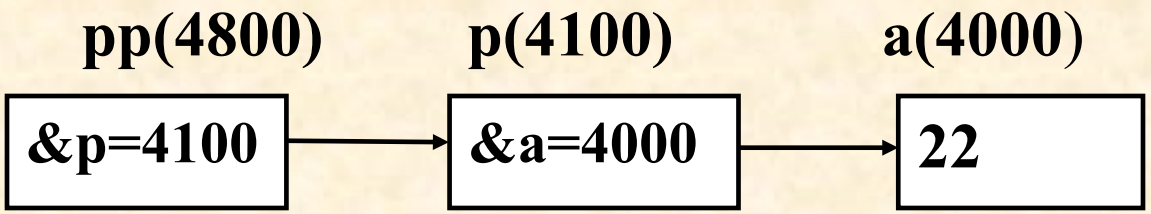
数据类型 **指针变量

例如：int a,*p,**pp;

a=22;

p=&a;

pp=&p;



假设变量a的地址为4000，指针p的地址为4100，二级指针pp的地址为4800。a、p、pp三者的关系如上图。

8.2 指针与函数

8.2.1 指针作为函数参数

利用指针作函数参数，能够实现函数之间多种数据的传递，当形参为指针变量时，其相应实参能够是指针变量或存储单元地址。

函数形参为指针变量，用指针变量或变量地址作实参

例8.3 编写一种互换两个变量的函数，在主程序中调用，实现两个变量值的互换。

```
#include<stdio.h>
main()
{  int  a,b;
   int  *pa,*pb;
void swap(int *p1,int *p2);  /*函数申明*/
```

```
scanf("%d%d",&a,&b);
   pa=&a;              /* pa指向变量a */
   pb=&b;              /* pb指向变量b  */
   swap(pa,pb);
   printf("\na=%d,b=%d\n",a,b);
}
void swap(int *p1,int *p2)
 {int  temp;
   temp=*p1; /*互换指针p1、p2所指向的变量的值*/
   *p1=*p2;
   *p2=temp;
   }
```

或：swap(&a,&b);

程序运营成果如下：

输入：　12　22↙

输出：　a=22，b=12

两点阐明

(1) 若在函数体中互换指针变量的值，实参a、b的值并不变化，指针参数亦是传值。 例如：

```
int *p;

p=p1; p1=p2; p2=p;

不要希望如此完毕处理。
```

(2) 函数中互换值时不能使用无初值的指针变量作临时变量。 例如： int *p;

```
*p=*p1; *p1=*p2; *p2=*p;
```

p无拟定值，对 p的使用可能带来不可预期的后果。

8.2.2 指针函数

指针函数:是指返回值为指针的函数

指针函数的定义形式:

类型标示符 *函数名（参数）

例如:

```
int *fun(int a,int b)
 {
     函数体语句
 }
```

在函数体中有返回指针或地址的语句，形如:

return (&变量名); 或 return (指针变量);

而且返回值的类型要与函数类型一致。

例8.3 分析如下程序

```
main( )
{   int a,b,*p;
    int  *max(int x,int y);

scanf("%d,%d",&a,&b);
   p=max(a,b);
   printf("max=%d",*p);
 }
```

```
int  *max(int x,int y)
{ if x>y) return  (&x);
  else  return  (&y);
 }
```

8.2.3 指向函数的指针

- 一种函数涉及一组指令序列，存储在某一段内存中，这段内存空间的起始地址称为函数的入口地址
- 称函数入口地址为函数的指针。函数名代表函数的入口地址
- 能够定义一种指针变量，其值等于该函数的入口地址，指向这个函数，这么经过这个指针变量也能调用这个函数。这种指针变量称为指向函数的指针变量。

定义指向函数的指针变量的一般形式为：

类型标识符（*指针变量名）( )；

例：int (*p)(); /* 指针变量p能够指向一种整型函数*/

float (*q)(); /* 指针变量q能够指向一种浮点型函数*/

- 刚定义的指向函数的指针变量，亦象其他指针变量一样要赋以地址值才干引用。当将某个函数的入口地址赋给指向函数的指针变量，就可用该指针变量来调用所指向的函数
- 给函数指针赋初值：将函数名（函数的入口地址值）赋给指针变量

例如 int m, (*p)( );

int max(int a,int b);

则能够 p=max; /* p指向函数max() */

指针调用函数的 一般形式为：

(*指针变量)(实参表);

如上例：m=(*p)(12,22); /*比较 m=max(12,22); */

注意

- 用函数指针调用函数是间接调用，没有参数类型阐明，C编译系统也无法进行类型检验，所以，在使用这种形式调用函数时要尤其小心。实参一定要和指针所指函数的形参类型一致。
- 函数指针能够作为函数参数，此时，当函数指针每次指向不同的函数时，可执行不同的函数来完毕不同的功能

例 8.4 函数max()用来求一维数组的元素的最大值，在主调函数中用函数名调用该函数与用函数指针调用该函数来实现。

```
#include "stdio.h"
  #define M 8
  main()
{float sumf,sump;
  float a[M]={11,2,-3,4.5,5,69,7,80};
  float (*p)();          /*定义指向函数的指针p*/
  float  max(float a[ ],int n);  /*函数申明*/
  p=max;  /*函数名（函数入口地址）赋给指针p*/
```

```
sump=(*p)(a,M);     /*用指针方式调用函数*/
sumf=max(a,M);      /*用函数名调用max()函数*/
      printf("sump=%.2f\n",sump);
      printf("sumf=%.2f\n",sumf);
      }
float max(float a[],int n)
 {    int k;
        float s;
        s=a[0];
        for (k=0;k<n;k++)
           if (s<a[k]) s=a[k];
   return s;
}
```

程序运营成果：

sump=80.00

sumf=80.00

指向函数的指针的使用环节

（1） 定义一种指向函数的指针变量，形如：

```
float (*p)();
```

（2） 为函数指针赋值，格式如下：

```
p=函数名;
```

注意：赋值时只需给出函数名，不要带参数。

（3） 经过函数指针调用函数，调用格式如下：

```
s=(*p)(实参);
```

8.3 指针与数组

8.3.1 指向一维数组的指针

数组名是一种常量指针，它的值为该数组的首地址

1.指向数组的指针的定义措施与指向基本类型变量的指针的定义措施相同，例如：

```
int a［10］=｛1,3,5，7，9｝;
int *p;
p=&a[2]; (把数组元素a[2]的地址赋给指针变量p)
p=a;  (把数组的首地址赋给指针变量p)
```

❖C语言要求：数组名代表数组首地址,是一种地址常量。

所以，下面两个语句等价：

```
p=&a[0];
p=a;
```

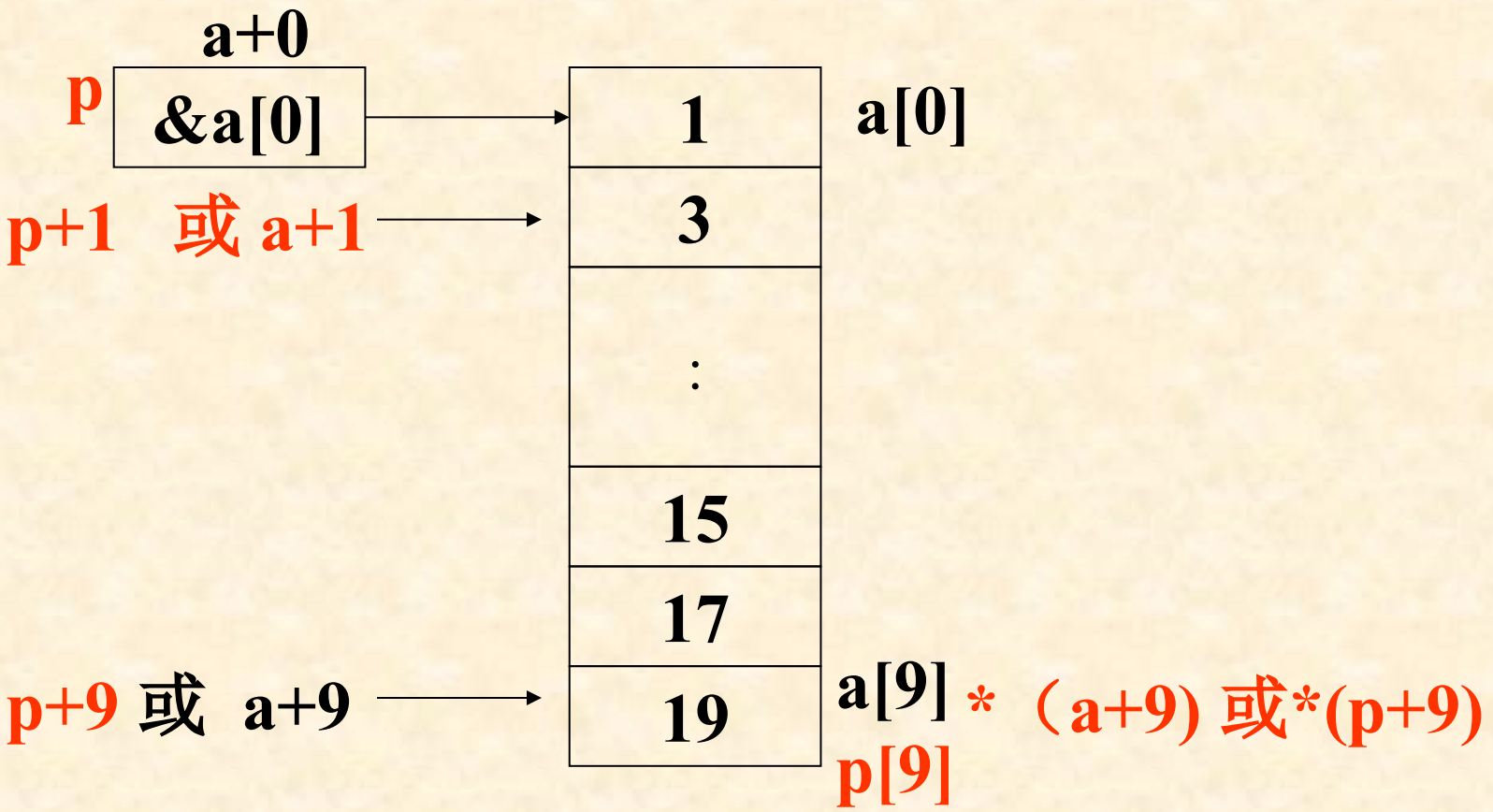
❖在定义指针变量的同步可赋初值：

```
int  a[10], *p=&a[0]; (或 int *p=a;)
```

等价于：

```
int  *p;
p=&a[0];   两句
```

指向数组的指针变量p



以上内容仅为本文档的试下载部分，为可阅读页数的一半内容。如要下载或阅读全文，请访问：https://d.book118.com/355141222212011302