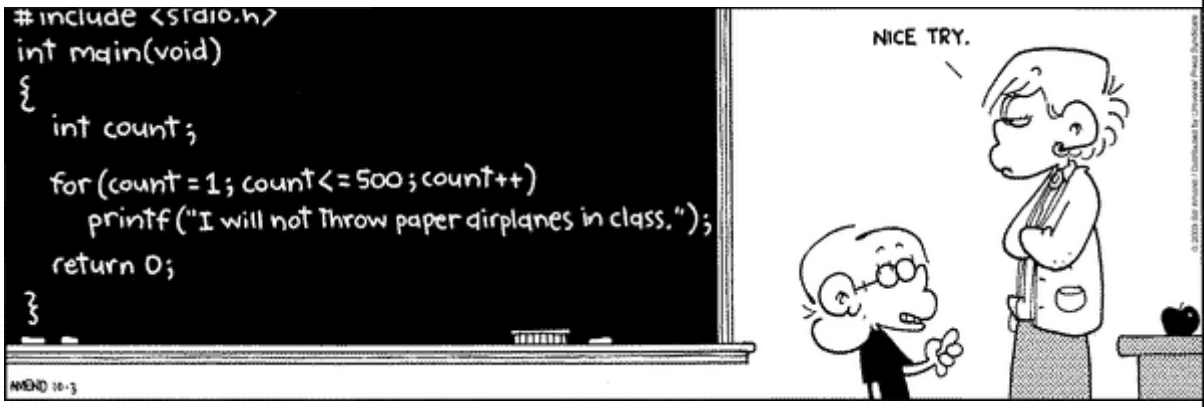
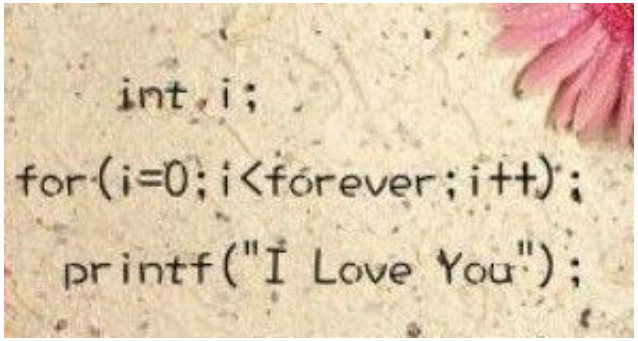
第5章 循环结构程序设计（2）

复习回顾

- 上次课的内容：
  - while语句
  - do…while语句
  - for语句
    - 看看机智的熊孩子怎样应对老师的惩罚
    - 以及C语言程序员的情书
  - 循环的嵌套（待续）





循环实现举例：百钱买百鸡

- 《算经》有云：鸡翁（公鸡）一值钱五，鸡母（母鸡）一值钱三，鸡雏（小鸡）三值钱一，百钱买百鸡，问鸡翁、鸡母、鸡雏各几只？
- **解题思路：**假设鸡翁、鸡母、鸡雏的个数分别是x、y、z，根据题意给定用百钱买百鸡，若全买公鸡最多买20只，显然x的值在0～20之间；同理，y的取值范围在0～33之间，可得到下面的不定方程：

$$5x+3y+z/3=100$$

$$x+y+z=100$$

可通过**对未知数可变范围**的**穷举**，验证方程在什么情况下成立，从而得到相应的解。

循环实现举例：百钱买百鸡

- 《算经》有云：鸡翁（公鸡）一值钱五，鸡母（母鸡）一值钱三，鸡雏（小鸡）三值钱一，百钱买百鸡，问鸡翁、鸡母、鸡雏各几只？
- **第一步：确定实现方式。**
  - 根据题意，我们需要穷举x，y，z的所有合理的组合。其中z的值可以依赖于x，y，即z＝100-x-y，所以程序应包括两层循环，一层循环控制公鸡的数量（即x），一层循环控制母鸡的数量（即y），选用两层嵌套的for循环实现比较合理。
  - 让我们用外层循环控制公鸡数量，内层循环控制母鸡数量

循环实现举例：百钱买百鸡

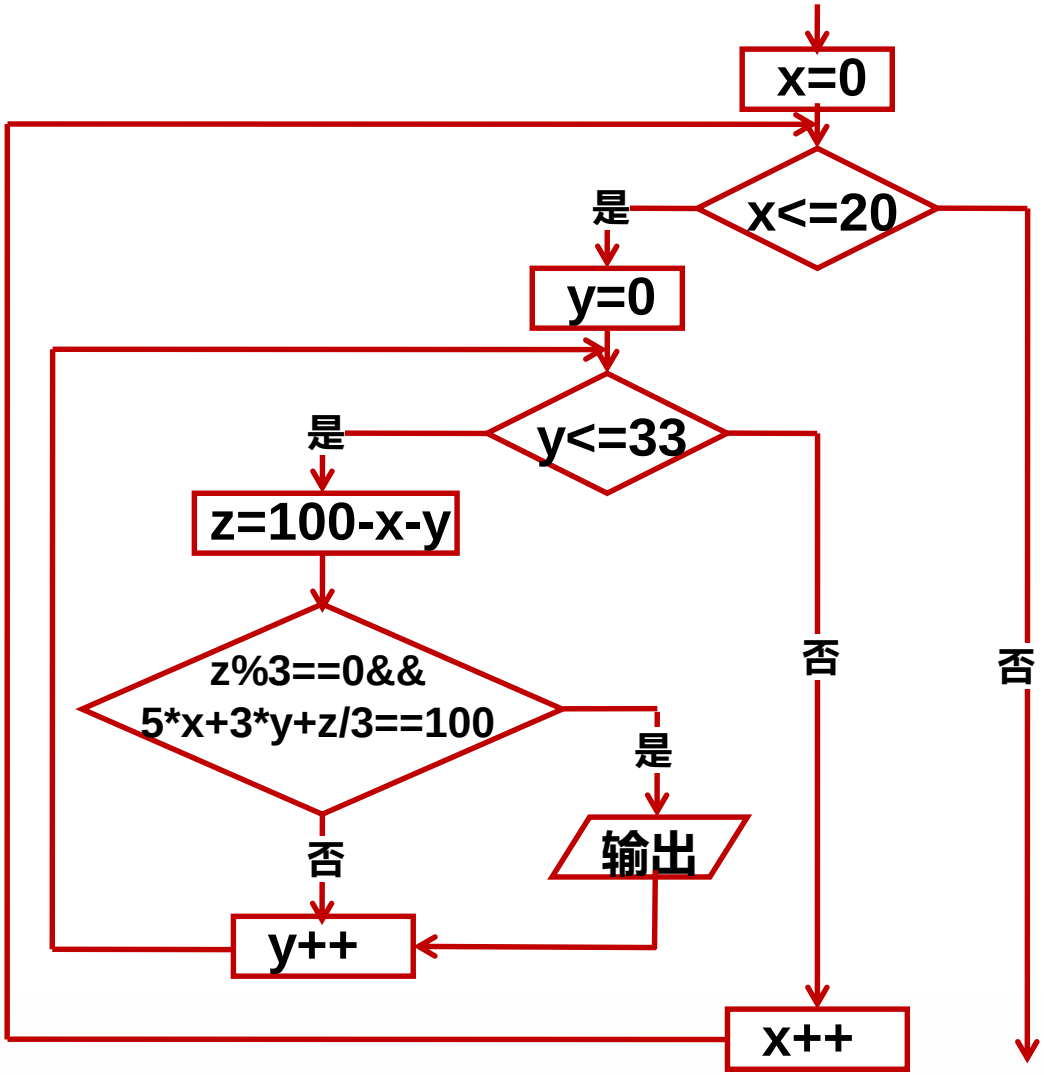
- 《算经》有云：鸡翁（公鸡）一值钱五，鸡母（母鸡）一值钱三，鸡雏（小鸡）三值钱一，百钱买百鸡，问鸡翁、鸡母、鸡雏各几只？
- 第二步：确定每个循环的终止条件，初始化语句和更新语句
  - ◆先确定内层循环。因为内层控制的是母鸡数量，其值从0递增到33。假设内层循环计数变量为y，则循环的终止条件为y<=33，初始化语句为y=0，更新操作需要x的值每次递增1，语句为y++;
  - ◆再确定外层循环。因为外层控制的是公鸡数量，其值从0递增到20，假设外层循环计数变量为x，则循环终止条件x<=20，初始化语句x=0，更新操作需要x每次递增1，语句为x++。

循环实现举例：百钱买百鸡

- 《算经》有云：鸡翁（公鸡）一值钱五，鸡母（母鸡）一值钱三，鸡雏（小鸡）三值钱一，百钱买百鸡，问鸡翁、鸡母、鸡雏各几只？
- **第三步：确定每个循环的循环体**
  - ◆先确定内层循环的循环体。每次进入内层循环后，因为已经有了公鸡和母鸡的数量，所以首先要确定小鸡的数量，即z=100-x-y。接下来，因为x、y、z数量需满足条件(5*x+3*y+z/3==100)并且z必须能被3整除，即z%3==0。这些都是判断条件，可用if语句实现。判断条件成立则输出各种鸡的数量。
  - ◆再确定外层循环的循环体。因为外层循环只是用来控制公鸡的数量，所以其循环体只包括内层循环。

循环实现举例：百钱买百鸡

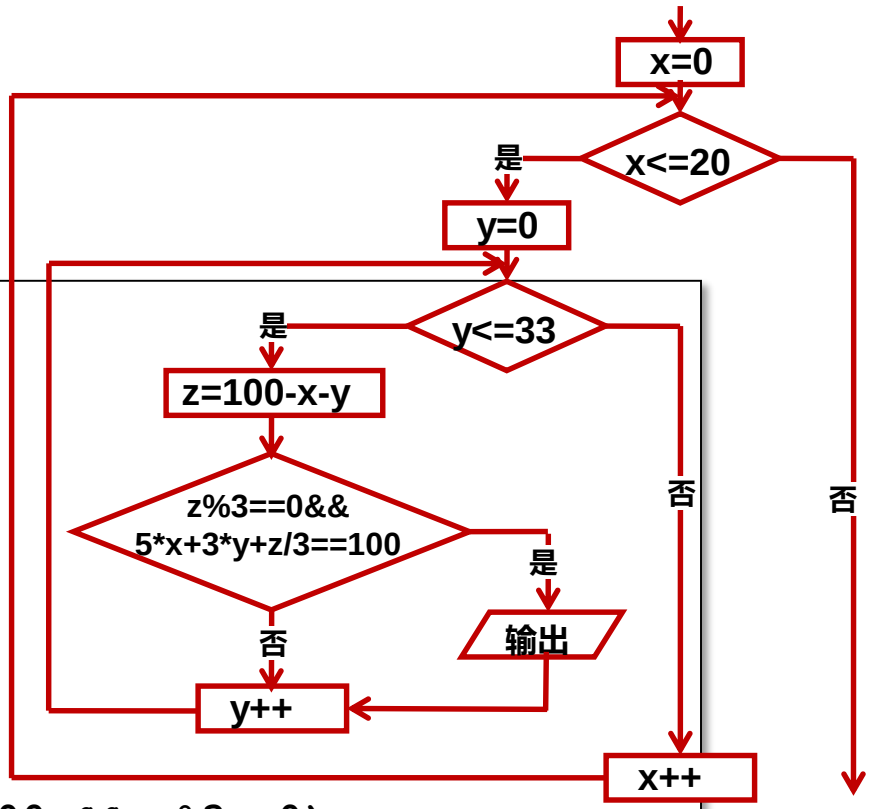
- 《算经》有云：鸡翁（公鸡）一值钱五，鸡母（母鸡）一值钱三，鸡雏（小鸡）三值钱一，百钱买百鸡，问鸡翁、鸡母、鸡雏各几只？
- **第四步：画出程序的流程图**



循环实现举例：百钱买百鸡

➢ 第五步：编程实现

```
1. #include <stdio.h>
2.
3. int main()
4. {
5.     int x,y,z;
6.
7.     for (x=0; x<=20; x++)
8.     {
9.         for (y=0; y<=33; y++)
10.        {
11.            z=100-x-y;
12.            if (5*x+3*y+z/3==100 && z%3==0)
13.               printf("%d %d %d\n",x,y,z);
14.        }
15.    }
16.    return 0;
17.}
```



break语句的解释

- 一般形式：`break;`
- 可应用于：
  - switch语句的case分支中
  - 循环结构的循环体中
    - for/while/do...while语句的循环体中
    - 作用是在满足某种条件时跳出循环，故常与if搭配
- 注意，break不能用于除switch语句和循环语句之外的任何语句中

循环体中的break举例

- 思考：下列程序执行后sum的值是多少？

```
1. #include <stdio.h>
2. int main()
3. {
4.      int i, sum = 0;
5.      for (i=1; i<11; i++)
6.      {
7.          if (i==5)
8.              break;
9.          sum += i;
10.     }
11.     printf("sum=%d\n",sum);
12.     return 0;
13.}
```

sum=10

多层循环中的break举例

➢思考：下列程序执行后输出结果是什么？

```
1. #include <stdio.h>
2. int main()
3. {
4.     int i,j;
5.     for (i=0; i<5; i++)
6.     {
7.         for (j=0; j<5; j++)
8.         {
9.             if (j==i+1)
10.                break;
11.            printf("*");
12.        }
13.        printf("\n");
14.    }
15.    return 0;
16.}
```

```
*
**
***
****
*****
Press any key to continue
```

注意：在多层循环中，一个break语句只向外跳一层。

用break提前终止循环：募捐

- 在全系1000学生中，征集慈善募捐，当总数达到10万元时就结束，统计此时捐款的人数，以及平均每人捐款的数目
- 编程思路：
  - 循环次数不确定，但最多循环1000次
    - 在循环体中累计捐款总数
    - 用if语句检查是否达到10万元
    - 如果达到就不再继续执行循环，终止累加
  - 计算人均捐款数，需统计总捐款数和捐款人数

用break提前终止循环：募捐

```
1. #include <stdio.h>
2. #define SUM 100000      //指定符号常量SUM代表100000
3. int main()
4. {
5.     float amount,aver,total=0;
6.     int i;
7.     for (i=1;i<=1000;i++)                    // 应该执行1000次
8.     {
9.          printf("please enter amount:");
10.         scanf("%f",&amount);
11.         total= total+amount;
12.         if (total>=SUM)                     //达到10万，提前结束循环
13.             break;
14.    }
15.    aver=total / i ;
16.    printf("num=%d\naver=%10.2f\n",i,aver);
17.    return 0;
18.}
```

用break提前终止循环：募捐

```
1. #include <stdio.h>
2. #define SUM 100000
3. int main()
4. {
5.      float amount,aver,total;
6.      int i;
7.      for (i=1,total=0;i<=1000;i++)
8.      {
9.           printf("please enter amount:");
10.          scanf("%f",&amount);
11.          total= total+amount;
12.          if (total>=SUM) break;
13.     }
14.     aver=total / i ;   // i是实际捐款人数
15.     printf("num=%d\naver=%10.2f\n",i,aver);
16.     return 0;
17.}
```

```
please enter amount:12000
please enter amount:24600
please enter amount:3200
please enter amount:5643
please enter amount:21900
please enter amount:12345
please enter amount:23000
num=7
aver=  14669.71
```

continue语句的解释

- 一般形式：

```
continue;
```

- 可应用于：
  - ◆for/while/do...while语句的循环体中
    - ●作用是在满足某种条件时中止本次循环，就是跳过循环体中尚未执行的语句，接着进行下一次是否执行循环的判断
    - ●总是与if语句连在一起，用来加速循环。
- 注意，continue语句不能用于循环语句之外的任何语句中

循环体中的continue举例

- 思考：下列程序执行后sum的值是多少？

```
1. #include <stdio.h>
2. int main()
3. {
4.      int i,sum = 0;
5.      for (i=1; i<11; i++)
6.      {
7.          if (i==5)
8.              continue;
9.          sum += i;
10.     }
11.     printf("sum=%d\n",sum);
12.     return 0;
13.}
```

sum=50

多层循环中的continue举例

➢思考：下列程序执行后输出结果是什么？

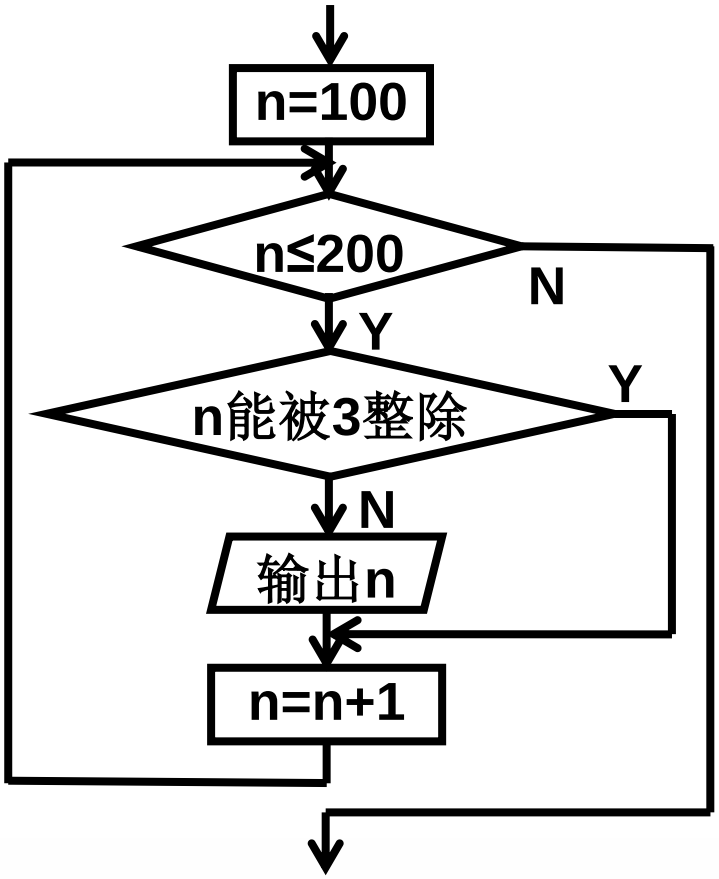
```
1. #include <stdio.h>
2. int main()
3. {
4.     int i,j;
5.     for (i=0; i<5; i++)
6.     {
7.         for (j=0; j<5; j++)
8.         {
9.             if (j==i+1)
10.                continue;
11.            printf("*");
12.        }
13.        printf("\n");
14.    }
15.    return 0;
16.}
```

```
*****
*****
*****
*****
******
Press any key to continue
```

注意：在多层循环中，一个continue语句只跳过本层循环一次

continue应用举例：不能整除

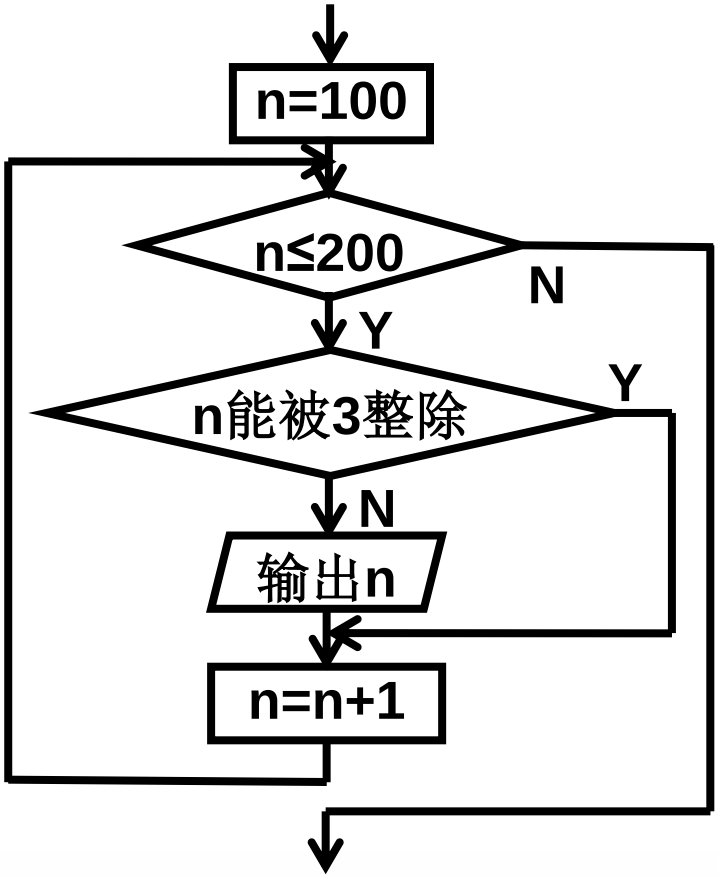
- 要求输出100～200之间的不能被3整除的数。
- 编程思路：
  - 对100到200之间的每一个整数进行检查
  - 如果不能被3整除，输出，否则不输出
  - 无论是否输出此数，都要接着检查下一个数(直到200为止)。



continue应用举例：不能整除

- 要求输出100～200之间的不能被3整除的数。

```
1. #include <stdio.h>
2. int main()
3. {
4.     int n;
5.     for (n=100; n<=200; n++)
6.     {
7.         if (n%3==0)
8.             continue;
9.         printf("%d ",n);
10.    }
11.    printf("\n");
12.    return 0;
13.}
```



break与continue的区别

- continue语句只结束本次循环，而不是终止整个循环的执行

- break语句结束整个循环过程，不再判断执行循环的条件是否成立

简单数据类型的局限

- 前几章使用的变量都属于基本类型，例如整型、字符型、浮点型数据，这些都是简单的数据类型。
- 有时，面对大量数据，只用简单的数据类型是不够的，难以反映出数据的特点，也难以有效地进行处理。
- 虽然循环可以利用屈指可数的变量处理大量数据，但这些数据只是“过客”，只参与运算，并没有保存。需要“先存储再运算”时就会有麻烦。

怎样“先存储再运算”

- 输入4个学生分数，并输出总分和平均分

```
1.  #include <stdio.h>
2.
3.  int main()
4.  {
5.    int i, tmp, sum=0;
6.
7.    for (i=0; i<4; i++)
8.    {
9.      printf("第%d个:", i+1);
10.     scanf("%d", &tmp);
11.     sum += tmp;
12.   }  //不存储学生分数，算过就找不到了

13.   printf("总分: %d\n", sum);
14.   printf("平均分: %.2f\n", sum/4.0);
15.
16.   return 0;
17.}
```

```
1.  #include <stdio.h>
2.
3.  int main()
4.  {
5.    int DaBao,ErBao,SanBao,SiBao,sum=0;

6.    printf("第1个"); scanf("%d", &DaBao);
7.    printf("第2个"); scanf("%d", &ErBao);
8.    printf("第3个"); scanf("%d", &SanBao);
9.    printf("第4个"); scanf("%d", &SiBao);
10.
11.   sum += DaBao;   //存储学生分数，但很繁琐
12.   sum += ErBao;
13.   sum += SanBao;
14.   sum += SiBao;

15.   printf("总分: %d\n", sum);
16.   printf("平均分: %.2f\n", sum/4.0);
17.
18.   return 0;
19.}
```

处理批量数据的尴尬

- 如果有1000名学生，每个学生有一个成绩，需要把这1000名学生成绩排序。
- 用$s_1,s_2,s_3,\ldots\ldots,s_{1000}$表示每个学生的成绩，能体现内在联系。那么定义float s1,s2,s3,……,s1000这1000个变量？
- 其实，定义float s[1000]就可以了。第i个学生的成绩可以用s[i-1]来表示。
- 注意：数组下标，一切从0开始！

利用数组“先存储再运算”

- 输入4个学生分数，并输出总分和平均分

```
1.  #include <stdio.h>
2.
3.  int main()
4.  {
5.     int stu[4],sum=0;

6.     printf("第1个"); scanf("%d", &stu[0]);
7.     printf("第2个"); scanf("%d", &stu[1]);
8.     printf("第3个"); scanf("%d", &stu[2]);
9.     printf("第4个"); scanf("%d", &stu[3]);
10.
11.    sum+=stu[0];      //6-9行，11-14行，语句
12.    sum+=stu[1];      //很相似，规律在哪？
13.    sum+=stu[2];
14.    sum+=stu[3];

15.    printf("总分: %d\n", sum);
16.    printf("平均分: %.2f\n", sum/4.0);
17.
18.    return 0;
19. }
```

```
1.  #include <stdio.h>
2.
3.  int main()
4.  {
5.     int i,stu[4], sum=0;
6.
7.     for (i=0; i<4; i++)
8.     {
9.        printf("第%d个:",i+1);
10.       scanf("%d", &stu[i]);
11.    }      //通常利用循环访问多个数组元素

12.    for (i=0;i<4; i++)
13.    {
14.       sum += stu[i];
15.    }

16.    printf("总分: %d\n", sum);
17.    printf("平均分: %.2f\n", sum/4.0);
18.
19.    return 0;
20. }
```

数组的基本概念

- 数组是一组有序数据的集合。数组中各数据的排列是有一定规律的，下标代表数据在数组中的序号
- 用一个数组名和下标惟一确定数组中的元素
- 数组中的每一个元素都属于同一个数据类型

现实中的数组/非数组

非数组：无序

非数组：不同类

数组：桃园三兄弟

数组：奶爸五人组

一维数组的概念

- 一维数组是数组中最简单的
- 它的元素只需要用数组名加一个下标，就能惟一确定
- 和普通变量一样，要使用数组，必须在程序中先定义

一维数组定义的形式

- 定义一维数组的一般形式为：

  类型符　数组名[常量表达式];

- 数组名的命名规则和变量名相同

如　int a[10];

一维数组定义的形式

- 定义一维数组的一般形式为：

  类型符　数组名[常量表达式];

- 数组名的命名规则和变量名相同

如　int a[10];

数组长度

一维数组定义的形式

- 定义一维数组的一般形式为：

  类型符 数组名[常量表达式];

- 数组名的命名规则和变量名相同

如 int a[10];

每个元素的数据类型

10个元素：a[0],a[1],a[2],...,a[9]

a[0]	a[1]	a[2]	a[3]	...	a[7]	a[8]	a[9]

一维数组定义的形式

- 定义一维数组的一般形式为：

  类型符 数组名[常量表达式];

- 例如
  - int a[100]; 合法
  - int a[4+6]; 合法
  - int n=10; int a[n]; 不合法

怎样引用一维数组元素

- 在定义数组并对其中各元素赋值后，就可以引用数组中的元素
- 注意：只能引用数组元素而不能一次整体调用整个数组全部元素的值

例如 int a[10];

a = 0;

不合法

引用数组元素的表示形式

- 引用数组元素的表示形式为：

数组名［下标］

如 int n=5,a[10];

a[0]=a[5]+a[7]-a[2*3] 合法

a[n]=20; 合法

更多合法的数组元素引用方式

- 引用的数组元素下标可以是数值也可以是表达式，例如

```
int i = 3;
printf("%d", a[2]);       //输出数组中第3个元素的值
printf("%d", a[i]);       //输出数组中第4个元素的值
printf("%d", a[i+2]);     //输出数组中第6个元素的值
printf("%d", a[i*2]);     //输出数组中第7个元素的值
printf("%d", a[i++]);     //输出数组中第4个元素，然后i=4
printf("%d", a[--i]);     //i=4-1=3，然后将a[3]输出
a[i] = a[i+1]+a[i+2];     //将a[4]+a[5]赋值给a[3]
```

数组引用简单举例

- 对10个数组元素依次赋值为0，1，2，3，4，5，6，7，8，9，要求按逆序输出。
- 解题思路：
  - 定义一个长度为10的数组，数组定义为整型
  - 要赋的值是从0到9，可以用循环来赋值
  - 用循环按下标从大到小输出这10个元素

数组引用简单举例

```
1.  #include <stdio.h>
2.  int main()
3.  {
4.      int i,a[10];
5.
6.      for (i=0; i<=9; i++)
7.          a[i] = i;
8.      for (i=9; i>=0; i--)
9.          printf("%d ",a[i]);
10.     printf("\n");
11.
12.     return 0;
13. }
```

使a[0]～a[9]的值为0～9

a[0]a[1]a[2]a[3]a[4]a[5]a[6]a[7]a[8]a[9]

0	1	2	3	4	5	6	7	8	9

数组引用简单举例

```
1.  #include <stdio.h>
2.  int main()
3.  {
4.      int i,a[10];
5.
6.      for (i=0; i<=9;i++)
7.          a[i] = i;
8.      for (i=9; i>=0; i--)
9.          printf("%d ",a[i]);
10.     printf("\n");
11.
12.     return 0;
13. }
```

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

先输出a[9]，最后输出a[0]

a[0]a[1]a[2]a[3]a[4]a[5]a[6]a[7]a[8]a[9]

0	1	2	3	4	5	6	7	8	9

数组访问的雷区：小心越界

- 在C语言中使用数组的一个**潜在危险**就是C并不提供对数组边界的检查。

```
1.  #include <stdio.h>
2.  int main()
3.  {
4.      int a[5]={1,2,3,4,5};
5.      int b[5]={6,7,8,9,10};
6.
7.      printf("a[5]=%d,b[5]=%d\n", a[5], b[5]);
8.
9.      return 0;
10. }
```

```
a[5]=1245064,b[5]=1
Press any key to continue
```

数组的初始化

- 在定义数组的同时，给各数组元素赋值
  - int a[10]={0,1,2,3,4,5,6,7,8,9};
  - int a[10]={0,1,2,3,4}; 相当于
    int a[10]={0,1,2,3,4,0,0,0,0,0};
  - int a[10]={0,0,0,0,0,0,0,0,0,0}; 相当于
    int a[10]={0};
  - int a[10]={5};相当于
    int a[10]={5,0,0,0,0,0,0,0,0,0};
  - int a[5]={1,2,3,4,5}; 可写为
    int a[ ]={1,2,3,4,5};

数组中未赋值元素的值

- 编译器在数组定义时就开辟了一定的存储空间，并给它们赋予了初值，但是这个初始值是莫名其妙的，不受我们控制

```
1.  #include <stdio.h>
2.  int main()
3.  {
4.      int a[10];
5.      printf("%d\n", a[5]);
6.      return 0;
7.  }
```

```
-858993460
Press any key to continue
```

- 所以，在使用数组之前最好对其进行初始化！

数组应用简单举例：数列

- 用数组处理求Fibonacci数列问题
- 解题思路：
  - 上一章中用简单变量处理，缺点是不能在内存中保存这些数。假如想直接输出数列中第25个数，是很困难的。
  - 如果用数组处理，每一个数组元素代表数列中的一个数，依次求出各数并存放在相应的数组元素中。

数组应用简单举例：数列

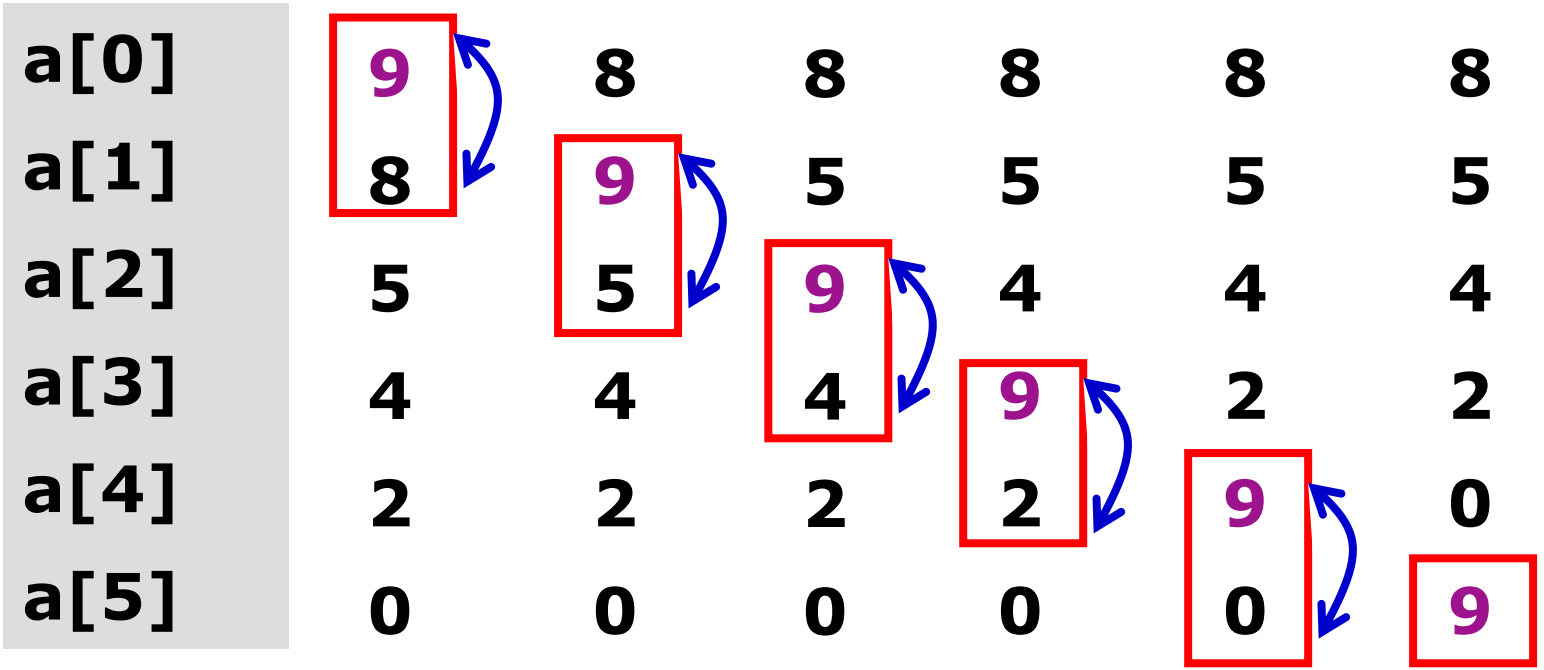
```
1. #include <stdio.h>
2. int main()
3. {
4.      int i;
5.      int f[20] = {1,1};
6.      for (i=2;i<20;i++)
7.          f[i] = f[i-2]+f[i-1];
8.      for (i=0;i<20;i++)
9.      {
10.         if (i%5==0)
11.             printf("\n");
12.         printf("%12d", f[i]);
13.     }
14.     printf("\n");
15.     return 0;
16.}
```

```
           1           1           2           3           5
           8          13          21          34          55
          89         144         233         377         610
         987        1597        2584        4181        6765
```

数组应用举例：排序

- 有6个人的代号，要求对它们按由小到大的顺序排列。
- 解题思路：
  - 排序的规律有两种：一种是“升序”，从小到大；另一种是“降序”，从大到小
  - 把题目抽象为：“对n个数按升序排序”
  - 采用“起泡法”排序

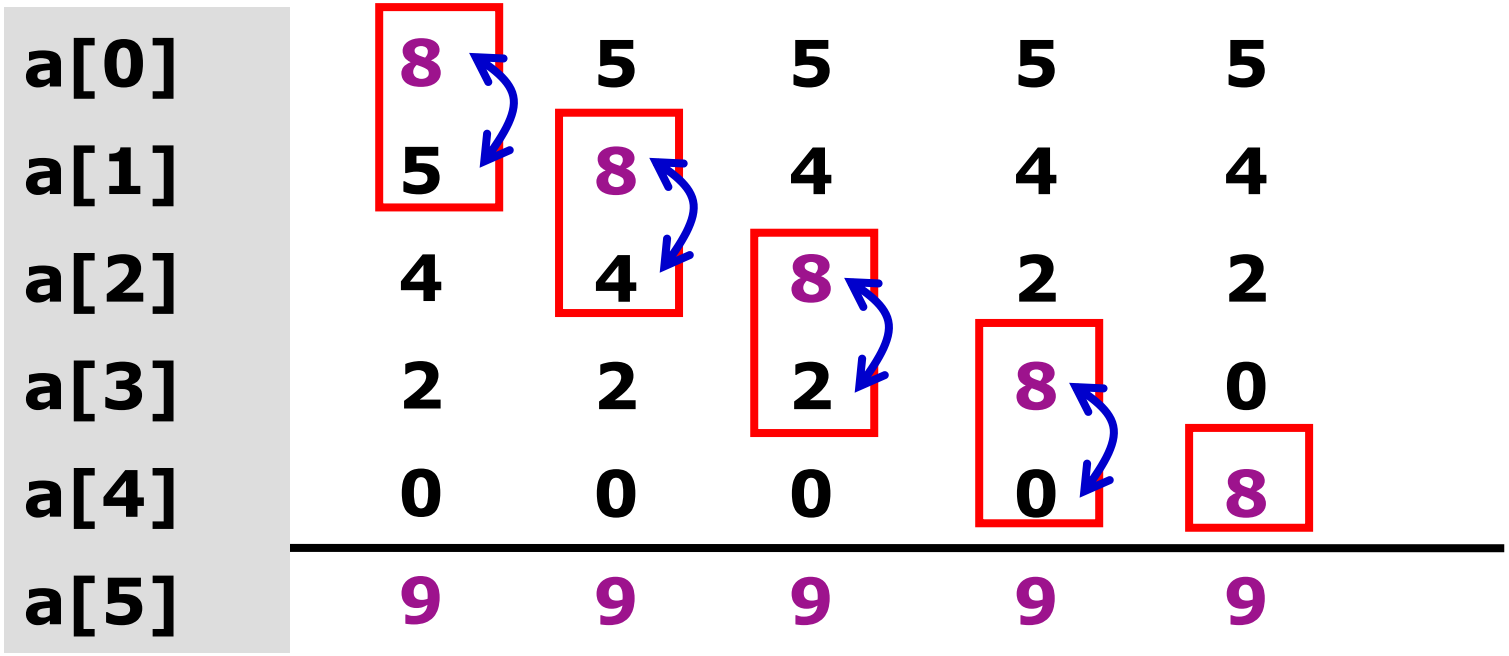
数组应用举例：排序




```
for (i=0;i<5;i++)
    if (a[i]>a[i+1])
    {
        t=a[i];a[i]=a[i+1];a[i+1]=t;
    }
```

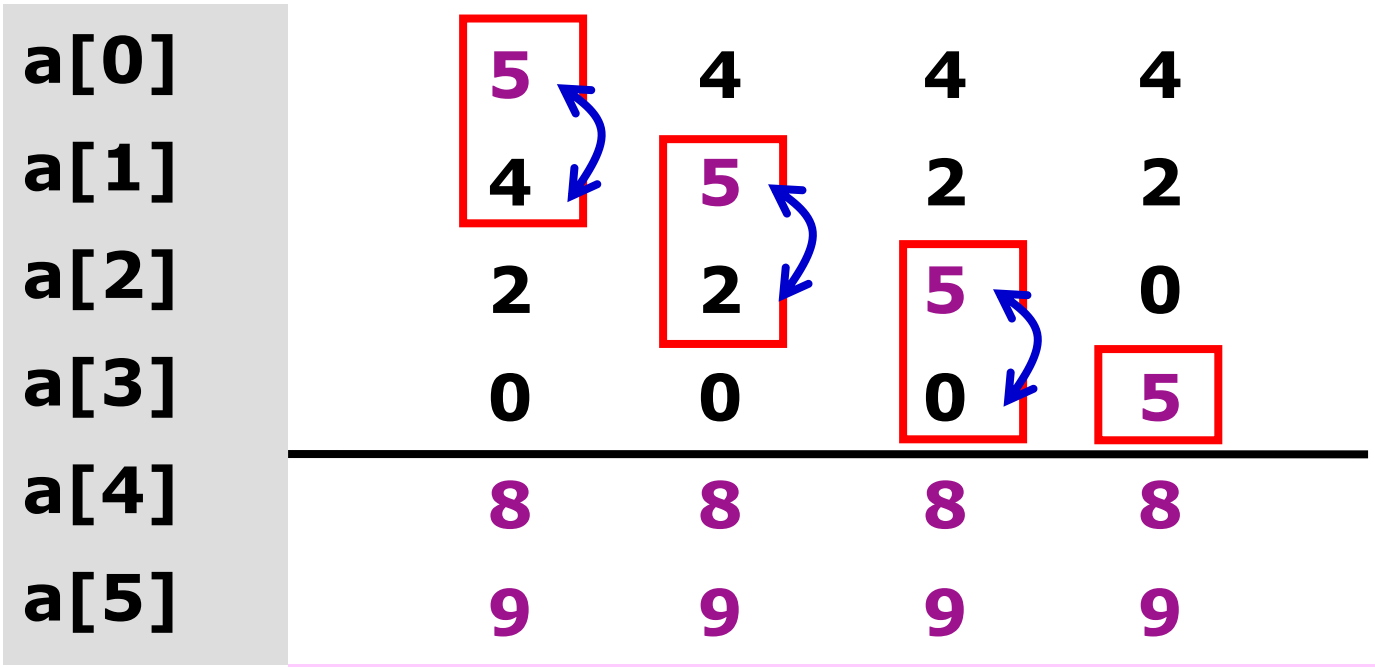
大数沉淀，小数起泡

数组应用举例：排序



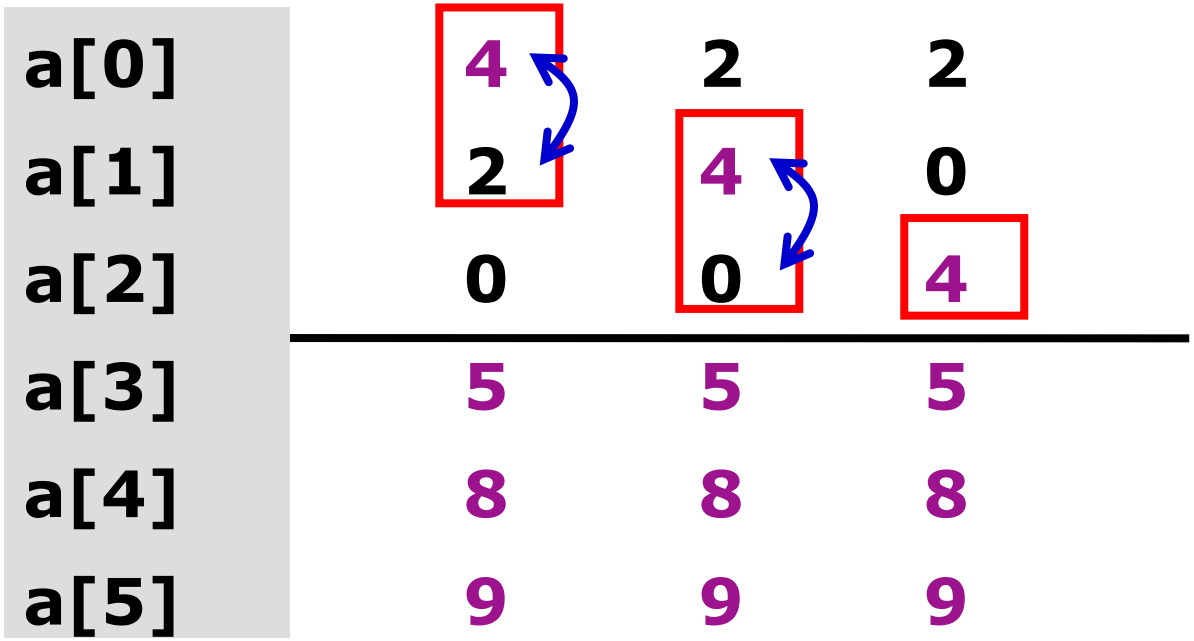

```
for (i=0;i<4;i++)
    if (a[i]>a[i+1])
    {
        t=a[i];a[i]=a[i+1];a[i+1]=t;
    }
```

数组应用举例：排序



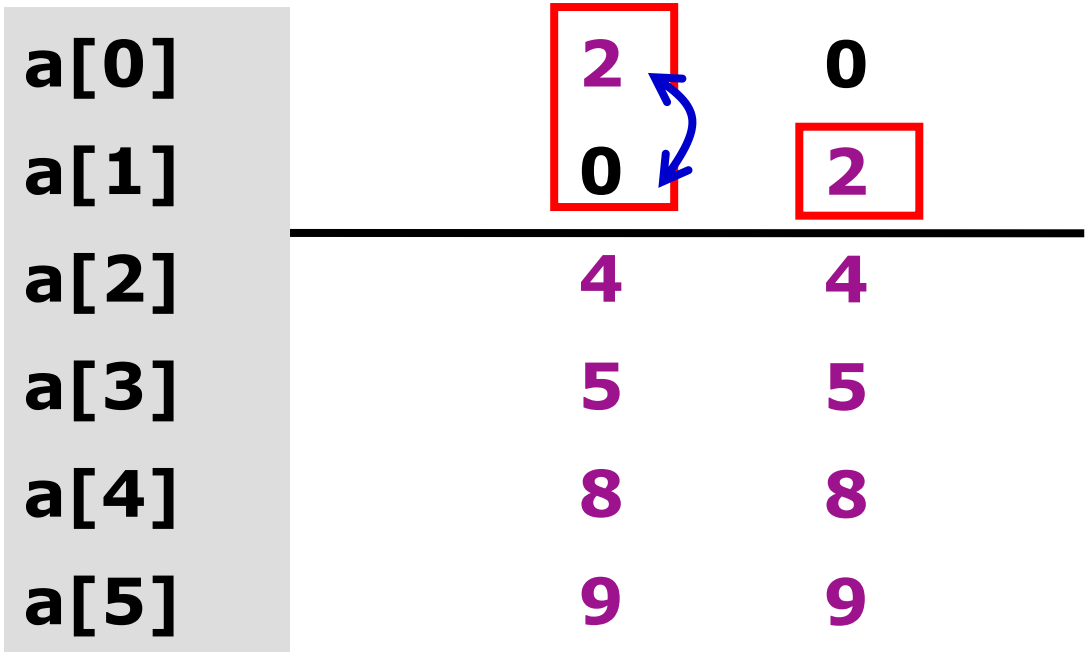

```
for (i=0;i<3;i++)
    if (a[i]>a[i+1])
    {
        t=a[i];a[i]=a[i+1];a[i+1]=t;
    }
```

数组应用举例：排序




```
for (i=0;i<2;i++)
    if (a[i]>a[i+1])
    {
        t=a[i];a[i]=a[i+1];a[i+1]=t;
    }
```

数组应用举例：排序




```
for (i=0;i<1;i++)
    if (a[i]>a[i+1])
    {
        t=a[i];a[i]=a[i+1];a[i+1]=t;
    }
```

数组应用举例：排序

```
for (i=0;i<5;i++)
    if (a[i]>a[i+1])
    { ……}
```

```
for (i=0;i<4;i++)
    if (a[i]>a[i+1])
    { ……}
```

……

```
for (i=0;i<1;i++)
    if (a[i]>a[i+1])
    { ……}
```

```
for (j=0;j<5;j++)
    for (i=0;i<5-j;i++)
        if (a[i]>a[i+1])
        { ……}
```

数组应用举例：排序

```
……
int a[10];    int i,j,t;
printf("input 10 numbers :\n");
for (i=0; i<10; i++)
    scanf("%d",&a[i]);
printf("\n");
for (j=0;j<9;j++)
    for (i=0;i<9-j;i++)
        if (a[i]>a[i+1])
            { t=a[i];a[i]=a[i+1];a[i+1]=t; }
printf("the sorted numbers :\n");
for (i=0; i<10; i++)
    printf("%d ",a[i]);
printf("\n");
……
```

```
input 10 numbers :
34 67 90 43 124 87 65 99 132 26

the sorted numbers :
26 34 43 65 67 87 90 99 124 132
```

二维数组举例

	队员1	队员2	队员3	队员4	队员5	队员6
1分队	2456	1847	1243	1600	2346	2757
2分队	3045	2018	1725	2020	2458	1436
3分队	1427	1175	1046	1976	1477	2018

```
float pay[3][6];
```

怎样定义二维数组

- 二维数组定义的一般形式为

  类型符 数组名[常量表达式][常量表达式];

  如：float a[3][4],b[5][10];

- 二维数组可被看作是一种特殊的一维数组：

  它的元素又是一个一维数组

作业 2017/11/10

- **按下列要求编写程序，提交手写源代码**

1. 读入用户输入的正整数，求其各位数字的和与乘积。

2. 编写一个程序，显示如右图案。

- **上机练习（不用交）：编译运行本讲义例程，教材第五章**

4、5、8、9、10