数组

- 基本数据类型的特点：
  - 变量是单值的，由变量名直接访问
  - 数据是不可再分的原子类型
- 构造类型的特点：
  - 由多个分量组成的数据结构
  - 每个分量都是一个变量（简单变量或构造变量）
  - 其数据难以用变量名直接整体访问

问题提出：

一个年级有 200 名学生，求全年级同学英语平均成绩。

可能的思路：

(1) 设 200 个变量 e1,e2,…,e200 用于存放每个同学的英语成绩；

(2) 计算 (e1+e2+…+e200)/200.0

存在问题：

(1) 变量过多过繁琐，如果是 20000 人怎么解决？

(2) 没有反映出数据间的内在联系

解决思路：

运用数组

➢数组基本概念

- 程序中为了处理方便常把具有相同类型的多个值有序组织起来；这些值称为元素。
- 数组是一种构造数据类型，它将相同类型的变量组合起来，用一个名称来表示。
- 相同性质的一组数据的集合——数组元素数据类型相同。
- 为了声明数组，需要说明数组元素的类型和数量（长度）。

1. 一维数组

[ ]：数组运算符
单目运算符
优先级最高
左结合
不能用 ( )

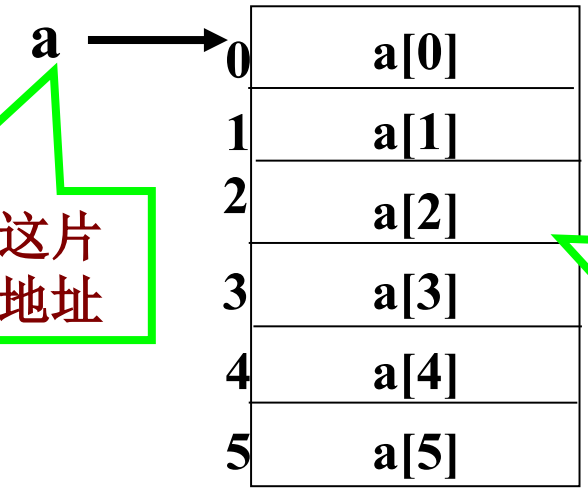
合法标识符

❑一维数组的定义

❖定义方式：数据类型 数组名 [常量表达式]；

表示元素个数，
下标从 0 开始。

例： int a[6];



数组名表示数组这片
连续内存的起始地址

编译时分配连续内存
内存字节数 = 数组长度 *
sizeof(元素数据类型)

❑一维数组的引用

- ❖数组必须先定义，后使用。
- ❖只能逐个引用数组元素，不能一次引用整个数组。
- ❖数组元素表示形式：数组名 [下标]

  其中：下标可以是常量或整型表达式。

例 `int a[10];`

`printf("%d",a);` (×)

必须 `for(j=0;j<10;j++)`

`printf("%d\t",a[j]);` (✓)

例 `int data[5];`

`data[5]=10;` (×)

/*C 语言对数组不作越界检查，使用时要注意 */

□ 一维数组的初始化

- 在定义数组时对数组元素赋以初值，将数组元素的初值依次放在一对花括弧内。
- 数组不初始化，其元素值为随机数；对 static 数组元素，不赋初值，系统会自动赋以 0 值。
- 可以只给部分数组元素赋初值，没有赋值的值为 0 。
- 在对全部数组元素赋初值时，可以不指定数组长度。

例： int a[5] = {1,2,3};

int a[10] = {0};

int a[ ] = {1,2,3,4,5};

指定初始化式 (C99)

- 经常有这样的情况：数组中只有相对较少的元素需要初始化，而其他元素可以默认赋值。例如：

  int a[15] = {0, 0, 29, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 48};

- 上面的例子可以使用指定初始化式写为：

  int a[15] = {[2] = 29, [9] = 7, [14] = 48};

  - 大括号中的中括号的数字称为**指示符**。

- 指定初始化式可以使赋值变得更简短、更易读。而且赋值的顺序也不再是一个问题，如：

  int a[15] = {[14] = 48, [9] = 7, [2] = 29};

指定初始化式 (C99)

- 指示符必须是整型常量表达式。
- 如果待初始化的数组长度为 n ，则每个指示符的值都必须在 0 和 n-1 之间。
- 如果数组的长度是省略的，指示符可以是任意非负整数。编译器将根据最大的指示符推断出数组的长度。例：下面这个数组包含 24 个元素：

  int b[ ] = {[5] = 10, [23] = 13, [11] = 36, [15] = 29};
- 初始化式中可以同时使用老方法 (逐个元素初始化) 和新方法 (指定初始化式) ：

  int c[10] = {5, 1, 9, [4] = 3, 7, 2, [8] = 6};

❑程序举例

例：读 10 个整数存入数组，找出其中最大值和最小值。

步骤：
1. 输入 :for 循环输入 10 个
2. 处理 :
(a) 先令 max=min=x[0]
(b) 依次用 x[i] 和 max,min
环)
若 max<x[i], 令 max=
若 min>x[i], 令 min=x
3. 输出 :max 和 min

```
#define SIZE 10
int main(void)
{   int  x[SIZE], i, max, min;
    printf("Enter 10 integers:\n");
    for(i=0;i<SIZE;i++)
    {   printf("%d:",i+1);
        scanf("%d",&x[i]);
    }
    max=min=x[0];
    for(i=1; i<SIZE; i++)
    {
      if(max<x[i])  max=x[i];
      if(min>x[i])  min=x[i];
    }
    printf("Maximum value is %d\n",max);
    printf("Minimum value is %d\n",min);
    return 0;
}
```

■ 读程序，写出程序运行的结果。

```
#include <stdio.h>
int main( )
{
        unsigned int x=99;
        int i, y[6]={0,0,0,0,0,0};
        for(i=5; i>0; i--)
        {
                y[i]=x%8;
                x=x/8;
                if(x = = 0) break;
        }
        for(i=0;i<6;i++)
                if(y[i]!=0) printf("%d",y[i]);
        return 0 ;
}
```

2. 二维数组及多维数组

❑二维数组的定义

❖ 定义方式：

数据类型　数组名[常量表达式][常量表达式]；



注：如果有一个一维数组，它的每一个元素又是类型相同的一维数组时，就形成一个二 维数组。

数组的类型相同指：大小相同；其中元素的类型也相同。

❑多维数组的定义方法与二维数组类似：

例： char a[2][3][4];

❖数组元素的存放顺序

- 原因：内存是一维的
- 二维数组：按行序优先
- 多维数组：最右下标变化最快

int c[2][3][4]

0	c[0][0][0]
1	c[0][0][1]
2	c[0][0][2]
3	c[0][0][3]
4	c[0][1][0]
5	c[0][1][1]
6	c[0][1][2]
7	c[0][1][3]
	c[0][2][0]
	c[0][2][1]
	c[0][2][2]
	c[0][2][3]
	c[1][0][0]
	c[1][0][1]
	c[1][0][2]
	c[1][0][3]
	c[1][1][0]
	c[1][1][1]
	c[1][1][2]
	c[1][1][3]
20	c[1][2][0]
21	c[1][2][1]
22	c[1][2][2]
23	c[1][2][3]

int a[3][2]

0	a[0][0]
1	a[0][1]
2	a[1][0]
3	a[1][1]
4	a[2][0]
5	a[2][1]

a[0][0] a[0][1]
a[1][0] a[1][1]
a[2][0] a[2][1]

❖二维数组元素的引用

形式： 数组名[下标][下标]

❖二维数组理解

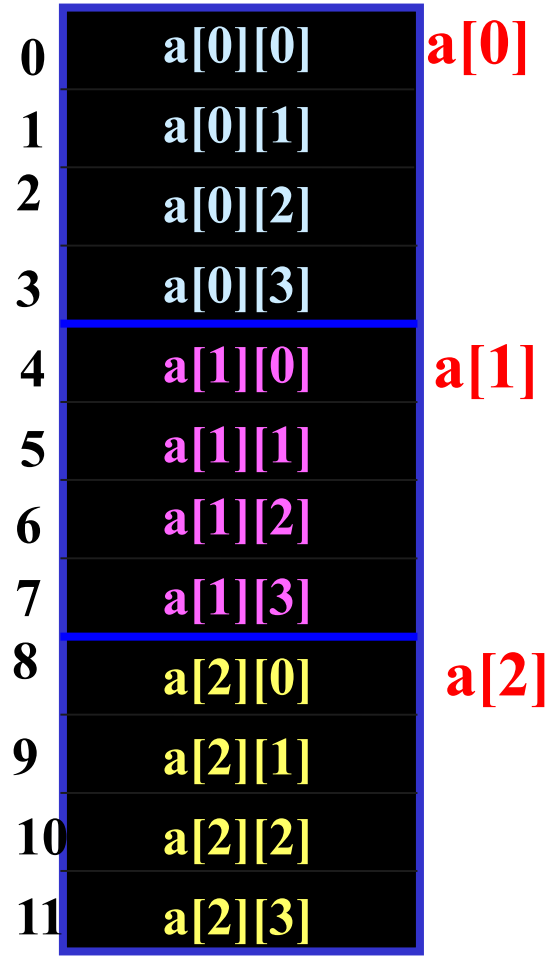
二维数组 a 是由 3 个元素组成

例　　int a[3][4];

a[0]	a[0][0]	a[0][1]	a[0][2]	a[0][3]
a[1]	a[1][0]	a[1][1]	a[1][2]	a[1][3]
a[2]	a[2][0]	a[2][1]	a[2][2]	a[2][3]

行名

每个元素 a[i] 由包含 4 个元素的一维数组组成



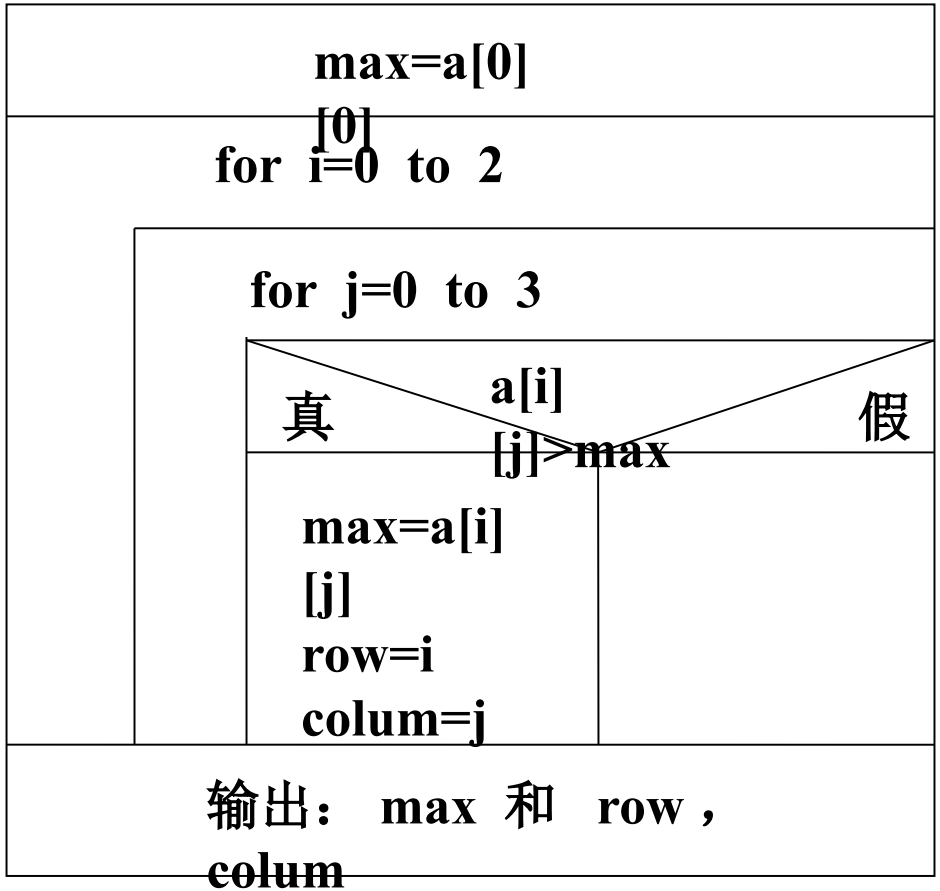
❖二维数组元素的初始化

- 分行初始化：例　　int a[2][3]={{1,2,3},{4,5,6}};
- 按元素排列顺序初始化：　int a[2][3]={1,2,3,4,5,6};
- 部分初始化：　int a[2][3]={1,2,4}; int a[2][3]={{1,2},{4}};
- 省略行数：　　int a[ ][3]={1,2,3,4,5};　int a[ ][3]={{1},{4,5}};
- C99 的指定初始化式对多维数组也有效。

  例如：可以这样创建　2 × 2　的单位矩阵：

  double ident[2][2] = {[0][0] = 1.0, [1][1] = 1.0};

例：求二维数组中最大元素值及其行号和列号。




```
int main(void)
{  int a[3][4]={{1,2,3,4},
                {9,8,7,6},
                {-10,10,-5,2}};
   int i, j, row=0, colum=0, max;
   max=a[0][0];
   for(i=0;i<=2;i++)
     for(j=0;j<=3;j++)
        if(a[i][j]>max)
        {
            max=a[i][j];
            row=i;
            colum=j;
        }
   printf("max=%d,row=%d,
colum=%d\n",max,row,colum);
   return 0;
}
```

例：读入下表中值到数组，分别求各行、各列及表中所有数之和。

12	4	6	
8	23	3	
15	7	9	
2	5	17	
37	39	35	111

```
int main( )
{
    int x[5][4], i, j;
    for(i=0;i<4;i++)
      for(j=0;j<3;j++)
          scanf("%d",&x[i][j]);
    for(i=0;i<3;i++)
        x[4][i]=0;
    for(j=0;j<5;j++)
        x[j][3]=0;
    for(i=0;i<4;i++)
      for(j=0;j<3;j++)
      {
          x[i][3]+=x[i][j];
          x[4][j]+=x[i][j];
          x[4][3]+=x[i][j];
      }
```

```
  for(i=0;i<5;i++)
  {
     for(j=0;j<4;j++)
         printf("%5d\t",x[i][j]);
     printf("\n");
  }
  return 0;
}
```

3. 字符数组和字符串

❑字符数组

- ❖定义
- ❖字符数组的初始化
  - 逐个字符赋值
  - 用字符串常量
- ❖字符数组的引用

例　char c[10], ch[3][4];

逐个字符赋值

用字符串常量

有问题！

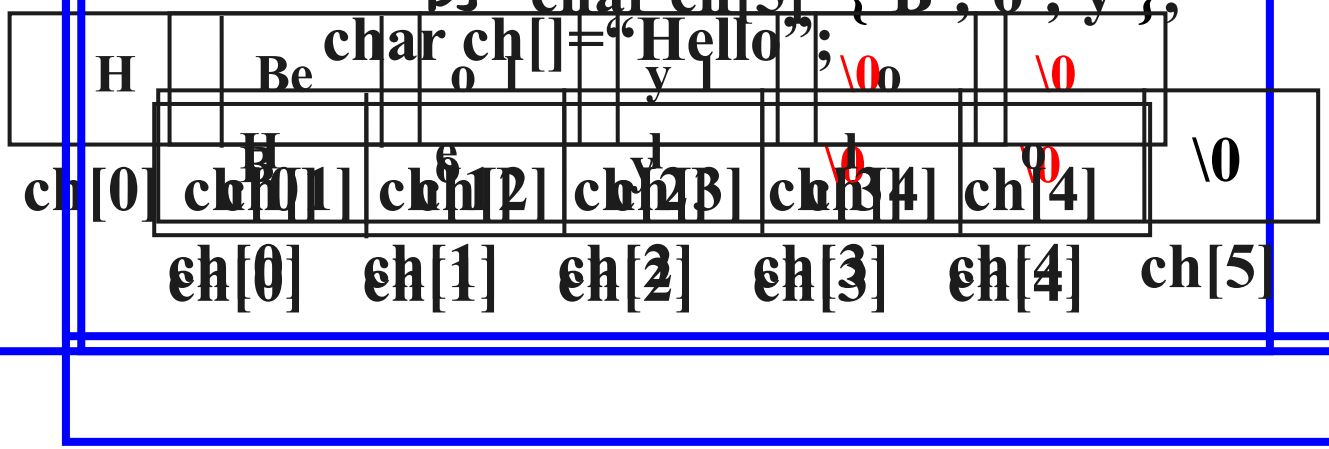
例　char ch[6]={“Hello”};

例　char str[5]={'B','o','y','\0'};

例　char ch[6]=“Hello”;

例　char ch[5]={‘B’,’o’,’y’};

char ch[]=“Hello”;



❑字符串

❖字符串及其结束标志

- 无字符串变量，用字符数组处理字符串
- 字符串结束标志：‘\0’

例 “hello”共5个字符，在内存占6个字节 字符串长度

h	e	l	l	o	\0

104	101	108	108	111	0

内存存放字符 ASCII 码

❖字符串的输入输出

- 逐个字符 I/O ： %c
- 整个字符串 I/O ： %s

例 用 %c

```
int main( )
{   char  str[5];
    int i;
    for(i=0;i<5;i++)
       scanf("%c", &str[i]);
    for(i=0;i<5;i++)
       printf("%c", str[i]);
    return 0 ;
}
```

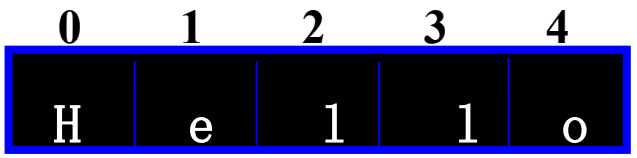
例 用 %s

```
int main( )
{   char  str[5];
    scanf("%s", str);
    printf("%s", str);
    return 0;
}
```

用字符数组名，不要加
&
输入串长度＜数组长度
遇空格或回车结束
自动加‘\0’

用字符数组名，
遇‘\0’结束

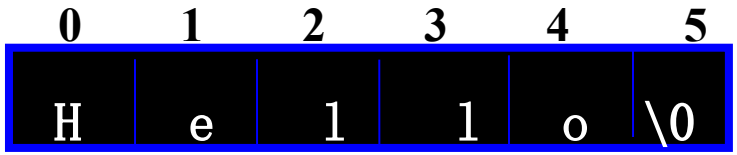
例
```
int main( )
{
    char a[5]={'H','e','l','l','o'};
    printf("%s",a);
    return 0;
}
```



结果： Hello#-

用“%s”输出时，
遇‘\0’结束

例
```
int main( )
{
    char a[ ]="Hello";
    printf("%s",a);
    return 0;
}
```



结果：

```
main( )
{
   int i;
   char a[5];
   scanf("%s", a);
   for(i=0;i<5;i++)
        printf("%d,",a[i]);
}
```

输入字符串长度 < 数组维数

h	e	l	\0	

h	e	l	l	

\0

h	e	l	l	

o

运行情况：

（1）若输入　hel，正常

（2）若输入　hell，正常

（3）若输入　hello，用 %s 输出时，会出现问题

❑字符串输入举例

```
int main( )
{
   char a[15],b[5],c[5];
   scanf("%s%s%s",a,b,c);
   printf("a=%s\nb=%s\nc=%s\n",a,b,c);
   scanf("%s", a);
   printf("a=%s\n",a);
    return 0;
}
```

运行情况：
输入： How are you?
输出： a=How
b=are
c=you?
输入： How are you?
输出：

scanf 中 %s 输入时，遇空格或回车结束

H	o	w	\0											
a	r	e	\0											
y	o	u	?	\0										

❖常用的字符串处理函数

包含头文件　　string.h

puts gets 包含头文件　　stdio.h

◆字符串输出函数 puts

格式：　puts(字符数组)

功能：向显示器输出字符串（输出完，换行）

说明：字符数组必须以‘\0’ 结束

◆字符串输入函数 gets

格式：　gets(字符数组)

功能：从键盘输入一以回车结束的字符串放入字符数组中，并自动加‘\0’

说明：输入串长度应小于字符数组维数

VS 中用：gets_s

```
char str[10];
gets_s(str);
gets_s(str, 10);
```

◆字符串连接函数 strcat

格式： strcat(字符数组 1, 字符数组 2)

功能：把字符数组 2 连到字符数组 1 后面

返值：返回字符数组 1 的首地址

说明： 字符数组 1 必须足够大

连接前 , 两串均以 ‘ \0’ 结束 ; 连接后 , 串 1 的 ‘ \0’ 取消，新串最后加 ‘ \0’

例 char str1[20],str2[20];

str1={“Hello!”}; (×)

str2=str1; (×)

◆字符串拷贝函数 strcpy

格式： strcpy(字符数组 1, 字符串 2)

功能：将字符串 2 ， 拷贝到字符数组 1 中去

返值：返回字符数组 1 的首地址

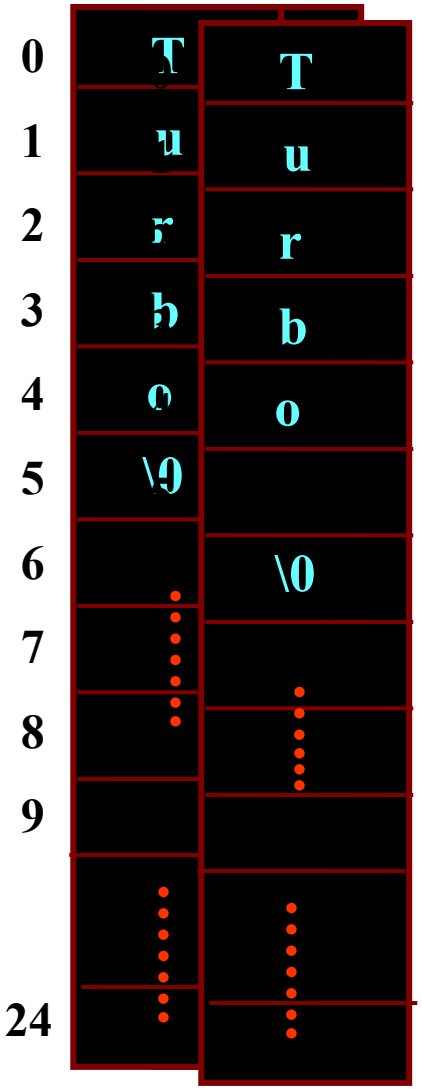
说明： 字符数组 1 必须足够大

拷贝时 ‘ \0’ 一同拷贝

不能使用赋值语句为一个字符数组赋值

例： strcpy 与 strcat

```
#include <string.h>
#include <stdio.h>
int main(void)
{
   char destination[25];
   char blank[] = " ", c[]= "C++",
        turbo[] = "Turbo";
   strcpy(destination, turbo);
   strcat(destination, blank);
   strcat(destination, c);
   printf("%s\n", destination);
   return 0;
}
```



◆字符串比较函数 strcmp

格式： strcmp(字符串 1, 字符串 2)

功能：比较两个字符串

比较规则：对两串从左向右逐个字符比较（ASCII 码），直到遇到不同字符或‘\0’为止

返值：返回 int 型整数，
a. 若字符串 1< 字符串 2， 返回负整数
b. 若字符串 1> 字符串 2， 返回正整数
c. 若字符串 1== 字符串 2， 返回零

◆字符串长度函数 strlen

格式： strlen(字符数组)

功能：计算字符串长度

返值：返回字符串实际长度，不包括‘\0’在内

例 对于以下字符串， strlen(s) 的值为：

（1） char s[10]={‘A’,‘\0’,‘B’,‘C’,‘\0’,‘D’};

（2） char s[ ]=“\t\v\\\0will\n”;

（3） char s[ ]=“\x69\082\n”

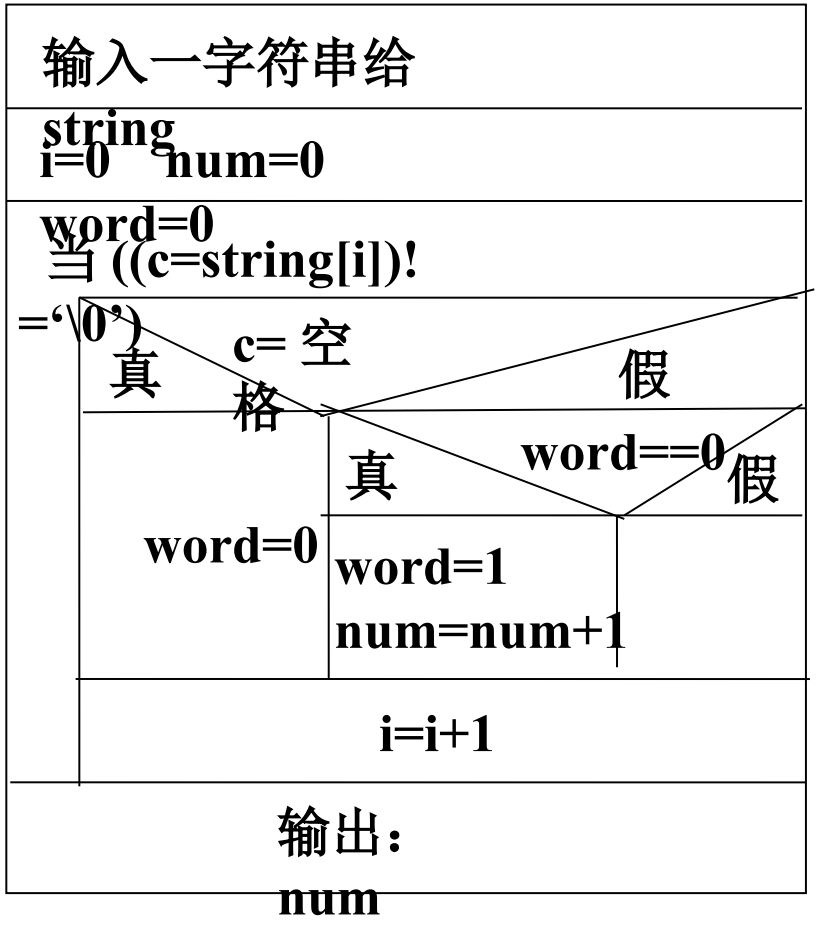
答案： 1 3

例　strcmp 与 strlen 举例

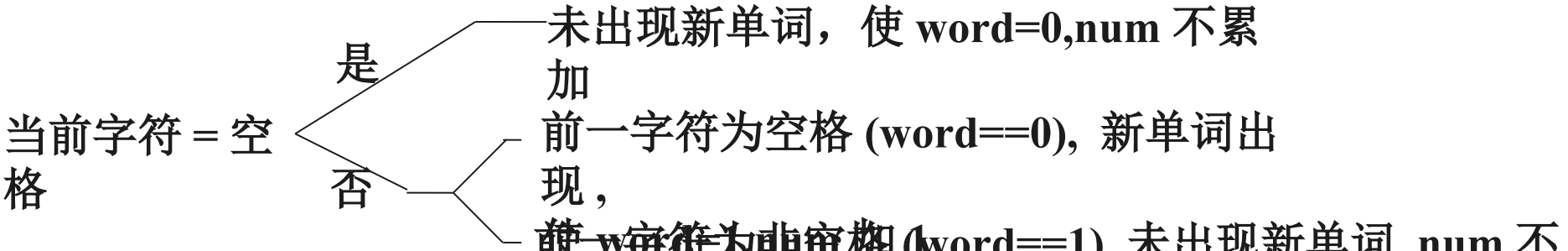
```
#include <string.h>
#include <stdio.h>
int main(void)
{   char str1[ ] = ”Hello!”, str2[ ] = ”How are you?”, str[20];
    int  len1, len2, len3;
    len1=strlen(str1);      len2=strlen(str2);
    if(strcmp(str1, str2)>0)
    {   strcpy(str,str1);     strcat(str,str2);   }
    else  if (strcmp(str1, str2)<0)
    {   strcpy(str,str2);     strcat(str,str1);   }
    else    strcpy(str, str1);
    len3=strlen(str);
    puts(str);
    printf(”Len1=%d,Len2=%d,Len3=%d\n”,len1,len2,len3);
    return 0;
}
```

```
How are you?Hello!
Len1=6,Len2=12,Len3=18
```

例：输入一行字符，统计其中有多少个单词




```
int main(void)
{   char string[81];
    int  i, num=0, word=0;
    char c;
    gets(string);
    for(i=0;(c=string[i])!='\0';i++)
      if(c==' ')  word=0;
      else if(word==0)
      {   word=1;  num++;   }
    printf("There are %d words
              in the line\n", num);
    return 0;
}
```



例输入： I ␣am ␣a ␣boy.

当前字符	I	␣	a	m	␣	a	␣	b	o	y	.
是否空格	否	是	否	否	是	否	是	否	否	否	否
word 原值	0	1	0	1	1	0	1	0	1	1	1
新单词开始否	是	未	是	未	未	是	未	是	未	未	未
word 新值	1	0	1	1	0	1	0	1	1	1	1
num 值	1	1	2	2	2	3	3	4	4	4	4

- **计算一个英文句子中最短单词的长度（单词长度为单词中字母的个数）假设该英文句子中只含有字母和空格，在空格之间连续的字母串称为单词，当输入回车符时标志句子结束。**

```
#include <stdio.h>
#define N 100
int main( )
{  char str[N];
   int minlen = N, len, inword = 0, i = 0;
   printf(" 请输入句子 , 字符小于 %d:",N);
   fgets(str,N,stdin);
   do
   {
      if (inword)
      {
         if ((str[i] != ' ') && str[i]) len++;
         else
         {  minlen = (len < minlen) ? len : minlen;
            inword = len = 0;
         }
      }
      else if ((str[i] != ' ') && str[i]) { inword = 1; len = 1; }
   } while (str[i++]);
   printf(" 最短单词长度 :%d\n", minlen);
   return 0;
}
```

例　float weight[10.3]; ✗

例　int a[-100]; ✗

例　int a[10];
float i=3; ✗
a[i]=10;

例　int a[5];
a={2,4,6,8,10}; ✗

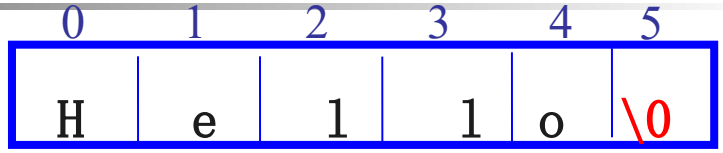
例　int a[][10];
float f[2][]={1.2, 2.2}; ✗

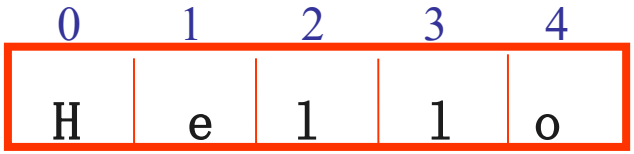
比较　int a[2][3]={{5,6},{7,8}};

与　int a[2][3]={5,6,7,8};

比较　char str[]="Hello";

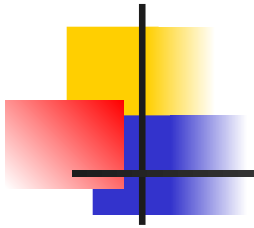
char str[]={'H','e','l','l','o'};





$$\begin{bmatrix} 5 & 6 & 0 \\ 7 & 8 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 5 & 6 & 7 \\ 8 & 0 & 0 \end{bmatrix}$$

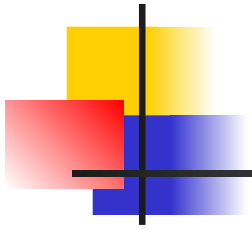


常量数组

- 无论一维数组还是多维数组，可以在数组声明开始加关键字 const ， 以避免程序对此种数组进行修改。

```
const char hex_chars[ ] =
  {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'D', 'E', 'F'};
```

- 声明数组是 const 的好处：
  - 它表明程序不会改变数组元素的值。
  - 对编译器发现错误也很有帮助。
- const 的使用不仅限于数组，但用于定义常量数组特别有用。



变长数组 (C99)

- 变长数组的长度可以用变量来指定，也可以用任意整型表达式：

```
int a[3*i+5];
int b[j+k];
```

- 像其它数组一样，变长数组也可以是多维的：

```
int c[m][n];
```

- 变长数组的限制：
  - 没有静态存储
  - 没有初始化式

本章小结

- 数组是一个顺序排列的有相同类型的若干元素的集合；
- C 数组的所有元素均按行主序方式存放在一个连续的存储空间中，数组名就是这个存储空间的首地址（即第一个元素的存放地址）。
- 数组类型说明由类型说明符、数组名、数组长度　（数组元素个数）三部分组成。数组元素采用下标变量形式表示；数组的类型是指元素取值的类型。
- 定义数组时必须用常量表达式来定义数组元素的个数；C99 允许变长数组。
- 在 C 语言中，数组的下标从 0 开始，最后一个下标是数组的长度减 1； C 的编译器不检查数组下标越界错误；