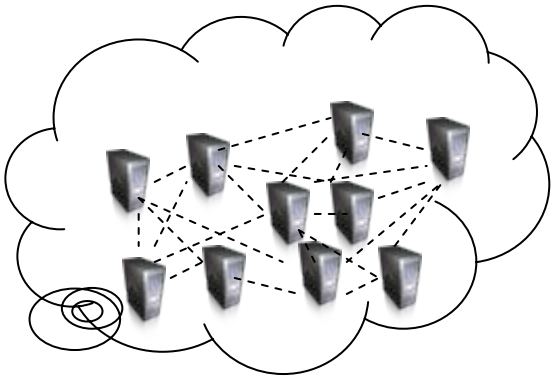
厦门大学非计算机专业本科生公共课

（2011-2012第2学期）



C语言程序设计

林子雨

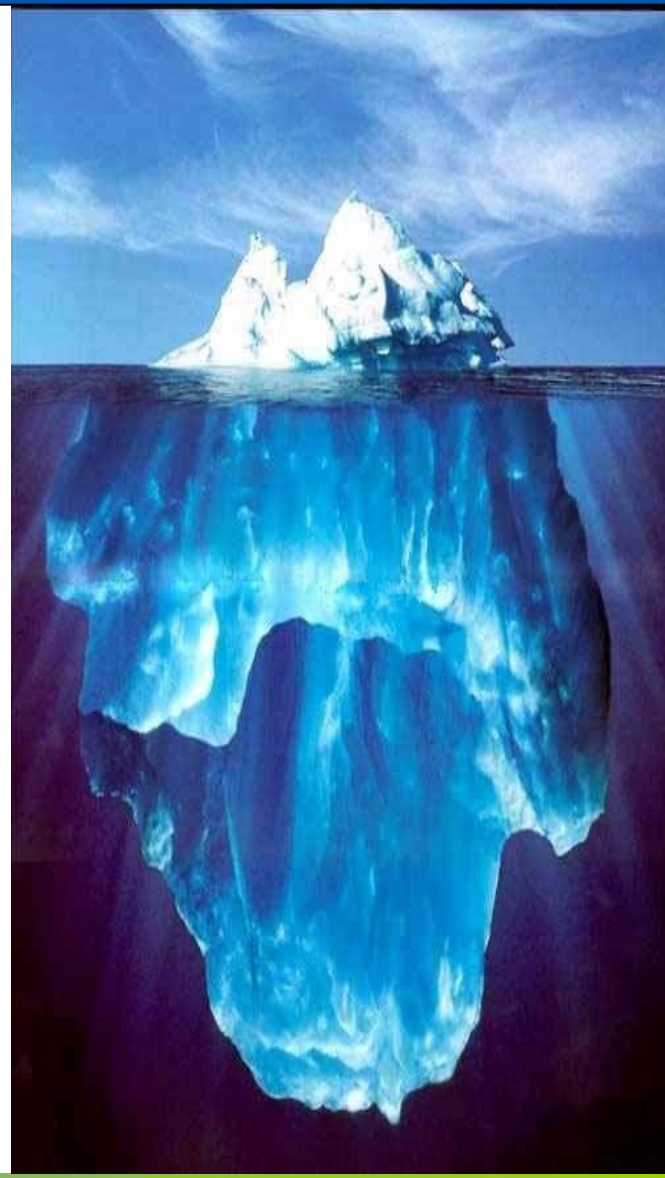
厦门大学计算机科学系

E-mail: ziyulin@xmu.edu.cn

个人主页：**http://www.cs.xmu.edu.cn/linziyu**

课程提要

- 第一章 绪论
- 第二章 C语言基础
- 第三章 结构化程序设计
- 第四章 选择结构
- 第五章 循环结构程序设计
- 第六章 函数
- 第七章 编译预处理
- 第八章 数组
- 第九章 结构体、共用体和枚举类型
- 第十章 指针



第10章 指针

10.0 指针概述

- 指针是C语言中的一个重要概念，也是C语言的一个重要特色。
- 指针可以有效地表达复杂的数据结构；
- 能动态地分配内存，更有效地利用内存空间；
- 能方便地表示数组和字符串，提高数据处理效率；
- 指针作为函数参数，能使函数调用得到多于一个值；
- 在程序中适当使用指针，会使程序显得灵活高效。

10.1 地址与指针变量

- 10.1.1 内存单元地址
- 10.1.2 指针
- 10.1.3 指针变量的定义和初始化
- 10.1.4 指针的运算
- 10.1.5 指向指针的指针(*)

10.1.1 内存单元地址

- 计算机主存储器由一个个存储单元组成，微型计算机以字节作为存储单元。
- 每个存储单元具有唯一的地址，存储单元的地址是一个无符号整数，主存储器的所有存储单元的地址是连续的。
- 程序中处理的数据需要在内存中占用存储单元，编译系统根据变量的数据类型，为变量分配若干个存储单元（字节）。例如VC++系统为每个字符变量分配一个字节，为每个整型变量分配4个字节，为每个单精度实数分配4个字节。
- 一个变量所占用存储区域的所有字节都有各自的地址，C系统把存储区域中第一个字节的地址作为变量的地址。
- 要访问变量中的数据，就要知道该变量的内存地址。

10.1.1 内存单元地址

- 前面章节使用了大量的变量，包括整型、字符型、数组和构造型变量，但并没有直接使用变量的地址。
- **直接访问**：C语言屏蔽了底层的实现细节，在程序中定义的变量，编译时系统就给这个变量分配适当的内存单元，并把变量名和变量存储地址对应起来，我们就可以直接在程序中 通过变量名访问存储单元中的数据。通过变量名访问变量中的数据，这种变量存取方式称为“直接访问”。
- **间接访问**：变量的地址也是数据，如果定义一种特殊的变量（即指针变量），用指针变量存放某变量的地址，就可以由前一个变量的值得到后一个变量的地址，然后通过该地址取得后一个变量的值，这就是所谓的“间接访问”。

10.1.2 指针

- 存储整数的变量称为整型变量，存储字符的变量称为字符变量，存储某个变量地址的变量应该称为“地址变量”。在C语言中，把地址变量称为“**指针变量**”。
- 指针变量存储的是另一个变量的地址，即指针变量“指向”另一个变量对应的存储区域。这种指向是通过地址来体现的，因此地址也被称为指针，那么存放某个变量地址的变量就被称为“指针变量”。
- 变量的地址是在程序编译时由系统分配的，变量地址一经分配就不能改变，因此变量地址是一个常量。
- 变量地址可由运算符“&”和变量名运算获得。例如输入函数scanf(“%d”,&a),其中，表达式“&a”的值就是变量a的地址，执行该函数从键盘读取一个整数，存入变量a的存储区域中。
- 数组名本身就是地址，它代表该数组首元素的地址。设a为一维数组，则有a==&a[0]，即a代表a[0]的地址。

char a[6]=“Hello”;printf(“%s”,a)和printf(“%s”,&a[0])结果是否相同?

10.1.2 指针

- 例10.1.1 变量地址输出示例，

```
#include <stdio.h>
void main()
{
   int a,b[4];
   printf("a的地址（十进制）：      %d\n",&a);
   printf("b的地址（十进制）：      %d\n",b);//b的地址就是b[0]的地址
   printf("b[1]的地址（十进制）：   %d\n",&b[1]);
   printf("a的地址（十六进制）：    %x\n",&a);
   printf("b的地址（十六进制）：    %x\n",b);
   printf("b[1]的地址（十六进制）：  %x\n",&b[1]);
}
```

```
a的地址（十进制）：     1245052
b的地址（十进制）：     1245036
b[1]的地址（十进制）：  1245040
a的地址（十六进制）：    12ff7c
b的地址（十六进制）：    12ff6c
b[1]的地址（十六进制）：  12ff70
```

10.1.3 指针变量的定义和初始化

- 指针变量属于简单变量，一个指针变量存储一个变量的地址。
- 定义指针变量的一般形式为：

  基类型 * 指针变量名;

  其中，“基类型”为某已定义的数据类型，可以是系统预定义的类型，也可以是用户自己定义的类型，可以是简单类型，也可以是构造类型。符号“*”表示定义的是一个指向“基类型”的指针变量。例如：

```
int var=10;
int * pointer = &var;
```

注意：指针变量的变量名是pointer，而不是*pointer。指针变量pointer的基类型为int，表示变量pointer只能存储一个整型变量的地址，不能存储其他类型变量的地址。例如下面的初始化是错误的：

```
float f;
int *p = &f;
```

即不能把一个实型变量的地址赋给一个基类型为整型的指针。

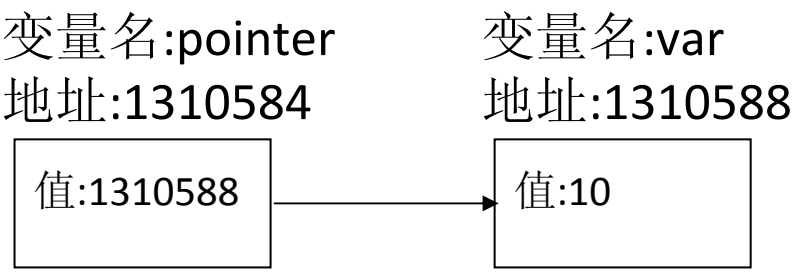


图10.1.1 指针变量示意图

10.1.3 指针变量的定义和初始化

- 可以在一个语句中同时定义整型变量和指针变量，例如：
  int a,b,*p1,*p2;
- 当定义一个指针变量而没有初始化时，指针变量的值没有确定，即该指针不指向特定的变量。这时使用这个变量是危险的，可能造成不可预料的后果。
- 可以把指针变量初始化为0，即不指向任何变量。例如：
  int *pinter=0;
  或
  int *pointer=NULL;

10.1.4 指针的运算

- 10.1.4.1 指针专用运算符
- 10.1.4.2 赋值运算
- 10.1.4.3 算术运算
- 10.1.4.4 关系运算(*)

10.1.4.1 指针专用运算符

- **（1）取地址运算符&**
- 运算符&是个单目运算符，结合方向是右结合。&的运算对象只能是变量名（包括简单变量和构造型变量）、数组元素或结构体成员，不能是表达式。&运算结果就是运算对象（变量）的地址。

  例如：

```
int *p1,*p2,a,b[10];
p1=&a; //把整型变量a的地址赋给指针变量p1
p2=&b[0];//把数组元素b[0]的地址赋给指针变量p2
```

10.1.4.1 指针专用运算符

- **（2）指向运算符***
- 运算符*在算术运算中表示乘法运算，是双目运算符。在这里*称为指向运算符或间接访问运算符，是单目运算符，结合方向是右结合。
- *的运算对象只能是指针变量，*的运算结果得到运算对象（指针变量）所指的变量。例如当p1=&a时，*p1就表示变量a，是对变量a的间接引用。*p1是整型变量，不是指针，*p1可以像一般整型变量一样使用。
- 注意：符号*出现在不同的位置，含义也不同。在变量定义语句中（如int * p;），int * 是类型名，p是变量名；在表达式中，符号*即可表示乘法，也可表示指向。

10.1.4.1 指针专用运算符

- 例**10.1.2**输入两个整数，按先大后小的顺序输出。程序清单如下：

```
#include <stdio.h>
void main()
 {
   int * p1, * p2, * p, a, b;
   printf("输入任意两个整数：");
   scanf("%d%d",&a,&b);
   p1=&a; p2=&b;          //如图10.1.2（a）所示
   if(a<b)
   {
      p=p1; p1=p2; p2=p;//如图10.1.2（b）所示
   }
   printf("a=%d\tb=%d\n",a,b);
   printf("max=%d\tmin=%d\n",*p1, *p2);
}
```

程序运行结果：
输入任意两个整数：3 5
a=3 b=5
max=5 min=3

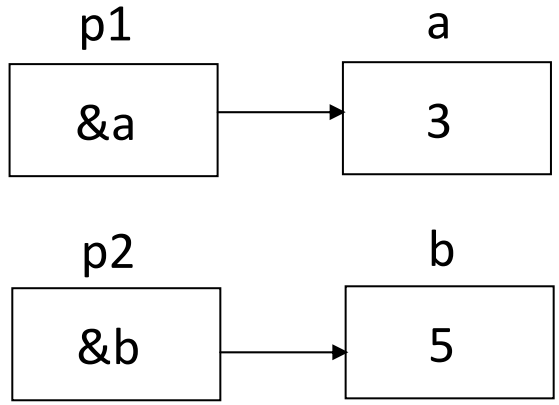


图10.1.2(a)

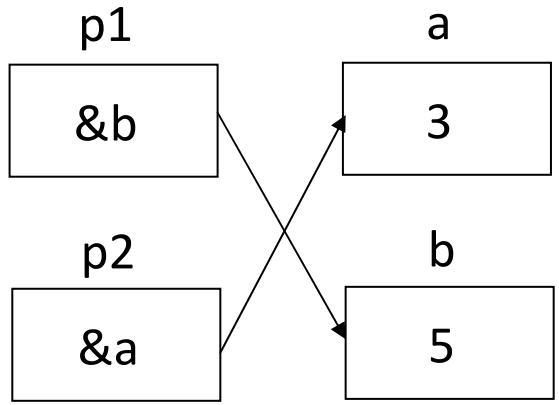


图10.1.2(b)

10.1.4.2 赋值运算

- 可以把指针常量或另一个同类型的指针变量赋给指针变量。例如：

```
int a,*p1,*p2=0;  //对p2初始化
p1=p2;  //把p2的值赋给p1
p2=&a;  //把变量a的地址赋给p2
```

- 注意：基类型相同的指针才能进行赋值运算。
- 指针变量的值虽然是一个整数，但不能把整数（0除外）直接赋给指针变量，也不能把指针赋给整型变量。例如下面的运算时错误的：

```
p1=100;//错误
a=p1;// 错误
```

10.1.4.3 算术运算

- 指针和整数进行加减运算
- 只有当指针指向数组时，这种运算才有意义。例如当指针p指向数组时，下面运算时合法的：

p++; //p指向下一个数组元素

p=p+2; //p指向当前元素后的第二个元素

```
#include <stdio.h>
void main()
{
        int *p1,b[4]={1,2,3,4};
        p1=b; //把数组变量b的首地址赋给指针变量p1
        printf("value=%d\n",*p1);
        p1++;
        printf("value=%d\n",*p1);
        p1=p1+2;
        printf("value=%d\n",*p1);
}
```

运行结果：
value=1
value=2
value=4

10.1.4.3 算术运算

- 例**10.1.3**指针算术运算示例，程序清单如下：

```
#include <stdio.h>
void main()
 {
    int i,a[]={1,3,5,7,9},*p=a;
    for(i=0;i<5;i++)
       printf("a[%d]=%d\t",i,a[i]);
    printf("\n");
    printf("a=%d\n",p);                  //输出
    printf("p+2=%d\n",p+2);              //输出
    printf("*p+3=%d\n",*p+3);            //输出
    printf("*(p+3)=%d\n",*(p+3));        //输出
    printf("*p++=%d\n",*p++);            //输出
    p=a;
    printf("*++p=%d\n",*++p);            //使p指
    printf("++*p=%d\n",++*p);            //输出
}
```

程序运行结果：

```
a[0]=1 a[1]=3 a[2]=5 a[3]=7 a[4]=9
a=1310568
p+2=1310576
*p+3=4
*(p+3)=7
*p++=1
*++p=3
++*p=4
```

10.2 指针与函数

- 10.2.1 指针变量作为函数参数
- 10.2.2 函数的返回值为指针(*)
- 10.2.3 指向函数的指针(*)

10.2.1 指针变量作为函数参数

- 在C语言中，函数的形式参数不仅可以是整型变量、实型变量或字符型变量，也可以是指针变量。使用指针作为函数参数，主调函数将变量的地址传递给被调函数。
- 用指针做参数，实际参数和形式参数之间仍然是值传递，即在调用函数时，把实际参数的值赋给形式参数。被调函数中虽然不能改变作为实际参数的指针变量的值，但可以改变指针变量所指变量的值，从而达到在被调函数中修改主调函数中变量的目的。
- 用指针变量作为函数参数，可以获得从函数中带回多于一个值的客观效果。

10.2.1 指针变量作为函数参数

- **例10.2.1**调用函数，交换两个变量的值。
- //程序1，使用整型参数：

```
#include <stdio.h>
void swap(int x , int y)
{
  int temp;
  temp=x;        x=y;  y=temp;          //交换形参x，y的值
}
void main()
{
  int a,b;
  printf(“请输入两个整数a和b:”);
  scanf("%d%d",&a,&b);
  printf("调用函数前：a=%d\tb=%d\n",a,b);
  swap(a,b);
  printf("调用函数后：a=%d\tb=%d\n",a,b);
}
```

程序运行结果：
请输入两个整数a和b:3 5
调用函数前： a=3 b=5
调用函数后： a=3 b=5

10.2.1 指针变量作为函数参数

- **例10.2.1**调用函数，交换两个变量的值。
- //程序2：使用指针参数

```
#include <stdio.h>
void swap(int * p1 , int * p2)
{
  int temp;
  temp=*p1;      *p1=*p2; *p2=temp; //交换指针变量所指变量的值
}
void main()
{
  int a,b;
  printf(”请输入两个整数a和b:”);
  scanf("%d%d",&a,&b);
  printf("调用函数前：a=%d\tb=%d\n",a,b);
  swap(&a,&b);//实参为变量的地址
  printf("调用函数后：a=%d\tb=%d\n",a,b);
}
```

程序运行结果：
请输入两个整数a和b:3 5
调用函数前：a=3 b=5
调用函数后：a=5 b=3

10.2.1 指针变量作为函数参数

- 例**10.2.2**已知长方体的长、宽、高，求其体积和三个侧面的面积。程序清单如下：

```
#include<stdio.h>
int volume(int l,int w,int h,int *ps1,int *ps2,int *ps3)
{
    *ps1=l*w;
    *ps2=l*h;
    *ps3=w*h;
    return l*w*h;
}
void main()
{
    int a,b,c,v,s1,s2,s3;
    printf("请输入长方体的三条边：");
    scanf("%d%d%d",&a,&b,&c);
    v=volume(a,b,c,&s1,&s2,&s3);
    printf("volume=%d\n",v);
    printf("s1=%d\ts2=%d\ts3=%d\n",s1,s2,s3);
}
```

```
请输入长方体的三条边：1 2 3
volume=6
s1=2    s2=3    s3=6
```

10.3 指针与数组

- 10.3.1 一维数组与指针
  - 10.3.1.1 指向数组元素的指针
  - 10.3.1.2 数组名作为函数参数
- 10.3.2 二维数组与指针
- 10.3.3 字符串与指针
- 10.3.4 指针数组

10.3.1.1 指向数组元素的指针

- 一个数组的所有元素在内存中是连续存储的，数组元素也是变量，各数组元素都有各自的地址。数组第一个元素（下标为0的元素）的地址称为数组的首地址。在C语言中，数组名就代表该数组的首地址。例如有如下定义：

  int a[10],*p;

  则表达式p=a和表达式p=&a[0]等价，都表示指针变量p指向数组a的首地址。
- 数组首地址是系统编译时确定的，是一个常量。因此不能对数组首地址赋值，例如a=p、a++等都是错误的。
- 由10.1节介绍的指针运算规则可知：a代表数组元素a[0]的地址；a+1就是数组元素a[1]的地址;…;a+i就是数组元素a[i]的地址。因此，*(a+i)就表示a+i所指的变量，即数组元素a[i]。
- 用a[i]方式访问数组元素，称为下标引用法；而用*(a+1)方式访问数组元素，称为指针引用法。

10.3.1.1 指向数组元素的指针

- **例10.3.1**指针法引用数组元素示例，程序清单如下：

```
#include <stdio.h>
void main()
{
    int a[]={1,2,3,5,7,9},i,*p;
    printf("用下标引用数组元素：\n");
    for(i=0;i<6;i++)
      printf("%d\t",a[i]);
    printf("\n");
    printf("用数组名引用数组元素：\n");
    for(i=0;i<6;i++)
      printf("%d\t",*(a+i));
    printf("\n");
    printf("用指针引用数组元素：\n");
    for(p=a;p<a+6;p++)
      printf("%d\t",*p);
    printf("\n");
}
```

```
用下标引用数组元素：
1    2    3    5    7    9
用数组名引用数组元素：
1    2    3    5    7    9
用指针引用数组元素：
1    2    3    5    7    9
```

程序中3组输出结果相同，可见a[i]和*(a+1)完全等价。

10.3.1.1 指向数组元素的指针

- 例**10.3.2**数组输入示例，程序清单如下：

```
#include <stdio.h>
void main()
{
int a[5],i,*p;
printf("用下标引用数组元素： \n");
for(i=0;i<5;i++)
scanf("%d",&a[i]);
for(i=0;i<5;i++)
printf("%d\t",a[i]);
printf("\n");
//在右栏接续代码
```

```
//接续左栏的代码
printf("用数组名引用数组元素： \n");
for(i=0;i<5;i++)
scanf("%d",a+i);
for(i=0;i<5;i++)
printf("%d\t",a[i]);
printf("\n");
printf("用指针引用数组元素： \n");
for(p=a;p<a+5;p++)
scanf("%d",p);
for(i=0;i<5;i++)
printf("%d\t",a[i]);
printf("\n");
}
```

10.3.1.2 数组名作为函数参数

•数组名可以作为函数参数。因为数组名代表数组首地址，所以用数组名作为参数实际上就是用指针作为函数参数。

例10.3.3求数组元素的平均值，程序清单如下：

```
#include <stdio.h>
void main()
{
    int average(int a[]);              //函数原型声明
    int score[5]={60,80,90,70,50},i,aver;
    printf("Scores:\n");
    for(i=0;i<5;i++)
            printf("%d\t",score[i]);
    printf("\n");
    aver=average(score);               //调用求平均值函数
    printf("Average=%d\n",aver);
}
int average(int a[])                   //定义求平均值函数
{   int i,sum=0;
    for(i=0;i<5;i++)
            sum+=a[i];
    return sun/5;
}
```

```
Scores:
60    80    90    70    50
Average=70
```

10.3.1.2 数组名作为函数参数

说明:

- 数组作为函数的形式参数，数组长度可以省略，但方括号不能省略。即：
  int average(int a[5])//正确
  可以写成
  int average(int a[])//正确
  但是不能写成
  int average(int a)//错误
- 数组作为函数的实际参数，只能写数组名。即把
  aver=average(score)//正确
  写成以下形式都是错误的
  aver=average(score[]) //错误
  aver=average(score[5]) //错误
- 数组作为函数参数，函数调用时，系统把实参数组的首地址传递给形参，而不是把实参数组各元素的值传递给形参。因此，可以把数组形参写成指针形参。即可以把int average(int a[])写成int average(int * a)，函数体不变。同理，实参的数组名也可以由已赋值的指针变量代替。

10.3.1.2 数组名作为函数参数

- 例**10.3.4**逆序输出数组元素，程序清单如下：

```
#include <stdio.h>
void main()
{
    void list(int *a，int n);          //函数原型声明
    int score[5]={1,2,3,4,5},i,*p=score;
    printf(“scores:\n");
    for(i=0;i<5;i++)        //输出原数组
      printf("%d\t",*(p+i));
    printf("\n");
    list(p,5);                          //调用倒序函数
    for(i=0;i<5;i++)        //输出倒序后数组
      printf("%d\t",*(p+i));
    printf("\n");
}
void list(int *a, int n)                //倒序函数定义
{   int i,t;
    for(i=0;i<n/2;i++)
      t=a[i],a[i]=a[n-1-i],a[n-1-i]=t;
}
```

10.3.2 二维数组与指针

- 10.3.2.1 二维数组的地址
- 10.3.2.2 指向一维数组的指针
- 10.3.2.3 二维数组作为函数参数(*)

10.3.2.1 二维数组的地址

- 设二维数组a定义如下：
  int a[3][4]={{1,2,3,4},{5,6,7,8},{9,10,11,12}};
- C语言允许把二维数组看成多个一维数组，即可以把a理解为一维数组，它包含3个元素a[0]、a[1]和a[2]；而每个元素又是一个一维数组，即a[0]包含a[0][0]、a[0][1]、 a[0][2]和a[0][3]共4个元素， a[1]和a[2]也各包含4个元素。
- a是二维数组，a就代表二维数组的首地址，即&a[0][0]。（打印a[0][0]元素的值，用*printf("%d\n", *p);* 还是*printf("%d\n", **p);*？）
- a[0]、a[1]和a[2]是a的元素，但也是一维数组，因此，它们分别代表各一维数组（二维数组的行）的首地址。a[0]是元素a[0][0]的地址，即&a[0][0]； a[1]是元素a[1][0]的地址，即&a[1][0]；
- 数组元素有下标法和指针法两种引用方式，即a[0]等价于*a，a[1]等价于*(a+1)，因此，a、a[0]、*a和&a[0][0]都是表示元素a[0][0]的地址，a+1、a[1]和*(a+1)和&a[1][0]都是表示元素a[1][0]的地址。
- a[i][j]的地址是a[i]+j、*(a+i)+j或&a[i][j]。

10.3.2.1 二维数组的地址

- 例10.3.5二维数组元素地址示例，程序清单如下：

```
#include <stdio.h>
void main()
{
    int a[3][4]={{1,2,3,4},{5,6,7,8},{9,10,11,12}};
    printf("%d\n",a);              //输出&a[0][0]
    printf("%d\n", a+1);           //输出&a[1][0]
    printf("%d\n", *a+1);          //输出&a[0][1]
    printf("%d\n",a[0]+1);         //输出&a[0][1]
    printf("%d\n",*a[0]+1);        //输出a[0][0]+1
}
```

程序运行结果：

```
1245008
1245024
1245012
1245012
2
```

10.3.2.2 指向一维数组的指针

- C语言允许定义指向二维数组中的行（一维数组）的指针，然后应用指针变量灵活地处理二维数组。定义格式为：
  类型说明符 (*变量名)[长度];

  其中，“类型说明符”是数组元素的类型，“长度”是二维数组分解为多个一维数组时，每个一维数组的长度，即二维数组的行宽。定义语句中的圆括号是必须的，否则就成了“指针数组”。

10.3.2.2 指向一维数组的指针

例10.3.7指向行的指针变量应用示例：

```
#include <stdio.h>
void main()
{
        int a[3][4]={{1,2,3,4},{5,6,7,8},{9,10,11,12}},i,j,(*p)[4];
        p=a;
        for(i=0;i<3;i++)
        {       for(j=0;j<4;j++)
                        printf("%d\t", *(*(p+i)+j) );
                printf("\n");
        }
}
```

- 程序中语句“p=a;”表示把a[0]的地址赋给指针变量p，如果写成“p=a[0];”（会发生编译错误），则表示把a[0][0]的地址赋给指针变量p（这时p+1是什么？printf(“%d\n”,a);和printf(“%d\n”,a[0]);是否相同？），显然a[0][0]的地址与p的类型不相符。*(p+i)表示a[i]的首地址，*(p+i)+j则表示a[i][j]的地址，所以，* (*(p+i)+j)表示a[i][j]的值。

10.3.3 字符串与指针

- 在C语言中，使用字符数组存储字符串。例如：

  char str1[]="Hello";printf("str=%s\n",str1);

- C语言还允许用字符指针表示字符串，即定义一个指针，然后用指针指向字符串中的首字符。例如：

  char * str2="Hello"; printf("str=%s\n",str2);

•用字符数组表示字符串，系统为字符数组分配存储空间，数组名就是该存储空间的首地址；

•用字符指针变量表示字符串，指针变量只保存了字符串的首地址，而字符串本身并没有保存在字符指针中（实际上也无法存储），系统为字符串常量专门开辟了一块连续的存储空间，然后把该空间的首地址赋给字符指针。

10.3.3 字符串与指针

- **1、赋值运算**
- 对字符数组不能进行赋值运算，因为字符数组名是指针常量。而对字符指针可以进行赋值运算。例如：

  不能把char a[6]=“Hello”;写成：

  char a[6]; a=“Hello”;

  但可以把char *a=“Hello”;写成：

  char *a; a=“Hello”;

- **2、输入**
- 字符数组有确定的存储空间，可以通过外部设备把字符串输入到字符数组中。
- 而字符指针只是简单变量，不能把字符串输入到字符指针中，但可以把字符串输入到字符指针所指向的存储空间中。例如：

  char a[16]; scanf(“%s”,a); //正确

  char *a; scanf(“%s”,a);//可能会引起致命错误，因为a的值还没有确定

  char *a=“ ”; scanf(“%s”,a);//正确，但是输入字符个数不能超过空格数

10.3.3 字符串与指针

例10.3.10字符指针算术运算示例。

```
//程序1：
void main()
{
    char *a="ABCD";
    while(*a) printf("%s\n", a++);

}
```

程序1运行结果：

```
ABCD
BCD
CD
D
```

```
//程序2：
void main()
{
     char *a="ABCD";
     while(*a) printf("%c\n", *a++);
}
```

程序2运行结果：

```
A
B
C
D
```

10.3.4 指针数组

- 元素类型为指针的数组称为指针数组。定义指针数组的一般形式为：
  类型名 * 数组名[数组长度];
- 例如：
  int *p[5];
  定义了数组p，数组包含5个元素，各元素的类型是指向整型变量的指针。
- 也可以在定义指针数组时对指针数组进行初始化，例如：
  char * name[]={“lin yu”,”li qiang”,”wang hong”};
  其中，数组元素name[0]指向字符串”lin yu”, name[1]指向字符串”li qiang”，name[2]指向字符串”wang hong”。
- *特别要注意：指针数组定义和“指向一维数组的指针变量”定义这二者之间的差别。*

附件：课程教师和助教（2011-2012第2学期）

主讲教师：林子雨

单位：厦门大学信息科学与技术学院计算机科学系
办公地点：福建省厦门市思明区厦门大学海韵园
E-mail:ziyulin@xmu.edu.cn
个人主页：http://www.cs.xmu.edu.cn/linziyu

助教：林尚青

单位：厦门大学计算机科学系2010级硕士研究生
E-mail:lsq1015@qq.com
手机：15959206201

助教：赖明星

单位：厦门大学计算机科学系2011级硕士研究生
E-mail:joy_lmx@163.com
手机：18050056577

附件：课程FTP （2011-2012第2学期）

- FTP地址：ftp://218.193.53.74
- 用户名： stu_linziyu
- 密码：123456
- 目录：“下载教学内容”→”C语言”

附件：课程教材 （2011-2012第2学期）

- 《C语言程序设计（第2版）》
- 清华大学出版社，黄保和，江弋 编著
- 版次：2011年10月第2版
- ISBN:978-7-302-26972-4
- 定价：35元

Department of Computer Science, Xiamen University, May 8, 2012