高级语言程序设计

主讲：王老师

尚德机构

学习是一种信仰

第七章　指针

本章内容

- 指针和指针变量
- 指针和数组
- 指针和字符串
- 指针和函数
- 指针数据
- 指针的程序设计举例

第一节 指针和指针变量

指针：

C语言中指针是一种数据类型。指针是存放数据的内存单元地址。

计算机系统的内存拥有大量的存储单元（每个存储单元的大小为1字节），为了便于管理，必须为每个存储单元编号，该编号就是存储单元的“地址”。每个存储单元拥有一个唯一的地址。

指针变量除了可以存放变量的地址外，还可以存放其他数据的地址，例如可以存放数组和函数的地址。

指针变量的定义和初始化

指针变量定义的一般形式：

【格式】数据类型符 *指针变量名[=初始地址值],...;

【功能】定义指向“数据类型符”的变量或数组的指针变量，同时为其赋初值。

【说明】

1、“*”表示定义的是一个指针变量。指针变量的前面必须有“*”号。

2、在定义指针变量的同时也可以定义普通变量或数组等。

3、“数据类型符”是指针变量所指向变量的数据类型，可以是任何基本数据类型，也可以是其他数据类型。

指针变量的定义和初始化

4、“初始地址值”通常是“&变量名”“&数组元素”或“一维数组名”，这里的变量或数组必须是已定义的。

5、在定义指针变量时，可以只给部分指针变量赋初值。

6、指针变量的初始化，除了可以是已定义变量的地址，也可以是已初始化的同类型的指针变量，也可以是NULL（空指针）。

7、指针变量初始化时，指针变量的“数据类型符”必须与其“初始地址值”中保存的数据的类型相同。

★指针变量的初始化

一般形式：[存储类型] 数据类型 *指针名=初始地址值;

```
例   int i;
     int  *p=&i;
```

赋给指针变量，
不是赋给目标变量

变量必须已说明过
类型应一致

```
例   int i;
     int  *p=&i;
     int  *q=p;
```

×

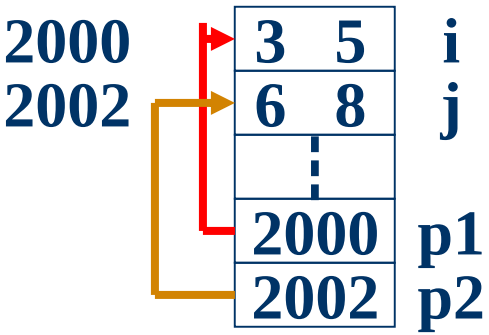
用已初始化指针变量作初值

```
例   main( )
     {   int  i;
         static  int   *p=&i;
         ..............
     }              (×)
```

不能用auto变量的地址
去初始化static型指针

指针变量赋值:

```
int  i, j;
int  *p1, *p2;
p1=&i; p2=&j ;
i=3; *p1=5;
j=6; *p2=8;
```



指针变量的一般使用

1、给指针变量赋值

【格式】指针变量=地址型表达式

其中，“地址型表达式”即运算结果是地址型的表达式。C语言规定，变量地址只能通过取地址运算符获得，即“&”，其运算对象是变量或数组元素名，运算结果是对应变量或数组元素的地址。

需要注意的是，虽然地址是一个整数，但是C语言中不允许把整数看成“地址常量”，所以此处的“地址型表达式”不能是整数。

指针变量的一般使用

2、直接使用指针变量

【格式】指针变量名

需要使用地址时，可以直接引用指针变量名。

3、通过指针变量引用所指向的变量

【格式】*指针变量名

C程序中，“*指针变量名”代表其指向的变量或数组元素，其中的“*”称为指针运算符。需要注意的是，这种引用方式要求指针变量必须已经定义且有值。

指针的基本运算

1、取地址运算符&

取地址运算符“&”的功能是取变量的地址，它是单目运算符。取地址运算符的运算对象必须是已经定义的变量或数组元素，但不能是数组名。运算结果是运算对象的地址。

指针的基本运算

2、指针运算符*

指针运算符“*”的功能是取指针变量所指向地址中的内容，与取地址运算符“&”的运算是互逆的，它是单目运算符。指针运算符的运算对象必须是地址，可以是已赋值的指针变量，也可以是变量或数组元素的地址，但不能是整数，也不能是非地址型的变量。运算结果就是地址对应的变量。

指针的基本运算

表 7-1　取地址运算符和指针运算符的说明

名称	运算符	运算对象个数	运算规则	运算对象类型	结果类型	结合性
取地址	&	单目前缀	获取运算对象的地址	变量或数组元素	运算对象的地址	从右至左
指针	*		获取运算对象对应的变量或数组元素	地址（指针变量，变量地址，数组元素的地址）	运算对象对应的数据	

取地址运算符和指针运算符的优先级和结合性：

1、取地址运算符、指针运算符和自增、自减等单目运算符的优先级相同。

2、所有单目运算符的结合性为从右至左。

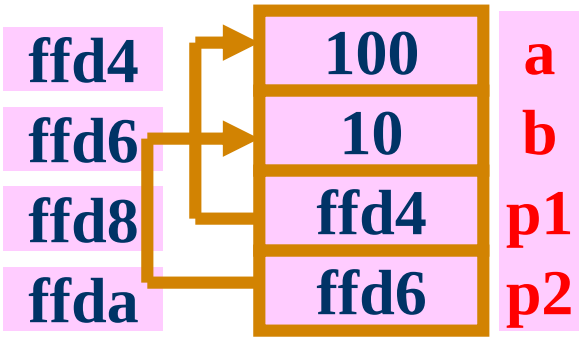
★指针变量的引用

例 通过指针变量访问整型变量

```
#include <stdio.h>
void main( )
{
   int  a, b, *p1, *p2 ;
   a=100; b=10;
   p1=&a; p2=&b;
   printf("a=%d, b=%d\ n",a, b);
   printf("* p1=%d, * p2=%d\ n", *p1, * p2);
   printf("&a=%x,& b=%x\ n",&a, &b);
   printf("p1=%x, p2=%x\ n", p1,  p2);
   printf("& p1=%x, &p2=%x\ n", &p1, & p2);
}
```

运行结果：

```
a=100, b=10
*p1=100, *p2=10
&a=ffd4, &b=ffd6
p1=ffd4, p2=ffd6
&p1=ffd8, &p2=ffda
```



★指针变量作为函数参数——地址传递

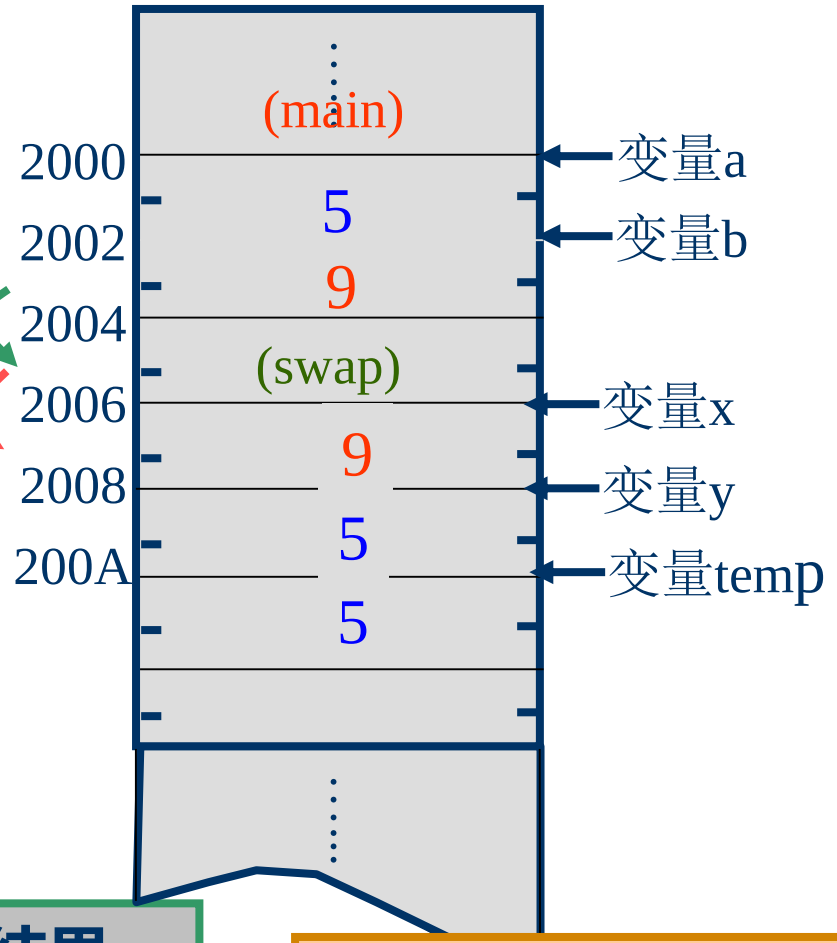
❖特点：共享内存,“双向”传递

例 将数从大到小输出

```
#include <stdio.h>
void swap(int  x,int y)
{
   int  temp;
   temp=x;
   x=y;
   y=temp;
}
void main()
{  int a,b;
    scanf("%d,%d",&a,&b);
    if(a<b)  swap(a,b);
    printf("\n%d,%d\n",a,b);
}
```

值传递

COPY



运行结果：

5 , 9 ↵

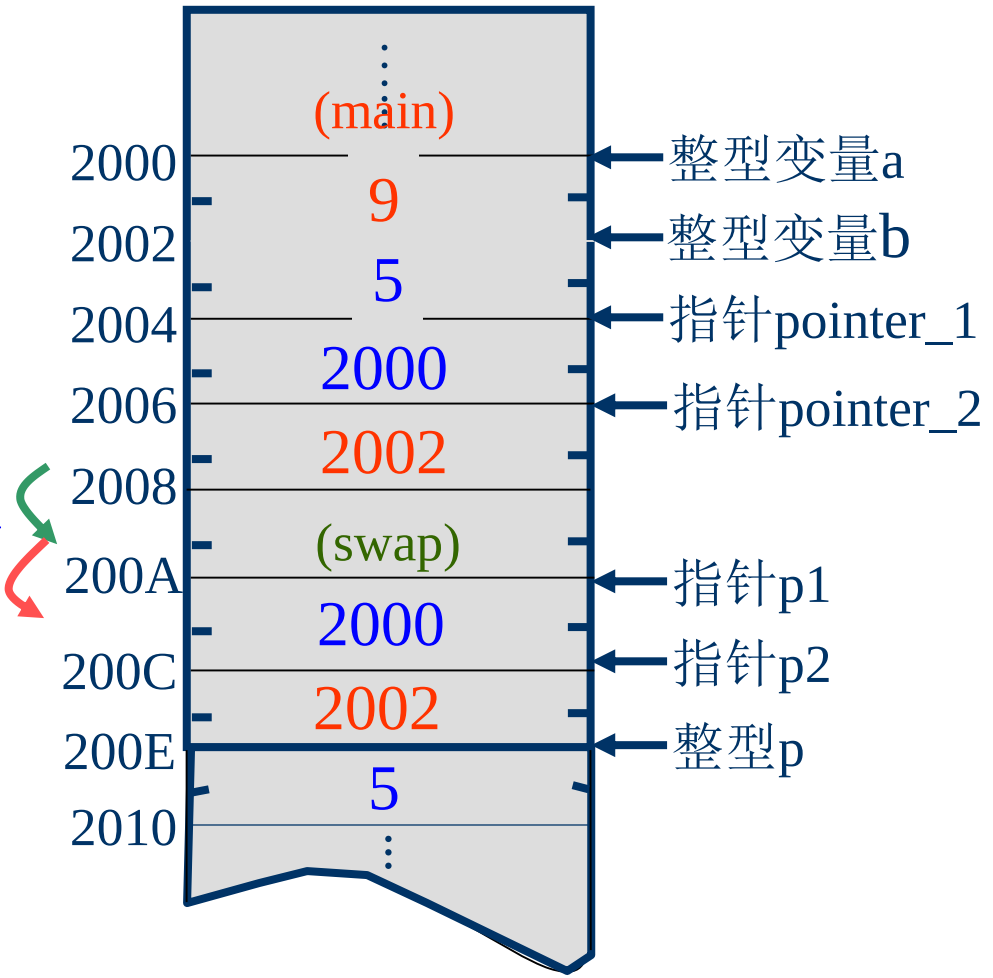
5 , 9

函数调用结束后，分配给x，y，temp单元释放

例 将数从大到小输出
（使用指针变量作函数参数）

```
#include <stdio.h>
void main()
{
   void swap(int  *p1, int  *p2);
   int a,b;
   int *pointer_1,*pointer_2;
   scanf("%d,%d",&a,&b);
   pointer_1=&a;  pointer_2=&b;
   if(a<b)swap(pointer_1,pointer_2);
   printf("\n%d,%d\n",a,b);}
void swap(int  *p1, int  *p2)
{  int temp;
    temp=*p1;
    *p1=*p2;
    *p2=temp;
}
```

地址传递



运行情况：

5,9 ↵

9,5

问题：函数调用结束后，分配给p1,p2,p单元释放否?

课堂练习

1、设int A=100, *p1=&A, **p2=&p1；则表达式**p2 的值是_____。

课堂练习

1、设int A=100, *p1=&A, **p2=&p1；则表达式**p2 的值是_____。

答案：100

解析：因为int A=100, *p1=&A, **p2=&p1;,所以最后**p2指向A的值，即表达式**p2 的值是100。

课堂练习

2、设int a=10,*p=&a; 则执行printf(〝%d\n〞,*p+a); 的结果是（　　）

A. 10

B. 20

C. 30

D. 40

课堂练习

2、设int a=10,*p=&a; 则执行printf(〝%d\n〞,*p+a); 的结果是（　　）

A. 10

B. 20

C. 30

D. 40

答案：B

解析：int a=10,*p=&a; ,定义指针变量a以及指针变量p，同时对指针变量p进行初始化，将指针变量p指向整型变量a，*p代表a的值。故在执行printf(〝%d\n〞,*p+a);输出的结果是20。

指针和数组

指针和一维数组：

数组的指针是指向数组在内存的起始地址，数组元素的指针是指向数组元素在内存的起始地址。

若将指针变量指向一维数组，可以采用以下两种方法：

1、在数据定义语句中用赋初值的方式，即*指针变量=数组名。

2、在程序中用赋值的方式，即指针变量=数组名;。

指针和数组

若将指针变量指向一维数组元素，可以采用以下两种方法：

1、在数据定义语句中用赋初值的方式，即*指针变量=&数组名[下标]。

2、在程序中用赋值的方式，即指针变量=&数组名[下标];。

当指针变量指向一维数组，利用指针变量引用一维数组元素的方法如下：

1、引用下标为0的数组元素 *(指针变量+0)或 *指针变量 或 指针变量[0]

2、引用下标为i的数组元素 *(指针变量+i) 或 指针变量[i]

指针和数组

当指针变量指向下标为i的一维数组元素时，利用指针变量引用数组元素的方法如下：

1、引用下标为i的数组元素 *(指针变量+0) 或 *指针变量

2、引用下标为i-k的数组元素 *(指针变量-k)

3、引用下标为i+k的数组元素 *(指针变量+k)

指针和数组

当指针变量指向一维数组时，引用下标为i的一维数组元素可以采用四种方法：

1、*(指针变量+i)

2、*(数组名+i)

3、指针变量[i]

4、数组名[i]

★指向数组元素的指针

例：int a[10];

int *p;

p=&a[0]；或 p=a; /*定义后赋值，两者等价*/

定义指针变量时赋初值：

例：int *p=&a[0];

int *p=a;

如 int i, *p;

p=1000; (×)

i=p; (×)

不能把一个整数⇒p,也不能把p的值⇒整型变量

★通过指针引用数组元素

如果： int a[10];

int *p;

p=&a[1]；/* p指向数组元素a[1] */

则： *p=1

表示对p当前指向的数组元素a[1]赋予值1

而：p+1指向同一数组的下一个元素a[2]。

p的值（地址）加了2个字节，p+1=p+1×d（整型，d=2；实型，d=4；字符型d=1）指针变量所指数组元素的地址的计算，与数组数据类型有关。

设 p=&a[0]

则 ⑴ p+i和a+i就是a[i]的地址a+i×d

⑵ *(p+i)或*(a+i)是p+i或a+i指向的数组元素a[i]

⑶ 指向数组的指针变量可带下标，p[i]与*(p+i)等价

❖一级指针变量与一维数组的关系

int *p 与 int q[10]

- 数组名是指针（地址）常量
- p=q; p+i 是q[i]的地址
- 数组元素的表示方法:下标法和指针法，

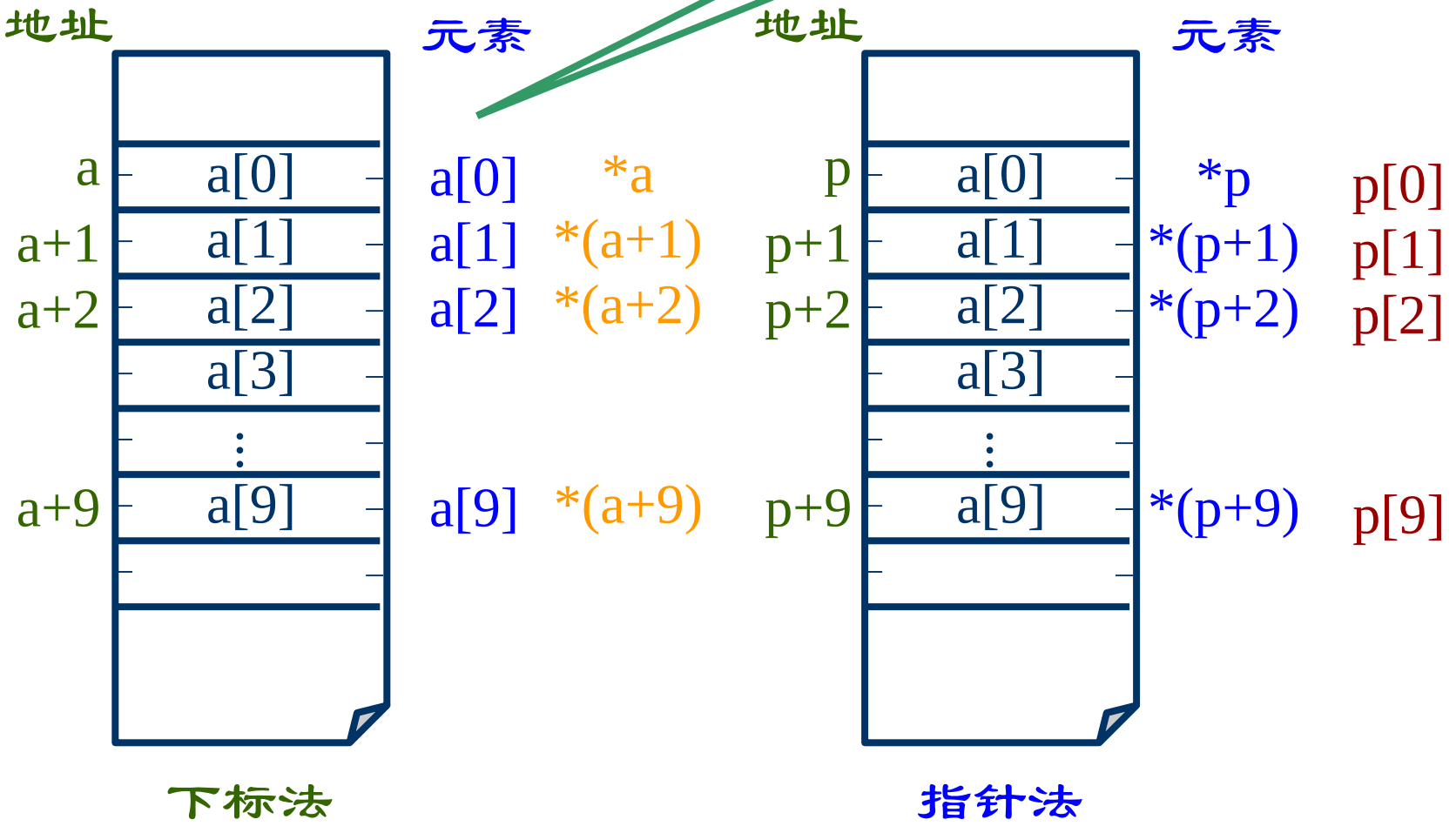
  即：若p=q,

  则：p[i] ⇔ q[i] ⇔ *(p+i) ⇔ *(q+i)
- 形参数组实质上是指针变量。

  即：int q[ ] ⇔ int *q
- 在定义指针变量（不是形参）时，不能把int *p 写成int p[];
- 系统只给p分配能保存一个指针值的内存区(一般2字节）；而给q分配2*10字节的内存区

表示数组元素的两种方法：

[] 变址运算符
a[i] ⇔ *(a+i)



a[i] ⇔ p[i] ⇔ *(p+i) ⇔*(a+i)

例 用三种方法输出数组中全部元素的值

(1) 下标法：

```
#include <stdio.h>
void main()
{int a[10];
 int i;
 for(i=0; i<10; i++)
   scanf("%d",&a[i]);
 printf("\n");
 for(i=0; i<10; i++)
   printf("%d",a[i] );}
```

(4) 指针法和指针下标：

```
#include <stdio.h>
void main()
{int a[10];
 int *p,i;
 for(i=0; i<10; i++)
   scanf("%d",&a[i]);
 printf("\n");
 for(p=a,i=0; i<10; i++)
   printf("%d",*(p+i));}
```

(3) 指针法：

```
#include <stdio.h>
void main()
{int a[10];
 int *p,i;
 for(i=0; i<10; i++)
   scanf("%d",&a[i]);
 printf("\n");
 for(p=a; p<(a+10); p++)
   printf("%d",*p );}
```

运行情况：

```
1 2 3 4 5 6 7 8 9 0↵
1 2 3 4 5 6 7 8 9 0
```

使用指针变量时要注意的问题：

(1) p++: 合法，因为p是指针变量，++只能用于变量。

a++: 不合法，因为a是数组名，其值是数组元素的首地址，是常量，程序运行期间值固定不变。

(2) 指针变量使用时要注意当前值。

算术运算

指针变量可以进行的算术运算包括：

1、指针变量±整数。

2、指针变量++/++指针变量。

3、指针变量--/--指针变量。

4、指针变量1-指针变量2。

由于指针运算符*与自增运算符++、自减运算符—的优先级相同，结合方向都是从右至左，因此需要注意以下各种形式的含义不同。

算术运算

```
*++px;      /*先使指针变量 px 加 1，再取新的指针变量 px 所指向地址的内容*/
*px++;      /*先取指针变量 px 所指向地址的内容，再使指针变量 px 加 1*/
*(++px);    /*等价于*++px; */
*(px++);    /*等价于*px++;*/
++*px;      /*等价于 ++(*px);，将指针变量 px 所指向地址的内容加 1*/
*--px;      /*等价于*(--px);，先使指针变量 px 减 1，再取新的 px 所指向地址的内容*/
*px--;      /*等价于*(px--);，先取指针变量 px 所指向地址的内容，再使指针变量 px 减 1*/
--*px;      /*等价于 --(*px);，将指针变量 px 所指向地址的内容减 1*/
px-py       /*是指两个相同类型的指针可以进行减法运算，运算结果是两个指针之间的数据个
              数，而不是两个指针的地址之差*/
```

关系运算

两个类型相同的指针变量可以运用关系运算符比较大小，表示两个指针变量所指向地址位置的前后关系，即前者为小，后者为大。

需要注意的是，如果两个指针变量不是指向同一个数组，则比较大小没有实际意义。

课堂练习

3、设int a[10],*p=a;,数组元素a[4]的正确引用是（　　）

A. *(p+4)

B. p+4

C. *p+4

D. a+4

课堂练习

3、设int a[10],*p=a;,数组元素a[4]的正确引用是（　　）

A. *(p+4)

B. p+4

C. *p+4

D. a+4

答案：A

解析：当指针变量指向一维数组，利用指针变量引用一维数组元素的方法：引用下标为i的数组元素即*（指针变量+i）或指针变量[i]，故在int a[10],*p=a;中数组元素a[4]的正确引用是*(p+4)。

课堂练习

4、若定义了int a[10]，*p；，将数组元素a[8]的地址赋给指针变量p的赋值语句是_____。

课堂练习

4、若定义了int a[10]，*p；，将数组元素a[8]的地址赋给指针变量p的赋值语句是_____。

答案：p=&a[8];

解析：在程序中用赋值的方式，即指针变量=&数组名[下标]，所以将数组元素a[8]的地址赋给指针变量p的赋值语句是p=&a[8]；。

指针和字符串

指向字符串的指针变量：

将指针变量指向字符串的方法如下：

1、在数据定义语句中用赋初值的方式　　*指针变量=字符串

2、在程序中用赋值的方式　　指针变量=字符串;

需要注意的是，这两种方法并不是将字符串赋予指针变量，而是将存放字符串的连续内存单元的首地址赋予指针变量。

指针和字符串

当指针变量指向字符串时，则可以利用指针变量处理字符串。处理方式主要有:

1、处理整个字符串

输出整个字符串 printf("%s",指针变量);

输入整个字符串 scanf("%s",指针变量);

程序代码如下：

```
#include <stdio.h>
int main(void)
{
    char *string="I love China. ";
    printf("%s\n",string);
    return 0;
}
```

程序运行结果如下：

```
I love China.
```

指针和字符串

2、处理字符串中的单个字符

使用指向字符串的指针变量处理字符串中单个字符的例子。

程序代码如下：

```
#include <stdio.h>
int main(void)
{
    char *string="I love China. ";
    for(; *string!='\0'; string++)    printf("%c", *string);     /*利用 for 循环和指向字符串的指针变量
                                                                 string 逐一输出字符串各字符*/
    printf("\n");
    return 0;
}
```

程序运行结果如下：

```
I love China.
```

指针与字符串

★字符串的表示形式

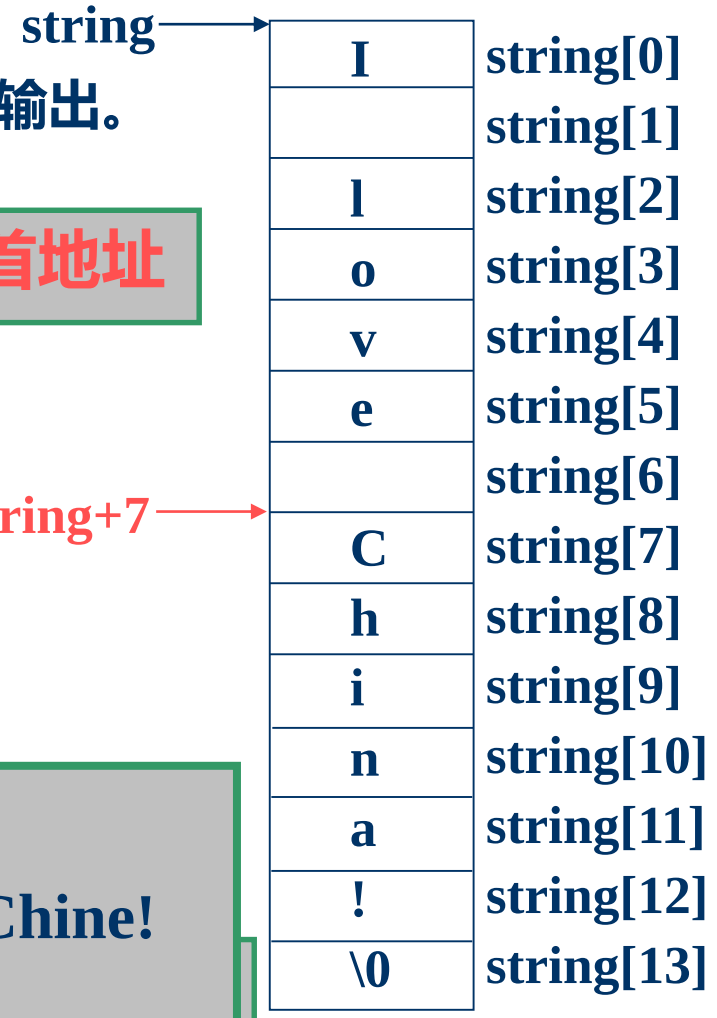
字符串: 用双引号括起的一串字符。
可赋给字符型的数组或指针变量，
可通过字符型数组名或字符型指针变量输出。

❖用字符数组实现

数组名，数组首地址

```
例 定义字符数组
#include <stdio.h>
void main( )
{ char string[]="I love China!";
   printf("%s\n",string);
   printf("%s\n",string+7);
}
```

输出：
I love Chine!
Chine!



string[4] ⇔ *(string+4)

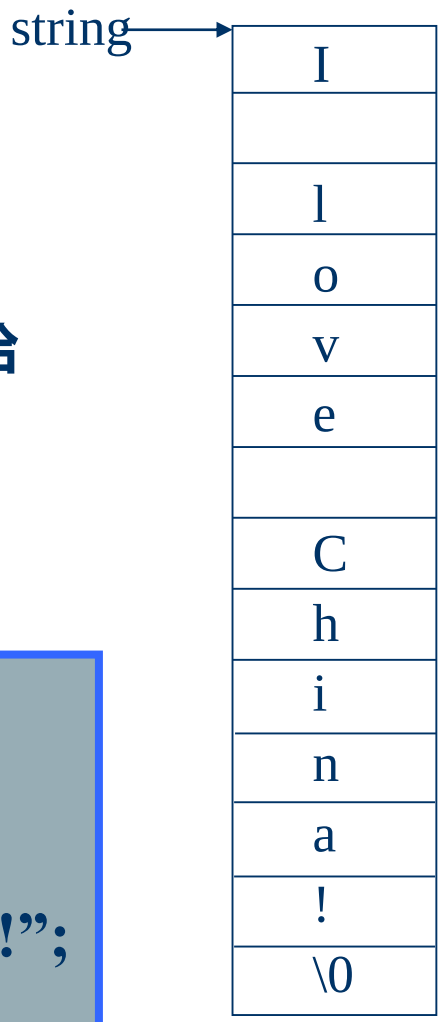
❖用字符指针实现

- 字符串的指针就是字符串的首地址,即第一个字符的地址,可以使用字符指针变量来保存这个地址。
- 使用字符指针可以处理字符串
- 字符指针的定义及使用
  - ◆定义和初始化。

  例： char *string="I love China!";
  - ◆在程序中可以直接把字符串常量赋给一个指针变量。

  例：char *string;

  string="I love China!";



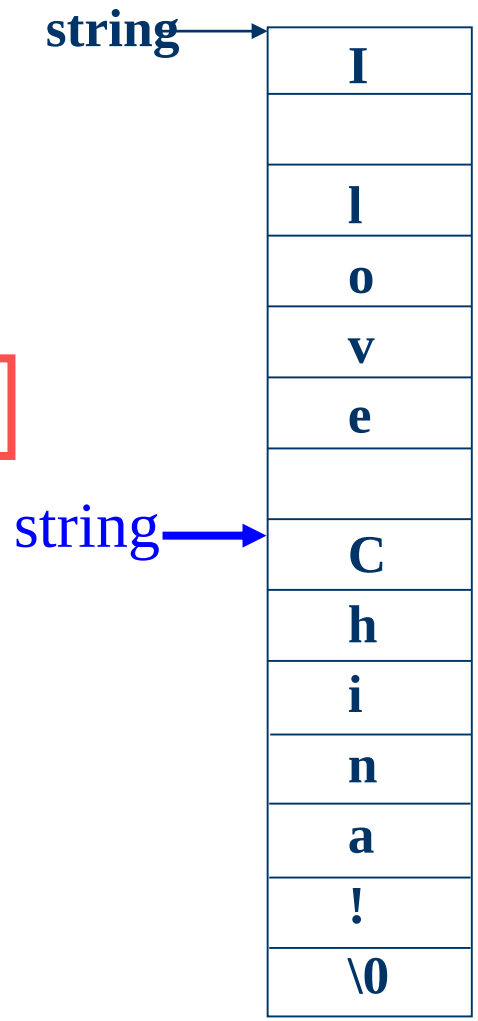
例 定义字符指针

```
#include <stdio.h>
void main( )
{ char  *string="I  love China!";
  printf("%s\n",string);}
```

改动后的

```
#include <stdio.h>
void main( )
 { char  *string="I  love China!";
   printf("%s\n",string);
   string+=7;
          while(*string)
           {     putchar(string[0]);
                  string++;
            }
 }
```

*string!='\0'



输出:

I love Chine!

Chine!

❖用下标法存取字符串中的字符

地址访问:
a[ ]复制到b[ ]

下标法输出

例 将字符串a复制为字符串b

```
#include <stdio.h>
void main( )
{ char   a[ ]="I am boy.",b[20];
  int i;
  for(i=0;*(a+i)!='\0';i++)
      *(b+i)=*(a+i);
  *(b+i)='\0';
  printf("string a is: %s\n",a);
  printf("string b is: ");
  for(i=0;b[i]!='\0';i++)
      printf("%c",b[i]);
  printf("\n");
}
```

*(a+i) = a[i]

⇔ b[i]=a[i]

运行结果:

```
string a is: I am boy.
string b is: I am boy.
```

指向字符数组的指针变量

C语言中，字符串是按字符数组进行处理的，系统存储一个字符串时先分配一个起始地址，从该地址开始连续存放字符串中的字符。这一起始地址即字符串首字符的地址。所以，可以将一个字符串赋值给一个字符数组，也可以赋值给一个字符指针变量。

字符指针和字符数组的区别

字符指针和字符数组的区别主要体现在：

1、存储内容不同

2、赋值方式不同

3、字符指针变量在定义后应先赋值才能引用

4、指针变量的值是可以改变的，字符指针变量也不例外；而数组名代表数组的首地址，是一个常量，而常量是不能改变的。

★对使用字符指针变量和字符数组的讨论

char *cp; 与 char str[20]; 的区别

- ❖str由若干元素组成，每个元素放一个字符；而cp中存放字符串首地址
- ❖赋值方式：
  - 字符数组只能对元素赋值。 char str[20];
    str="I love China!"; (×)
  - 字符指针变量可以用： char *cp;
    cp="I love China!"; (✓)
  - 赋初值：char *cp="China!"; 等价 char *cp; cp="China!";

char str[14]={"China"};不等价char str[14]; str[ ]="China" (×)

课堂练习

5、设char*st=〞China，Beijing〞；，执行语句printf(〞%s\n〞，st+6)；后的输出结果是_____。

答案：Beijing

课堂练习

5、设char*st=〞China，Beijing〞；，执行语句printf(〞%s\n〞，st+6)；后的输出结果是_____。

答案：Beijing

解析：char*st=〞China，Beijing〞；是定义字符型指针st，在输出语句中st+6是指针的首地址指向第六位进行输出，即在执行语句printf(〞%s\n〞，st+6)；后的输出结果是Beijing。

指针和函数

指针变量作为函数参数：

指针变量既可以作为函数的形参，也可以作为函数的实参。指针变量作为函数参数，形参和实参之间的数据传递方式本质上是值传递，只是在调用函数时传递的内容是地址，这样使得形参变量和实参变量指向同一个变量。若被调函数中有对形参所指变量内存的改变，实际上是改变了实参所指变量的内容。

指针和函数

数组名作为函数参数：

数组名作为函数形参时，接收实参数组的首地址；数组名作为函数实参时，将数组的首地址传递给形参数组。引入指向数组的指针变量后，数组及指向数组的指针变量作为函数参数时，可有四种等价形式：

1、形参、实参均为数组名。

2、形参、实参均为指针变量。

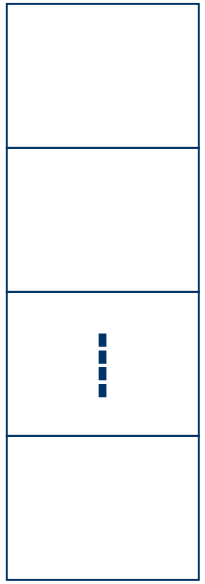
3、形参为指针变量、实参为数组名。

4、形参为数组名、实参为指针变量。

★用数组名作函数参数

数组名作函数参数，是地址传递

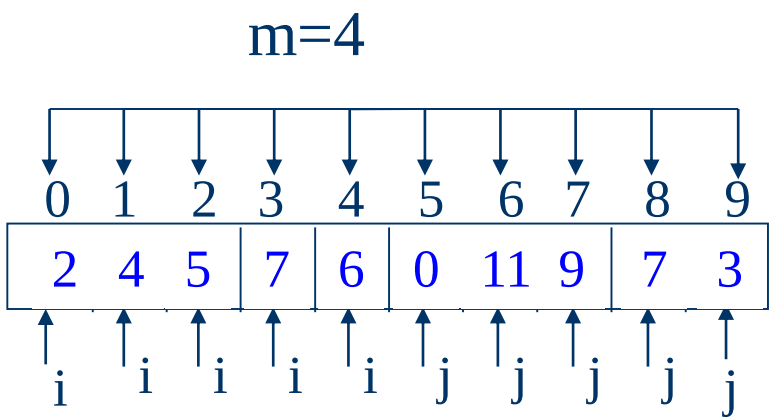
当用数组名做函数实参时相当于将数组的首地址传给被调函数的形参，此时，形参数组和实参数组占用的是同一段内存，所以当在被调函数中对形参数组元素进行修改时，实参数组中的数据也将被修改，因为它们是同一个地址。




```
数组名作函数参数
void main( )
{f(int arr[ ], int n);
  int  array[10];
        ┆
  f(array, 10);
        ┆
}
void f(int arr[ ], int n)
{
  ┆
}
```

编译时arr按指针变量处理，所以，此句与f(int *arr , int n)等价。

将数组a中n个整数按相反顺序存放。
思路：数组元素头尾对调。四种调用方式。

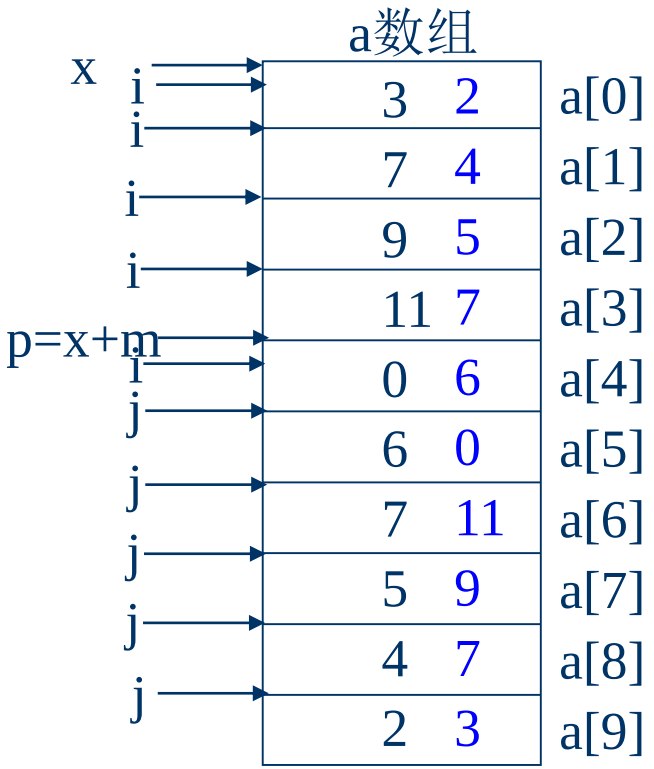


(1) 实参与形参均用数组

```
#include <stdio.h>
void main()
{void inv(int  x[ ], int n);
  int i,a[10]={3,7,9,11,0,6,7,5,4,2};
  printf("The original array:\n");
  for(i=0;i<10;i++)   printf("%d,"a[i]);
  printf("\n");
  inv(a,10);
  printf("The array has been inverted:\n");
  for(i=0;i<10;i++)   printf("%d,",a[i]);
  printf("\n");
}
void inv(int  x[ ], int n)
{ int temp,i,j,m=(n-1)/2;
  for(i=0;i<=m;i++)
  { j=n-1-i;
     temp=x[i];  x[i]=x[j];  x[j]=temp; }
  return;
}
```

(2) 实参用数组，形参用指针变量

```
#include <stdio.h>
void main()
{void inv(int  *x, int n);
  int i,a[10]={3,7,9,11,0,6,7,5,4,2};
  printf("The original array:\n");
  for(i=0;i<10;i++)   printf("%d,"a[i]);
  printf("\n");
  inv(a,10);
  printf("The array has been inverted:\n");
  for(i=0;i<10;i++)   printf("%d,",a[i]);
  printf("\n");
}
void inv(int  *x, int n)
{ int temp,*p,*i,*j,m=(n-1)/2;
  i=x;  j=x+n-1;  p=x+m;
  for(;i<=p;i++,j--)
    { temp=*i;  *i=*j;  *j=temp; }
  return;
}
```



（3）实参与形参均用指针变量

```
#include <stdio.h>
void main()
{void inv(int *x, int n);
  int i,arr[10],*p=arr;
  printf("The original array:\n");
  for(i=0;i<10;i++,p++)
     scanf("%d",p);
  p=arr;  inv(p,10);
  printf("The array has been inverted:\n");
  for(p=arr;p<arr+10;p++)
     printf("%d",*p);
  printf(“\n");
}
void inv(int *x, int n)
{ int *p, m, temp,*i,*j;
   m=(n-1)/2;
  i=x;  j=x+n-1;  p=x+m;
  for(;i<=p;i++,j--)
  { temp=*i; *i=*j; *j=temp; }
  return;}
```

此句用意?

⑷ 实参用指针变量，形参用数组

```
#include <stdio.h>
void main()
{void inv(int  x[ ], int n);
  int i,a[10],*p=a;
  for(i=0;i<10;i++,p++)
     scanf("%d",p);
  p=a;
  inv(p,10);
  printf("The array has been inverted:\n");
  for(p=arr;p<arr+10;p++)
     printf("%d ",*p);
  printf("\n ");
}
void inv(int  x[ ], int n)
{ int t,i,j,m=(n-1)/2;
  for(i=0;i<=m;i++)
    { j=n-1-i;
       t=x[i];  x[i]=x[j];  x[j]=t; }
  return;
}
```

归纳：用数组做函数参数有如下四种情况：

1、实参形参都用数组名：

```
int a[10];                    inv(int x[ ],int n)
inv(a,10)                     { ……  }
```

2、实参用数组名，形参用指针变量：

```
int a[10];                    inv(int *x,int n)
inv(a,10)                     { ……  }
```

3、实参形参都用指针变量：

```
int a[10];                    inv(int *x,int n)
int *p=a;                     {……}
inv(p,10)
```

4、实参用指针变量，形参用数组名：

```
int a[10];                    inv(int x[ ],int n)
int *p=a;                     {……}
inv(p,10)
```

指针型函数的定义

- C语言中，函数可以返回整型、实型、字符型数据，也可以返回指针类型数据，即返回一个地址。指针型函数是指函数的返回值是指针型，即这类函数的返回值必须是地址值，调用该类函数时，接收返回值的必须是指针变量、指针数组元素等能够存放地址值的对象。

- 定义指针型函数的格式和有返回值的函数的定义格式基本相同，唯一的区别是在函数名前面加一个“*”，表示函数的返回值是指针型数据。

指针型函数的调用

指针型函数的调用和一般函数的调用方法完全相同，但需要注意的是只能使用指针变量或指针数组元素接收指针型函数的返回值，不能使用数组名接收指针型函数的返回值，因为函数名是地址常量，不是地址型变量，不能接收地址型变量数据。

课堂练习

6、若定义的函数为float*fun( ){……}，则函数的返回值是（　　）

A. float 型数据

B. void 型数据

C. float型指针

D. void型指针

课堂练习

6、若定义的函数为float*fun( ){……}，则函数的返回值是（　　）

A. float 型数据

B. void 型数据

C. float型指针

D. void型指针

答案：C

解析：指针型函数是指函数的返回值是指针型，即函数float*fun( ){……}，的返回值为float型指针。

课堂练习

7. 关于函数正确的说法是（　　）

A. 必须有返回值

B. 必须有形式参数

C. 返回值的类型可以是指针

D. 一个函数中可以定义另一个函数

课堂练习

7. 关于函数正确的说法是（　　）

A. 必须有返回值

B. 必须有形式参数

C. 返回值的类型可以是指针

D. 一个函数中可以定义另一个函数

答案：C

解析：C语言中，函数可以返回整型、实型、字符型数据，也可以返回指针类型数据，即返回一个地址。指针型函数是指函数的返回值是指针型。

第五节 指针数组

指针数组的定义：

指针数组是数组中的元素均为指针变量。

【格式】数据类型符 *指针数组名[数组长度];

【功能】定义指针数组，有“长度”个数组元素。

【说明】

1、指针数组名是标识符，前面必须加“*”号。

2、定义指针数组的同时可以定义普通变量、数组和指针变量等。

3、“数据类型符”可以是任何基本数据类型。“数据类型符”不是指针数组元素中存放的数据类型，而是其所指向数据的数据类型。

指针数组的初始化

例如：char *ps[]={"China","America","Russia",NULL};

定义了一个用于指向字符型数据的指针数组ps，其长度为4，同时对指针数组元素赋初值，前面三个是字符型指针，最后一个是空指针。

例如，int a[3][3]={1,2,3,4,5,6,7,8,9};

int *p[3]={a[0],a[1],a[2]};/*利用二维数组元素初始化指针数组p*/

指针数组元素的赋值

1、将数组名赋予指针数组各元素

例如：char s[4][20]={“China”,” America”,” Russia”,NULL};

```
char *p[4];
p[0]=s[0];/*给指针数组元素p[0]赋值s[0]，s[0]是字符串“China” 的首地址*/
```

2、将字符串直接赋予指针数组元素

例如：char *p[4];

```
p[0]= “China”;/*直接给指针数组元素p[0]赋值为字符串“China”的首地址*/
```

指针元素的使用

指针数组元素的表示方法和普通数组元素的表示方法完全相同。

指针数元素的使用和指针变量的使用完全相同，可以对其赋予地址值，可以利用其引用所指向的变量或数组元素，也可以参与运算。

指针元素的使用

表 7-2　常用指针数组元素的使用方法

使用方法		使用格式
对其赋值		指针数组名[下标]=地址表达式
引用所指向的变量或数组元素		*指针数组名[下标]
参与运算	算术运算	指针数组名[下标]+整数 指针数组名[下标]-整数 ++指针数组名[下标] --指针数组名[下标] 指针数组名[下标]++ 指针数组名[下标]--
	关系运算	指针数组名[下标 1]　关系运算符　指针数组名[下标 2]

其中算术运算和关系运算一般只用于指针数组元素指向同一个指针数组。

pointer[i]+j、array[i]+j、 &array[i][j] 等价，均指向array[i][j]

*(pointer[i]+j)、 *(*(pointer+i)+j)、 pointer[i][j] 与array[i][j] 等价。

```
#include <stdio.h>
#include <string.h>
int main(void)
{
    int array[3][3]={1,2,3,4,5,6,7,8,9),i,j;          /*定义一个整形二维数组 array*/
    int *pointer[3]={array[0],array[1],array[2]};          /*定义一个指向整形数据的指针数组
          pointer并初始化，即将二维数组array的每行元素的首地址赋予指针数组的各元素*/
    for(i=0;i<3;i++)
          for(j=0;j<3;j++)
               printf("%4d",pointer[i][j]);/*利用指向二维数组元素的指针数组输出二维数组各元素的值*/
     return 0;
}
```

输入三个国家的名称，按字母顺序排序后输出。

```
#include <stdio.h>
#include <string.h>
int main(void)
{
    char *s[]={"China","America","Russia"},*p;  /*定义指针数组s和指针变量p*/
    int i,j,k=3;
    for(i=0;i<k-1;i++)
        for(j=0;j<k-1;j++)
            if(stremp(s[j],s[j+1]>0)
            {   p=s[j];  s[j]=s[j+1];  s[j+1]=p;   }
    for(i=0;i<k;i++)
          printf("%s\n",s[i]);
     return 0;
}
```

祝大家顺利通过考试！