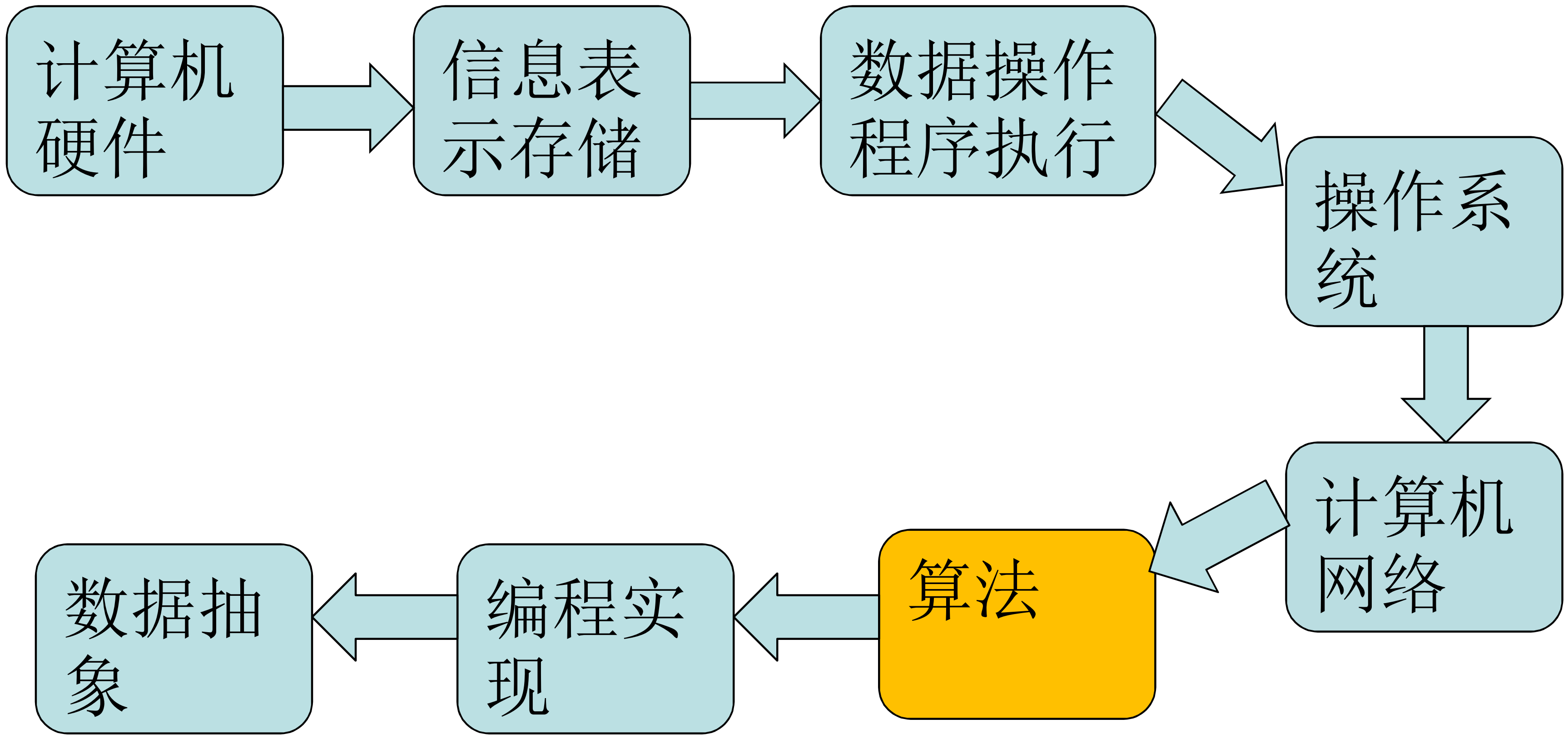
Chapter 5: Algorithms

Computer Science: An Overview

by
J. Glenn Brookshear

从程序的角度了解计算机系统



What have we learned?

- How does the computer / computer network work?

What are we going to learn?

- How to make computers work?

解决问题的方法

- 内功心法，菜谱，祖传秘方
- Algorithm(算法)

Chapter 5: Algorithm

- Basic concepts of algorithms
- Algorithm representation
- Some classical algorithm
  - Sequential search, binary search
  - Insertion sort
- Efficiency and correctness

Algorithm and program

- Algorithm
  - An **ordered** set of **unambiguous**, **executable** steps that defines a **terminating** process to solve the problem
- Program
  - A set of instructions, which describe how computers process data and solve the problem

Example: estimation of π

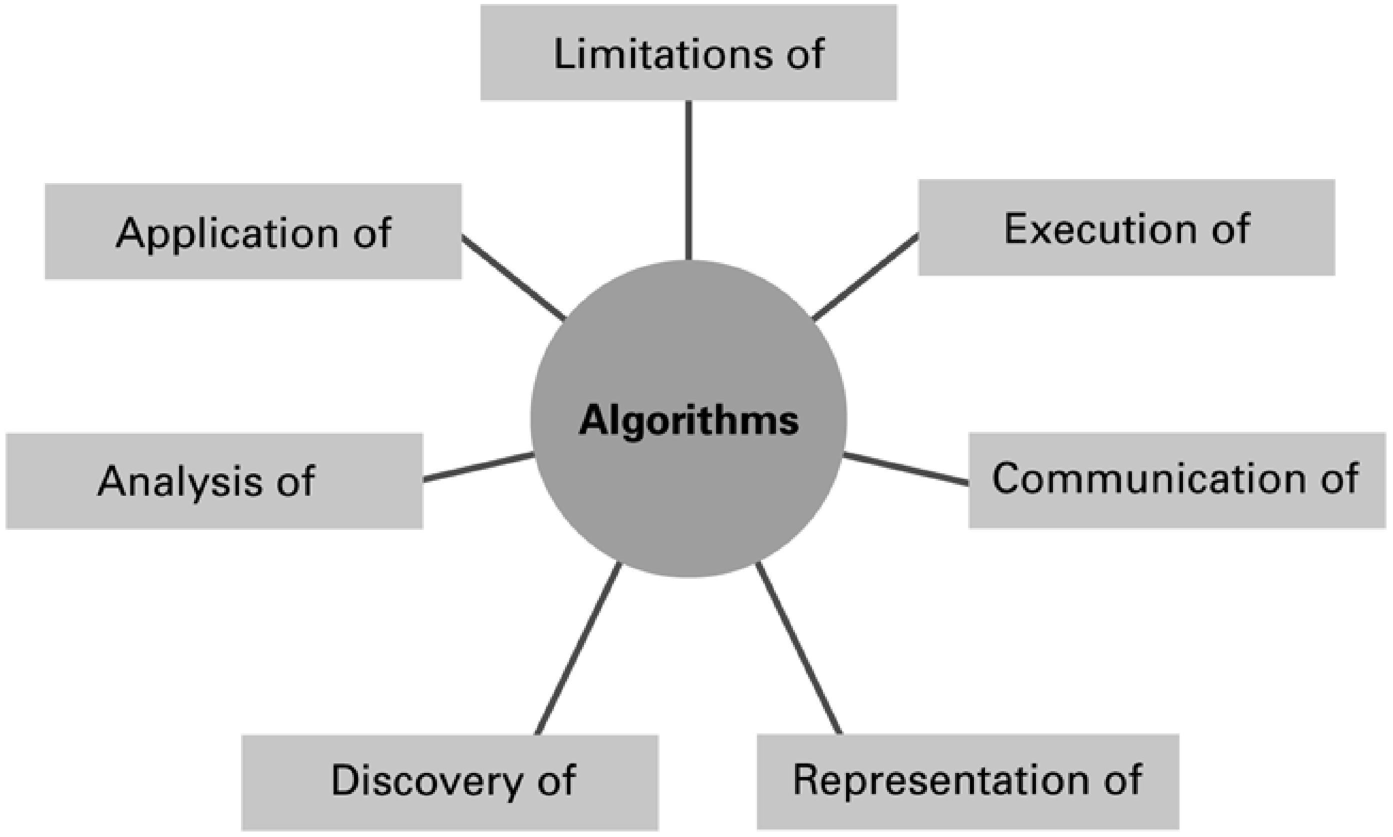
公式 1: $\frac{\pi}{2}=\frac{2^2}{1\times3}\times\frac{4^2}{3\times5}\times\frac{6^2}{5\times7}\times\frac{8^2}{7\times9}\times\cdots$

公式 2: $\frac{\pi}{4}=1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\frac{1}{9}-\frac{1}{11}+\cdots$

公式 3: $\frac{\pi}{6}=\frac{1}{\sqrt{3}}\times(1-\frac{1}{3\times3}+\frac{1}{3^2\times5}-\frac{1}{3^3\times7}+\cdots)$

- Different algorithms to solve the same problem
- Different efficiency
- Choose the one that can be easily understood and implemented

Central role of algorithms

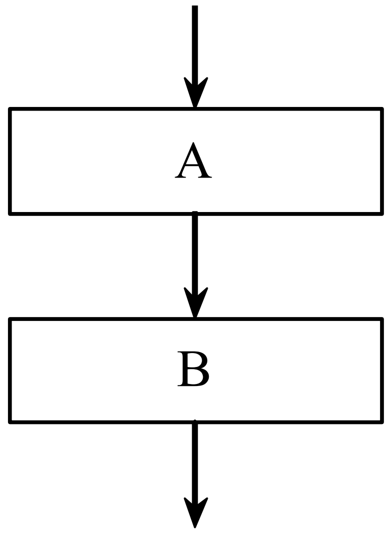


How to represent algorithms?

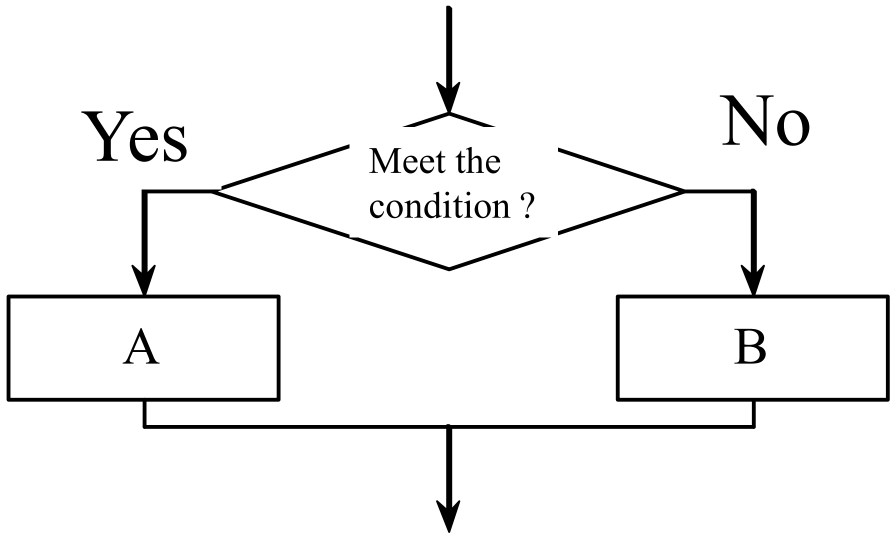
- **Algorithm: operation + control structure**
- **Operation:**
  - Arithmetic: +, -, *, / , etc.
  - Relation: >=, <=, etc.
  - Logic: and, or, not, etc.
  - Data transfer: load, store

Control structure

- Sequential
- Conditional
- loop

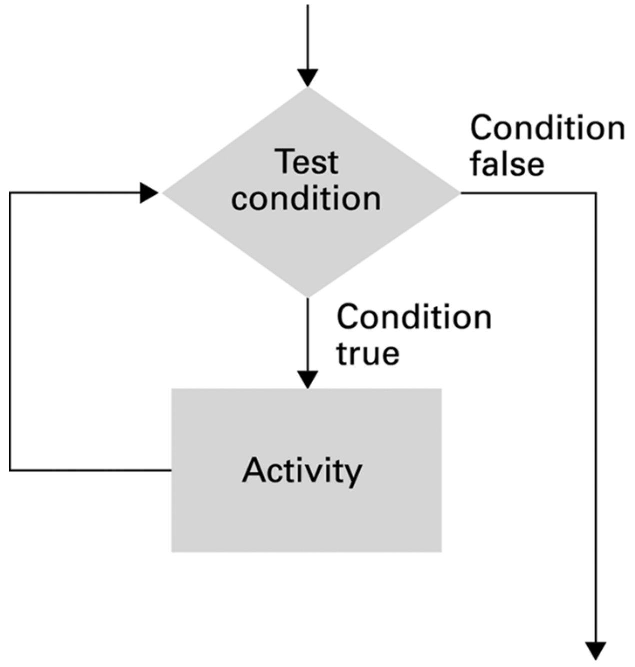


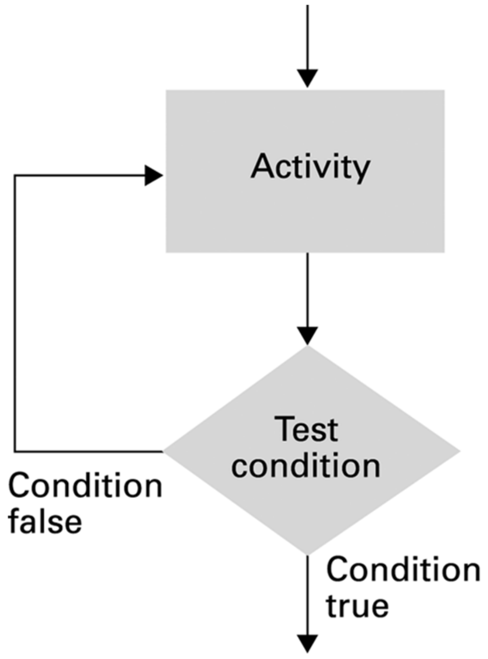
Sequential



Conditional

Loop structure





While loop structure

Repeat loop structure

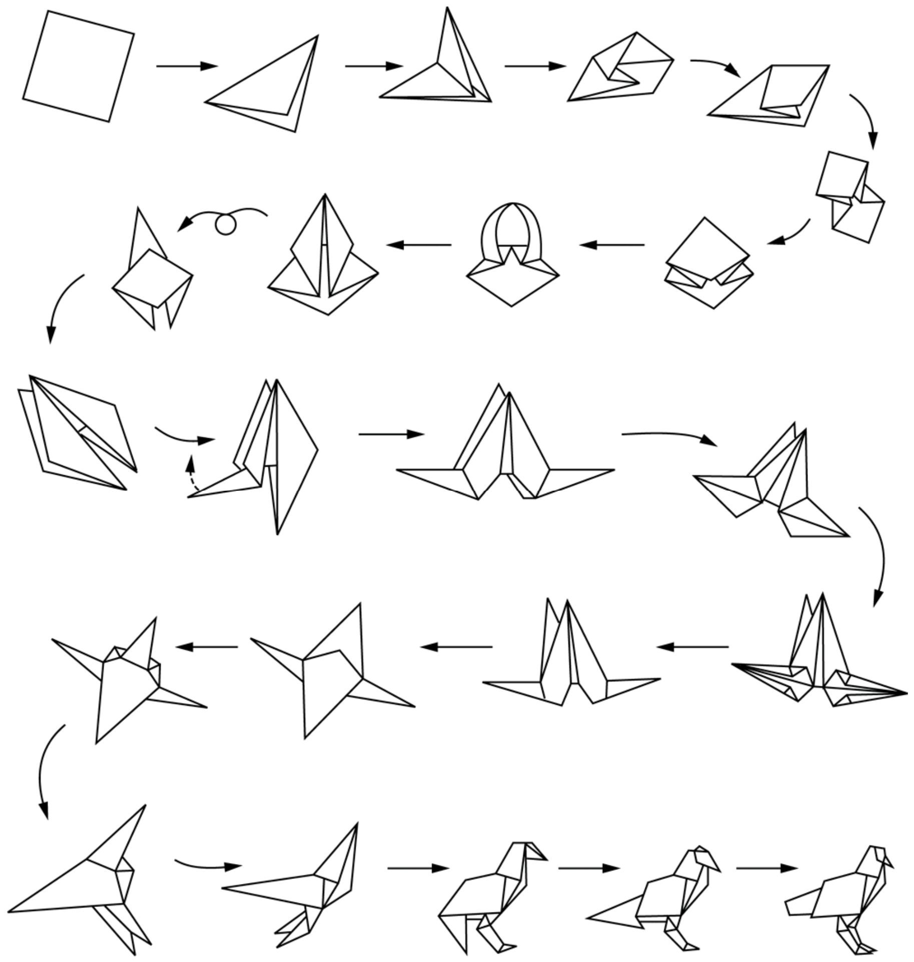
Algorithm Representation

- Requires well-defined primitives(原语)
- A collection of primitives constitutes a programming language.

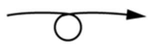
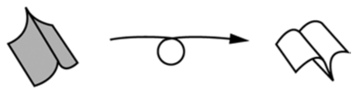
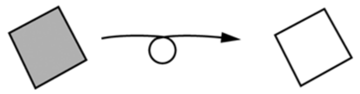
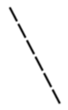
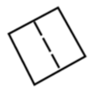
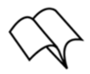
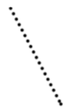
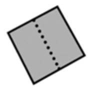
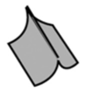
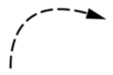
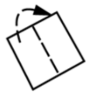
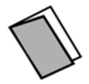
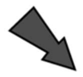
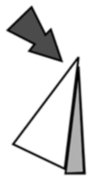
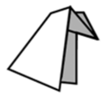
Algorithm representation

- Natural language
- Traditional flow chart 流程图(visio)
- N-S flow chart
- Pseudocode伪代码

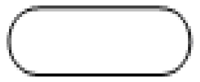
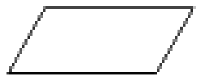
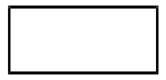
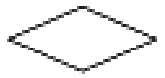
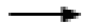
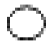
Figure 5.2 **Folding a bird from a square piece of paper**



Natural language: Origami primitives
折纸原语

Syntax	Semantics
	Turn paper over as in 
Shade one side of paper	Distinguishes between different sides of paper as in 
	Represents a valley fold so that  represents 
	Represents a mountain fold so that  represents 
	Fold over so that  produces 
	Push in so that  produces 

Flow chart

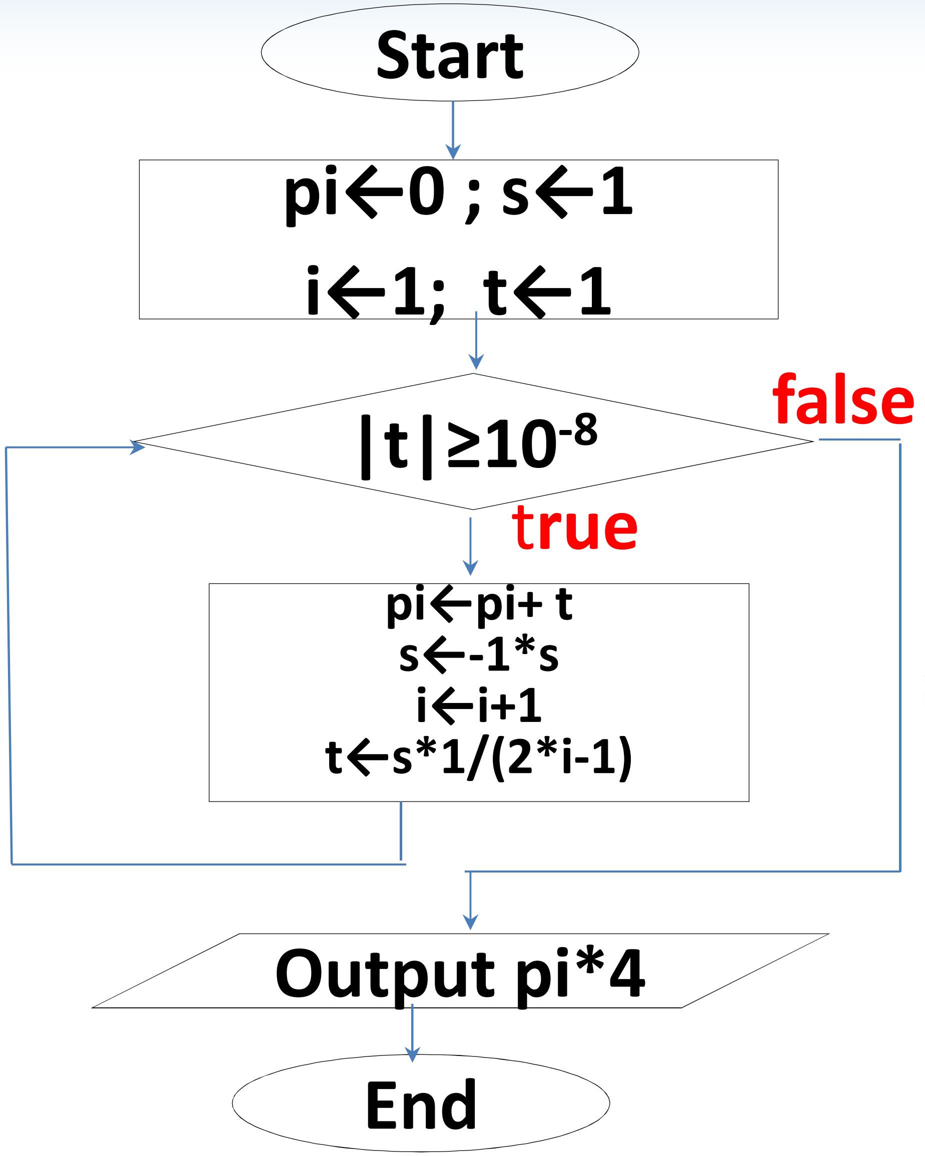
符号名称	图形	功能
起止框		表示算法的开始和结束
输入/输出框		表示算法的输入/输出操作
处理框		表示算法中的各种处理操作
判断框		表示算法中的条件判断操作
流程线		表示算法的执行方向
连接点		表示流程 图的延续

Computation of Pi

公式 1: $\frac{\pi}{2}=\frac{2^2}{1\times3}\times\frac{4^2}{3\times5}\times\frac{6^2}{5\times7}\times\frac{8^2}{7\times9}\times\cdots$

公式 2: $\frac{\pi}{4}=1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\frac{1}{9}-\frac{1}{11}+\cdots$

公式 3: $\frac{\pi}{6}=\frac{1}{\sqrt{3}}\times(1-\frac{1}{3\times3}+\frac{1}{3^2\times5}-\frac{1}{3^3\times7}+\cdots)$

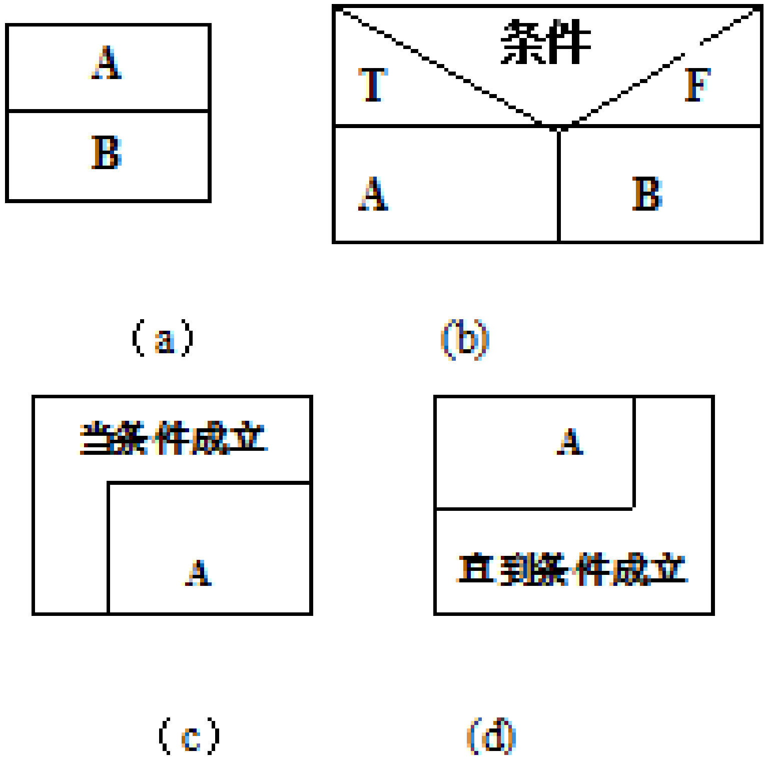


公式 2: $\frac{\pi}{4}=1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\frac{1}{9}-\frac{1}{11}+\cdots$

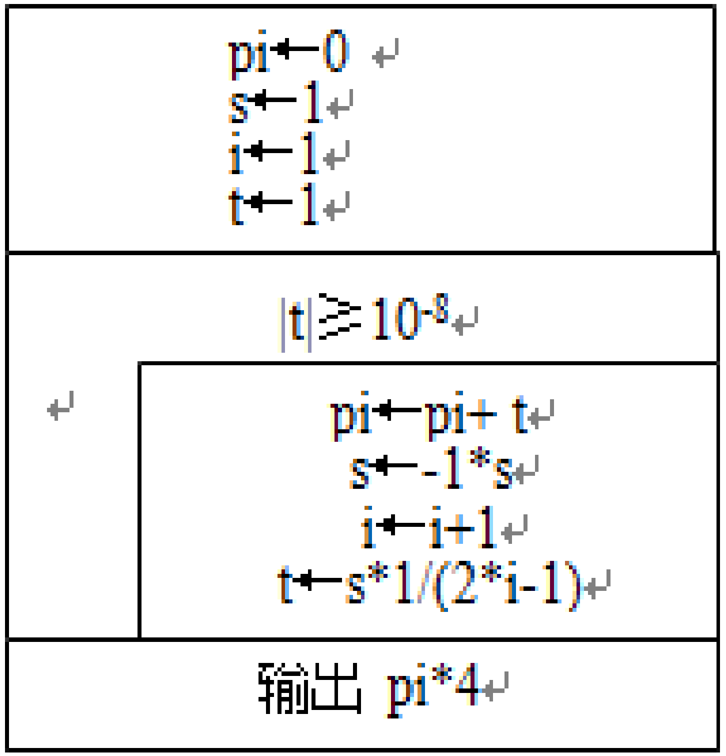
N-S flow chart

- **Proposed by I.Nassi and B.Shneideman**
- **For** structured programming

结构化程序设计（structured programming）的主要观点是采用自顶向下、逐步求精及模块化的程序设计方法；使用三种基本控制结构构造程序，任何程序都可由顺序、选择、循环三种基本控制结构构造。结构化程序设计主要强调的是程序的易读性。



▪



3 control structures for N-S flow chart

Pseudocode Primitives

- Assignment

  name ← *expression*

- Conditional selection

  if *condition* **then** *action*

Pseudocode Primitives (continued)

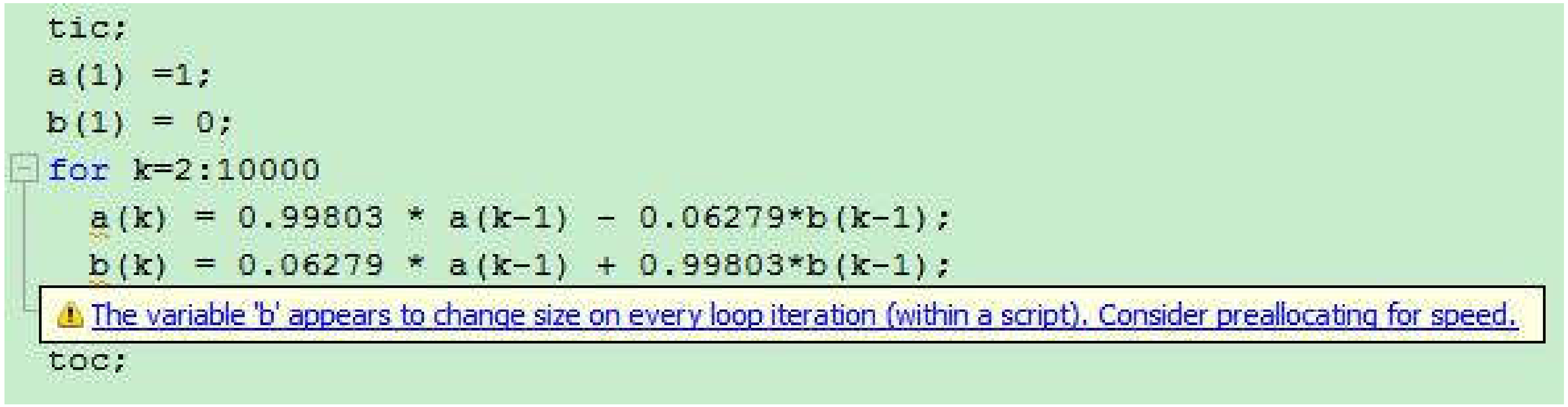
- Repeated execution

```
while condition do activity
```

- Procedure过程

```
procedure name (generic names)
```

Real code?



```
tic;
a(1) =1;
b(1) = 0;
for k=2:10000
    a(k) = 0.99803 * a(k-1) - 0.06279*b(k-1);
    b(k) = 0.06279 * a(k-1) + 0.99803*b(k-1);
```
⚠ The variable 'b' appears to change size on every loop iteration (within a script). Consider preallocating for speed.
```
toc;
```

What can algorithms do?

- Autonomous vehicle
- Robots
- Gene sequencing
- Climate prediction

Algorithm discovery

- Art

  = analysis + knowledge + experiment + inspiration (potential)

Polya's Problem Solving Steps

- 1. Understand the problem.
- 2. Devise a plan for solving the problem.
- 3. Carry out the plan.
- 4. Evaluate the solution for accuracy and its potential as a tool for solving other problems.

How to find algorithms?

- Try working the problem backwards
- Solve an easier related problem
  - Relax some of the problem constraints
  - Solve pieces of the problem first (bottom up methodology)
- Stepwise refinement: Divide the problem into smaller problems (top-down methodology)

Ages of Children Problem

- Person A is charged with the task of determining the ages of B’s three children.
  - B tells A that the product of the children’s ages is 36.
  - A replies that another clue is required.
  - B tells A the sum of the children’s ages.
  - A replies that another clue is needed.
  - B tells A that the oldest child plays the piano.
  - A tells B the ages of the three children.
- How old are the three children?

Figure 5.5

a. Triples whose product is 36

(1,1,36)	(1,6,6)
(1,2,18)	(2,2,9)
(1,3,12)	(2,3,6)
(1,4,9)	(3,3,4)

b. Sums of triples from part (a)

1 + 1 + 36 = 38	1 + 6 + 6 = 13
1 + 2 + 18 = 21	2 + 2 + 9 = 13
1 + 3 + 12 = 16	2 + 3 + 6 = 11
1 + 4 + 9 = 14	3 + 3 + 4 = 10

Learning Algorithms

- Every class in CS has some relations with algorithms
  - But some classes are not obviously related to algorithms, such as math classes.
  - 微积分, 离散数学, 线性代数, 概率统计, …
- Why should we study them?
  - They are difficult, and boring, and difficult…

Why Study Math?

- Train the logical thinking and reasoning
  - Algorithm is a result of logical thinking: correctness, efficiency, …
  - Good programming needs logical thinking and reasoning. For example, debugging.
- Learn some basic tricks
  - Many beautiful properties of problems can be revealed through the study of math
  - Standing on the shoulders of giants

Try to Solve These Problems

1. Given a network of thousands nodes, find the shortest path from node A to node B
   – The shortest path problem (离散数学)
2. Given a circuit of million transistors, find out the current on each wire
   – Kirchhoff's current law (线性代数)
3. In CSMA/CD, what is the probability of collisions?
   – Network analysis (概率统计)

微积分

- Limits, derivatives, and integrals of continuous functions.
- Used inalmost every field
  - 网络分析, 效能分析
  - 信号处理, 图像处理, 模拟电路设计
  - 科学计算， 人工智能, 计算机视觉, 计算机图学
  - …
- Also the foundation of many other math
  - 概率统计, 工程数学

离散数学

- Discrete structure, graph, integer, logic, abstract algebra, combinatorics
- The foundation of computer science
  - Every field in computer science needs it
  - Particularly, 算法, 数字逻辑设计, 密码学, 编码理论, 计算机网络, CAD, 计算理论
- Extended courses
  - Special topics on discrete structure, graph theory, concrete math

线性代数

- Vectors, matrices, tensors, vector spaces, linear transformation, system of equations
- Used in almost everywhere when dealing with more than one number
  - 网络分析, 效能分析
  - 信号处理, 图像处理
  - 科学计算， 人工智能, 计算机视觉, 计算机图学
  - CAD,电路设计

概率统计

- Probability models, random variables, probability functions, stochastic processes
- Every field uses it
  - Particularly,网络分析, 效能分析, 信号处理, 图像处理, 科学计算， 人工智能, 计算机视觉…
  - Other examples: randomized algorithm, queue theory, computational finance, performance analysis

工程数学

- A condensed course containing essential math tools for most engineering disciplines
  - Partial differential equations, Fourier analysis, Vector calculus and analysis
- Used in the fields that need to handle continuous functions
  - 网络分析, 效能分析, 信号处理, 图像处理, 科学计算, 人工智能, 计算机视觉, CAD,电路设计

Algorithm representation

- Pseudocode伪代码

Iterative Structures循环结构

- Pretest loop:

```
while (condition) do
    (loop body)
```

- Posttest loop:

```
repeat (loop body)
    until(condition)
```

Figure 5.8 **The while loop structure**

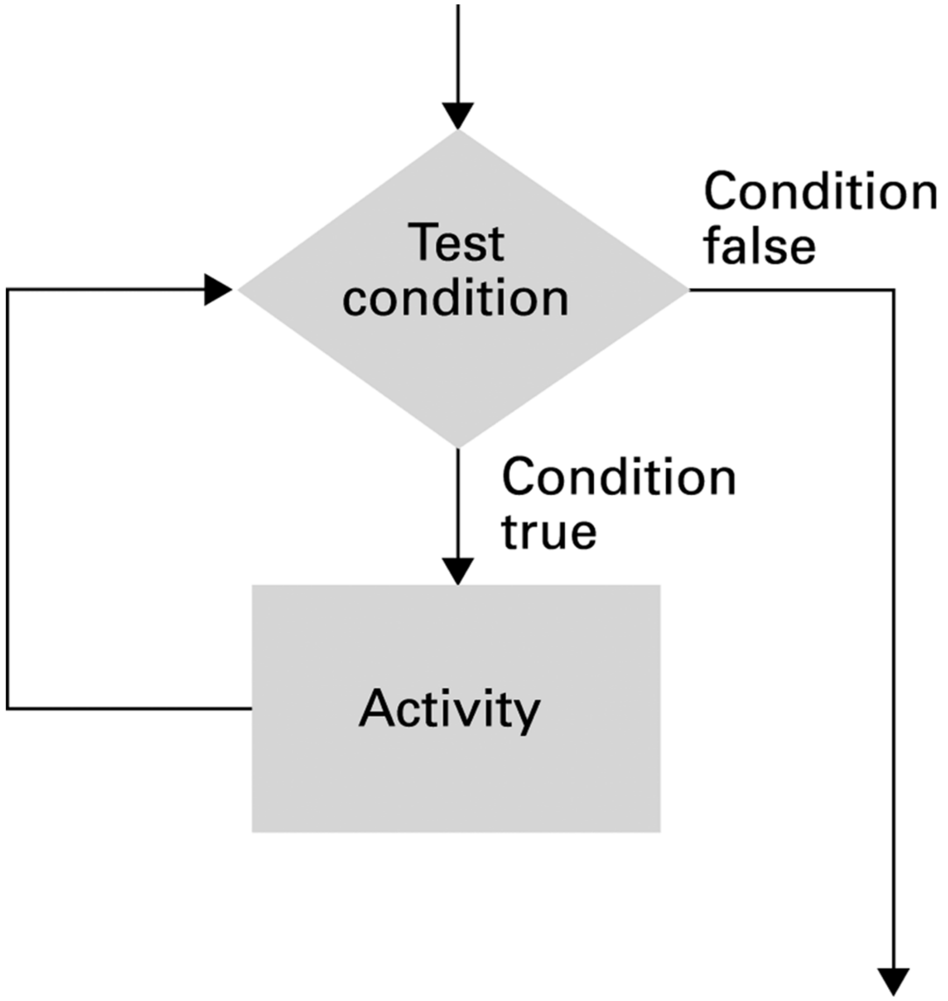


Figure 5.9 **The repeat loop structure**

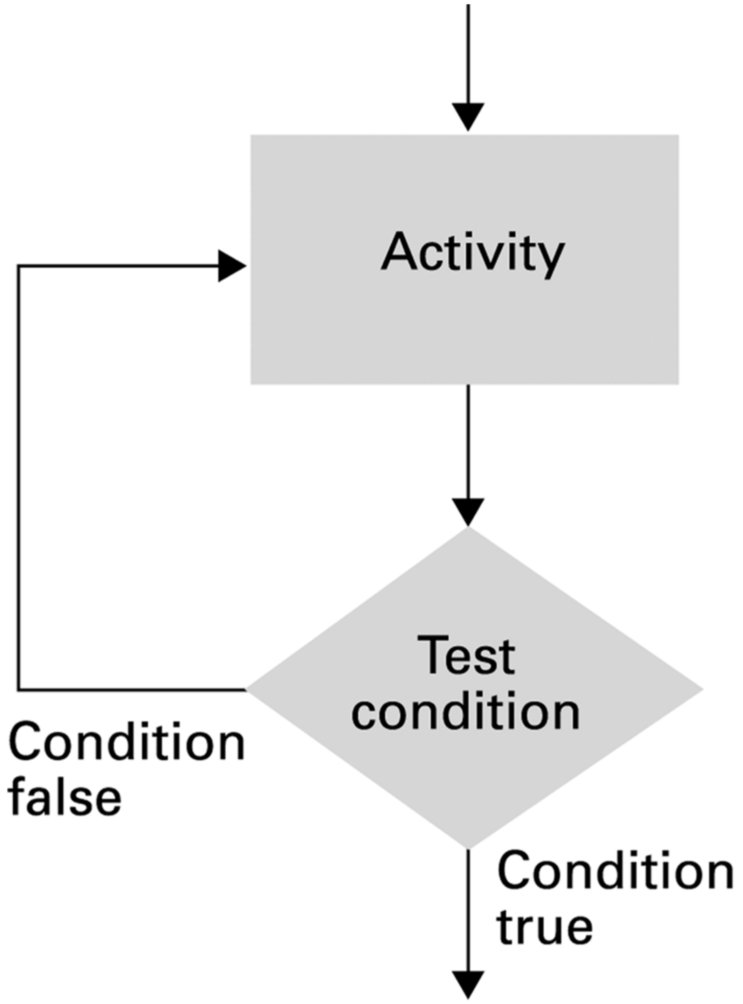


Figure 5.4 **The procedure Greetings in pseudocode**

```
procedure Greetings
Count ← 3;
while (Count > 0) do
    (print the message "Hello" and
        Count ← Count –1)
```

Recursion递归

- The execution of a procedure leads to another execution of the procedure.
- Multiple activations of the procedure are formed, all but one of which are waiting for other activations to complete.

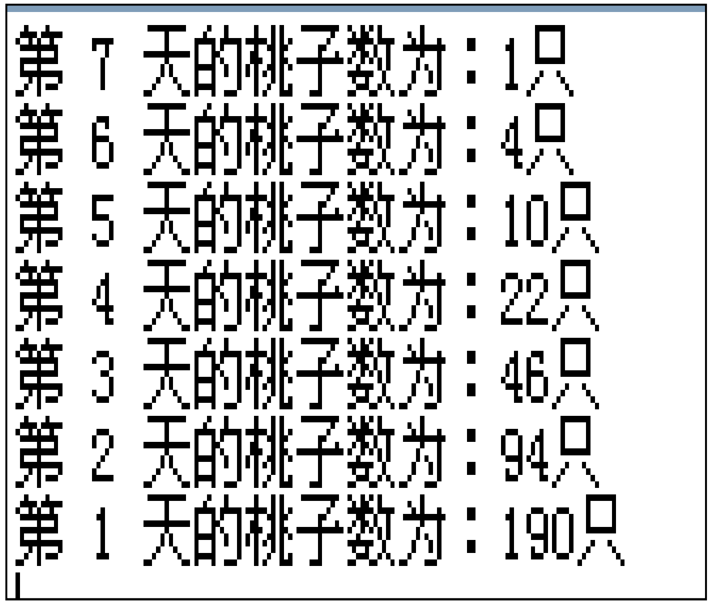
Recursion: example

- **例：猴子吃桃问题**

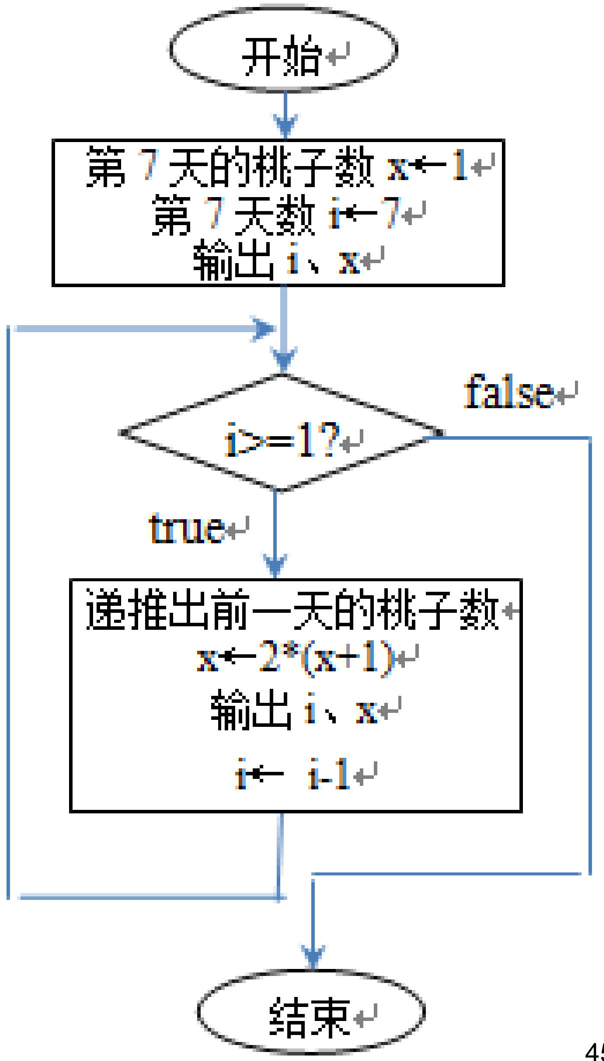
  小猴有桃若干，当天吃掉一半多一个；第二天接着吃了剩下的桃子的一半多一个；以后每天都吃尚存桃子的一半零一个，到第7天早上只剩下1个了，问小猴原有多少个桃子？

- 设第*n*天的桃子为x_n，它是前一天的桃子数的一半少1个，即

$x_{n-1}=(x_n+1)\times 2$ （递推公式）



```
第 7 天的桃子数为：1只
第 6 天的桃子数为：4只
第 5 天的桃子数为：10只
第 4 天的桃子数为：22只
第 3 天的桃子数为：46只
第 2 天的桃子数为：94只
第 1 天的桃子数为：190只
```



Chain Separating Problem

- A traveler has a gold chain of seven links.
- He must stay at an isolated hotel for seven nights.
- The rent each night consists of one link from the chain.
- What is the fewest number of links that must be cut so that the traveler can pay the hotel one link of the chain each morning without paying for lodging in advance?

Figure 5.21 **Separating the chain using only three cuts**

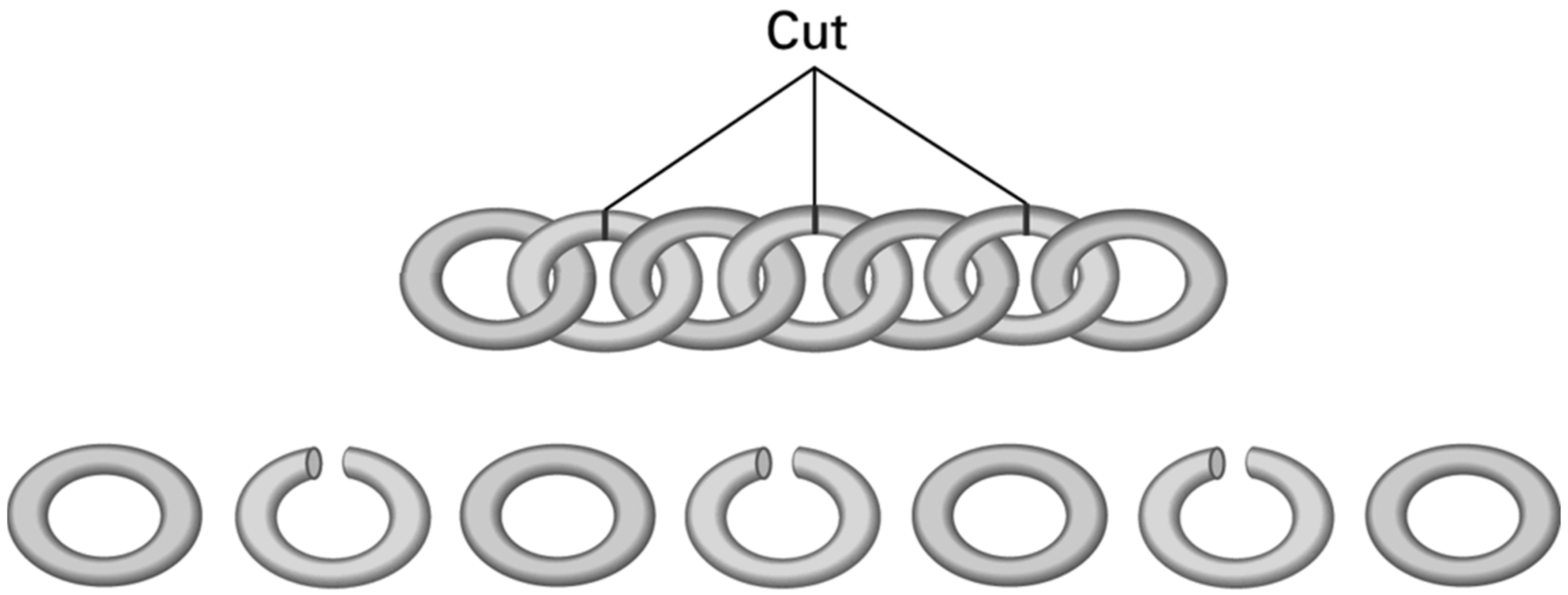
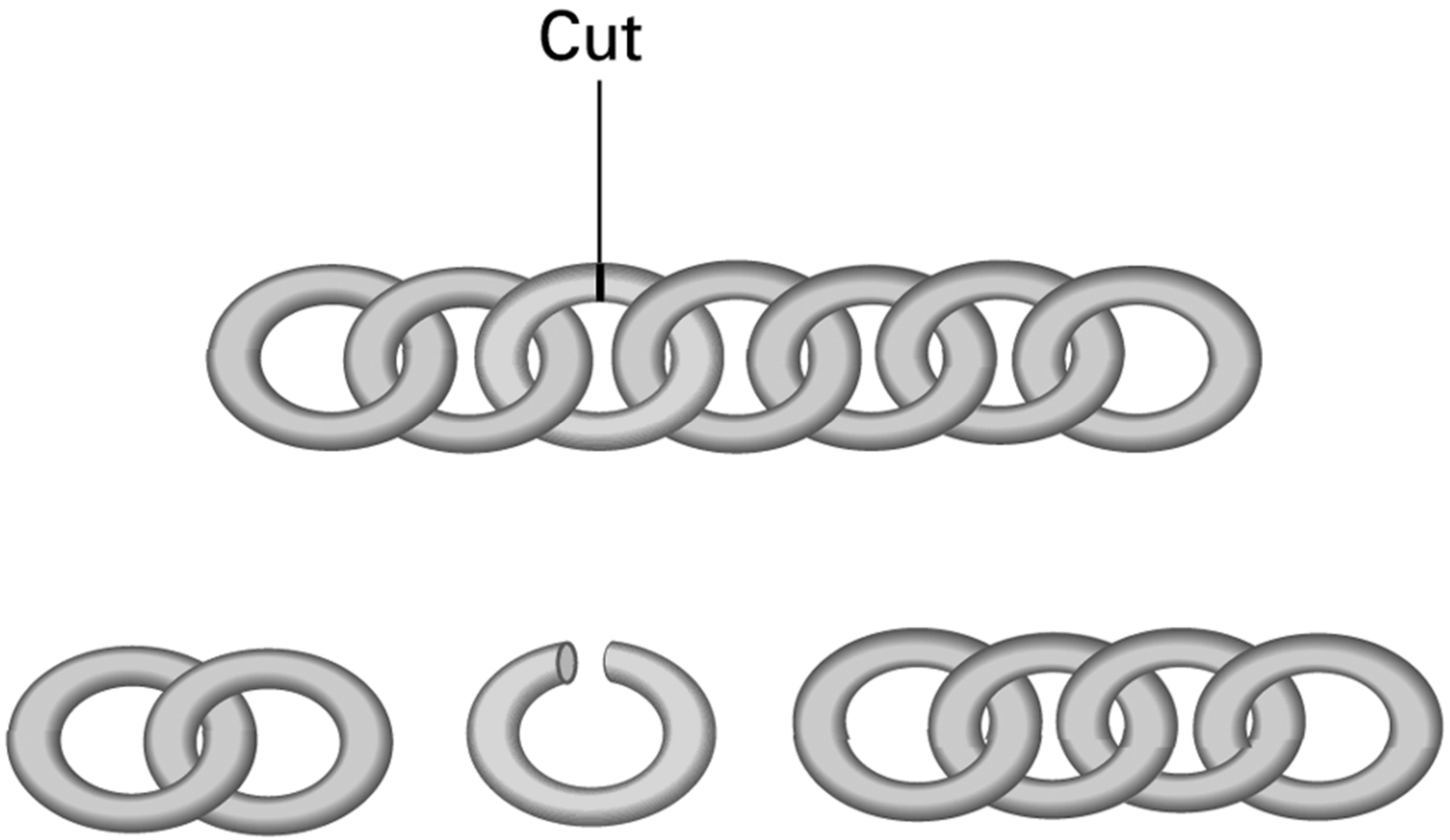


Figure 5.22 **Solving the problem with only one cut**



Classic algorithms

- Sort排序
- Search查找

Sort algorithm

Video

- Insertion sort (插入排序)
- Selection sort (选择排序)
- Bubble sort (冒泡排序)
- Quick sort (快速排序)

3、插入排序算法:

基本思想:

经过i-1遍处理后, a[1],a[2],……,a[i-1]已排好序。

第i遍处理是将元素a[i]插入到a[1],a[2],……,a[i-1]的适当的位置, 从而使得a[1],a[2],……,a[i-1],a[i]又是排好的序列。

排序的关键字: 从小到大

20 30 10 15 16 13 8

基本思想:
每次将一个待排序的数据元素, 插入到前面已经排好序的数列中的适当位置, 使数列依然有序; 直到待排序数据元素全部插入完为止。

Insertion sort

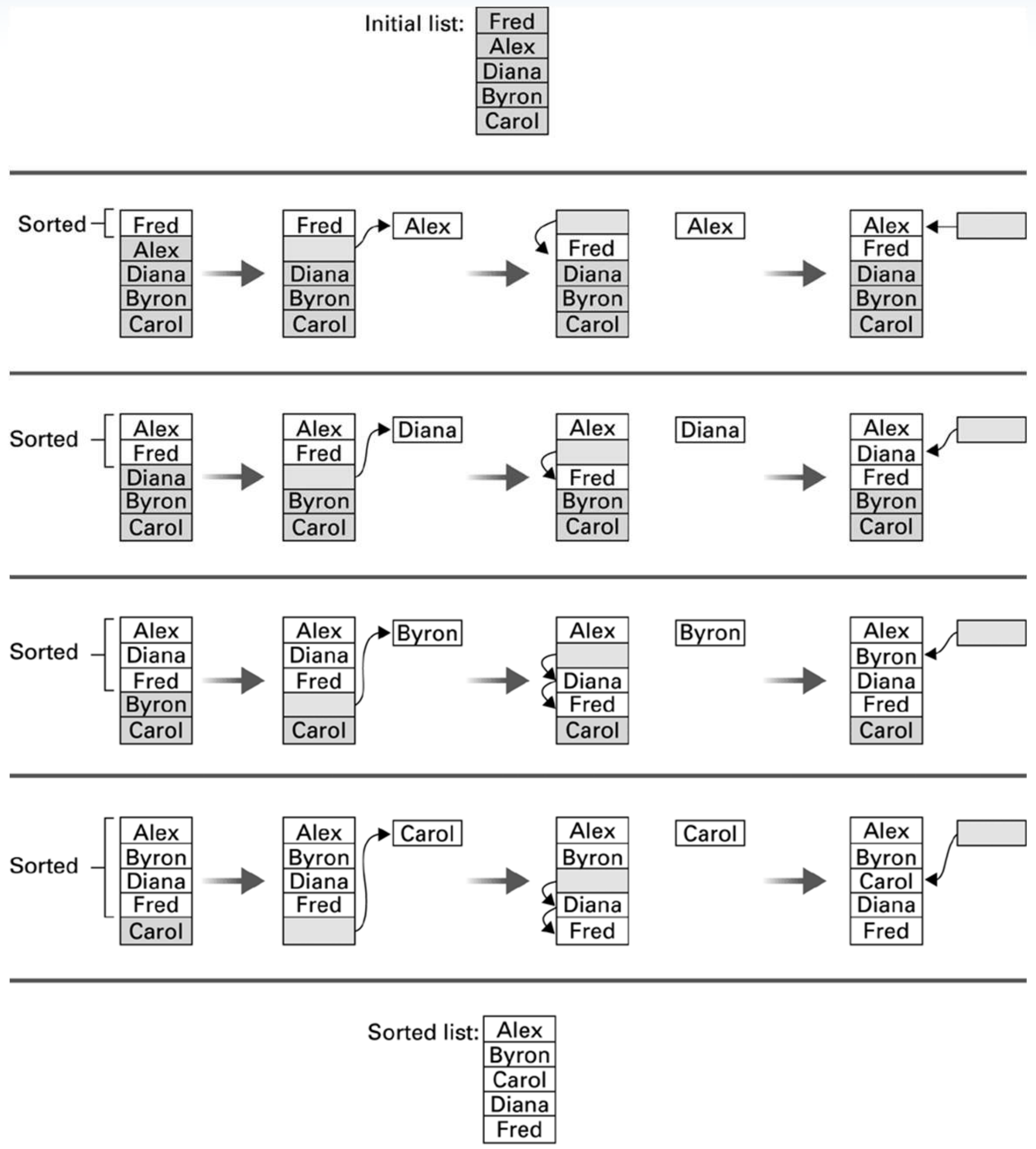
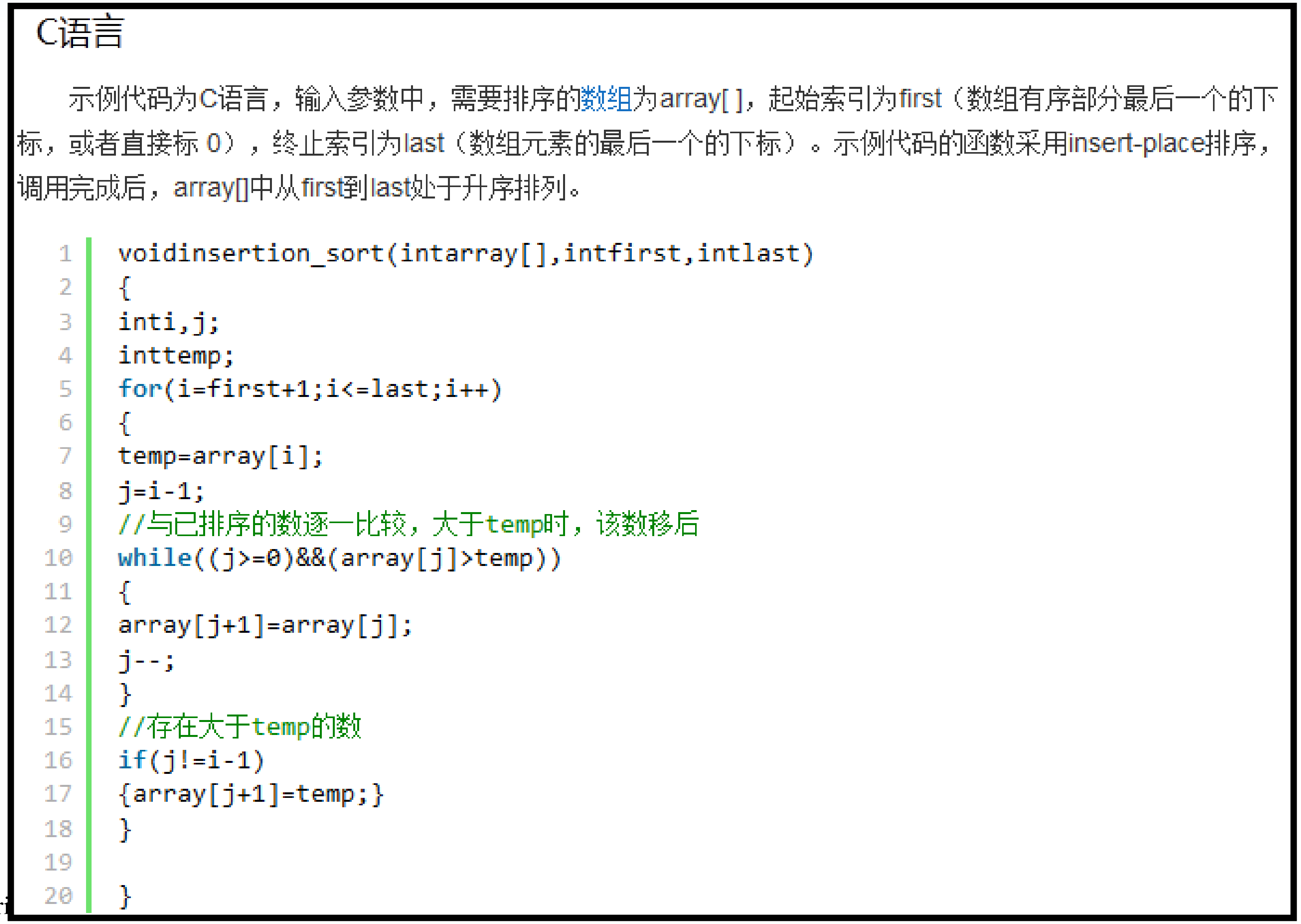


Figure 5.11 **The insertion sort algorithm expressed in pseudocode**

```
procedure Sort (List)
N ← 2;
while (the value of N does not exceed the length of List) do
    (Select the Nth entry in List as the pivot entry;
     Move the pivot entry to a temporary location leaving a hole in List;
      while (there is a name above the hole and that name is greater than the pivot) do
          (move the name above the hole down into the hole leaving a hole above the name)
      Move the pivot entry into the hole in List;
      N ← N + 1
    )
```

插入排序



C语言

示例代码为C语言，输入参数中，需要排序的数组为array[ ]，起始索引为first（数组有序部分最后一个的下标，或者直接标 0），终止索引为last（数组元素的最后一个的下标）。示例代码的函数采用insert-place排序，调用完成后，array[]中从first到last处于升序排列。

```
voidinsertion_sort(intarray[],intfirst,intlast)
{
inti,j;
inttemp;
for(i=first+1;i<=last;i++)
{
temp=array[i];
j=i-1;
//与已排序的数逐一比较，大于temp时，该数移后
while((j>=0)&&(array[j]>temp))
{
array[j+1]=array[j];
j--;
}
//存在大于temp的数
if(j!=i-1)
{array[j+1]=temp;}
}

}
```

1、选择排序

算法基本思想：

对待排序的序列进行**n-1**遍处理：

第1遍处理是从a[1],a[2],……a[n]中选择最小的放在a[1]位置；

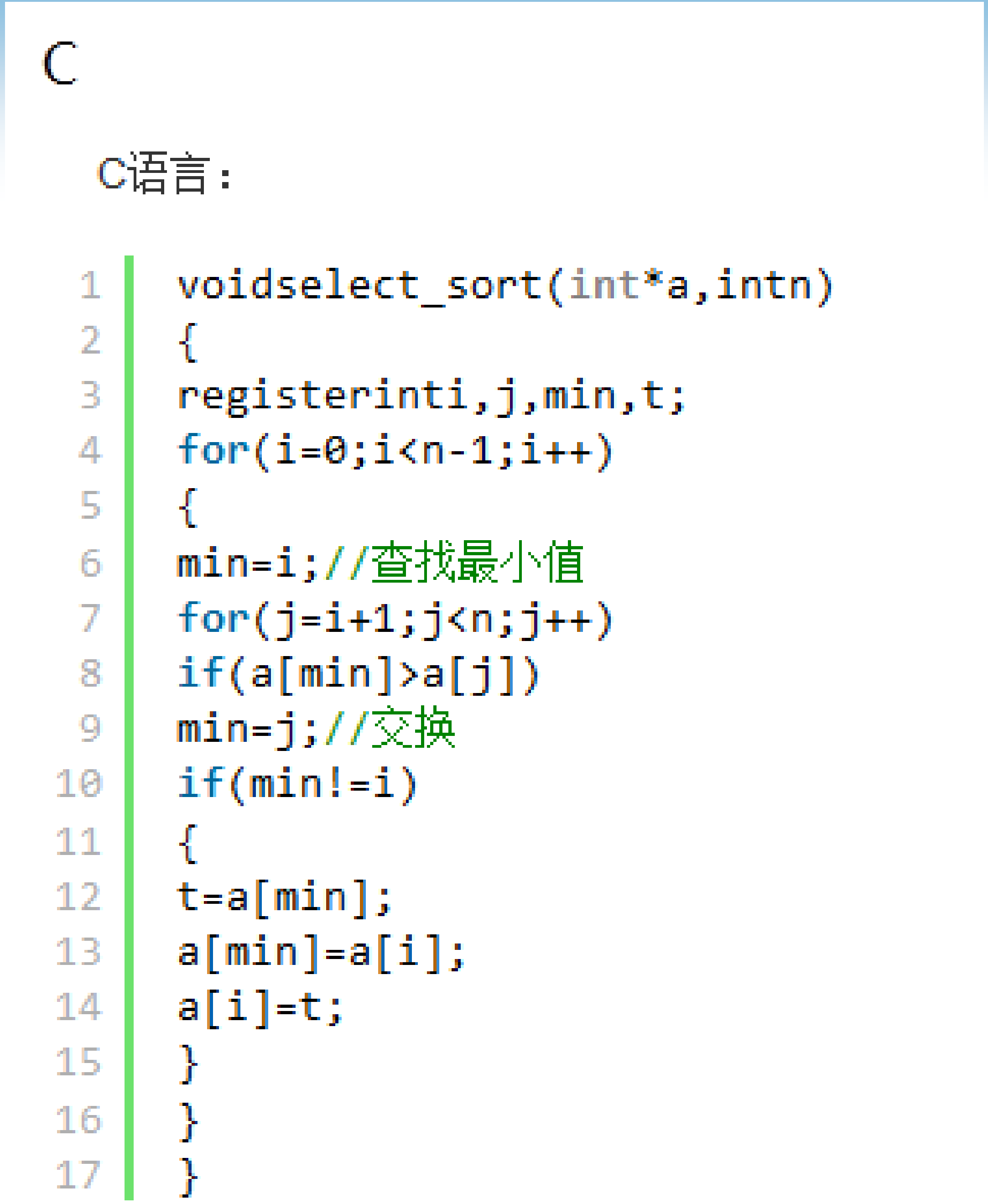
第2遍处理是从a[2],a[3],……a[n]中选择最小的放在a[2]位置；

……

第I遍处理是将a[i],a[i+1],……a[n]中最小的数与a[i]交换位置，这样经过第i遍处理后，a[i]是所有的中的第i小。即前i个数就已经排好序了。

N-1遍处理后，剩下的最后一个一定是最大的，不需要再处理了。

选择排序



C

C语言：

```
voidselect_sort(int*a,intn)
{
registerinti,j,min,t;
for(i=0;i<n-1;i++)
{
min=i;//查找最小值
for(j=i+1;j<n;j++)
if(a[min]>a[j])
min=j;//交换
if(min!=i)
{
t=a[min];
a[min]=a[i];
a[i]=t;
}
}
}
```

2、冒泡排序算法：

基本思想：（从小到大排序）

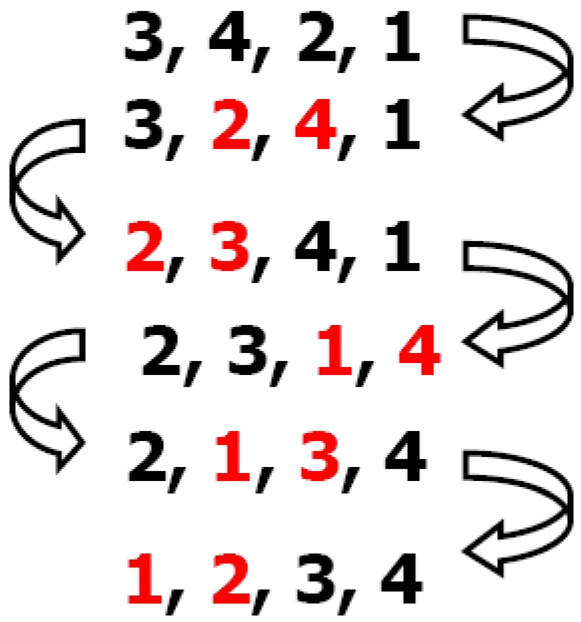
将待排序的数据看作竖派排的一列”气泡“，小的数据比较轻，从而要上浮。

共进行**n-1**遍处理，每一遍处理，就是从底向上检查序列，如果相邻的两个数据顺序不对，即轻（小）的在下面，就交换他们的位置。

第一遍处理完后，“最轻”的就浮到上面。

第二遍处理完后，“次轻”的就浮到上面。

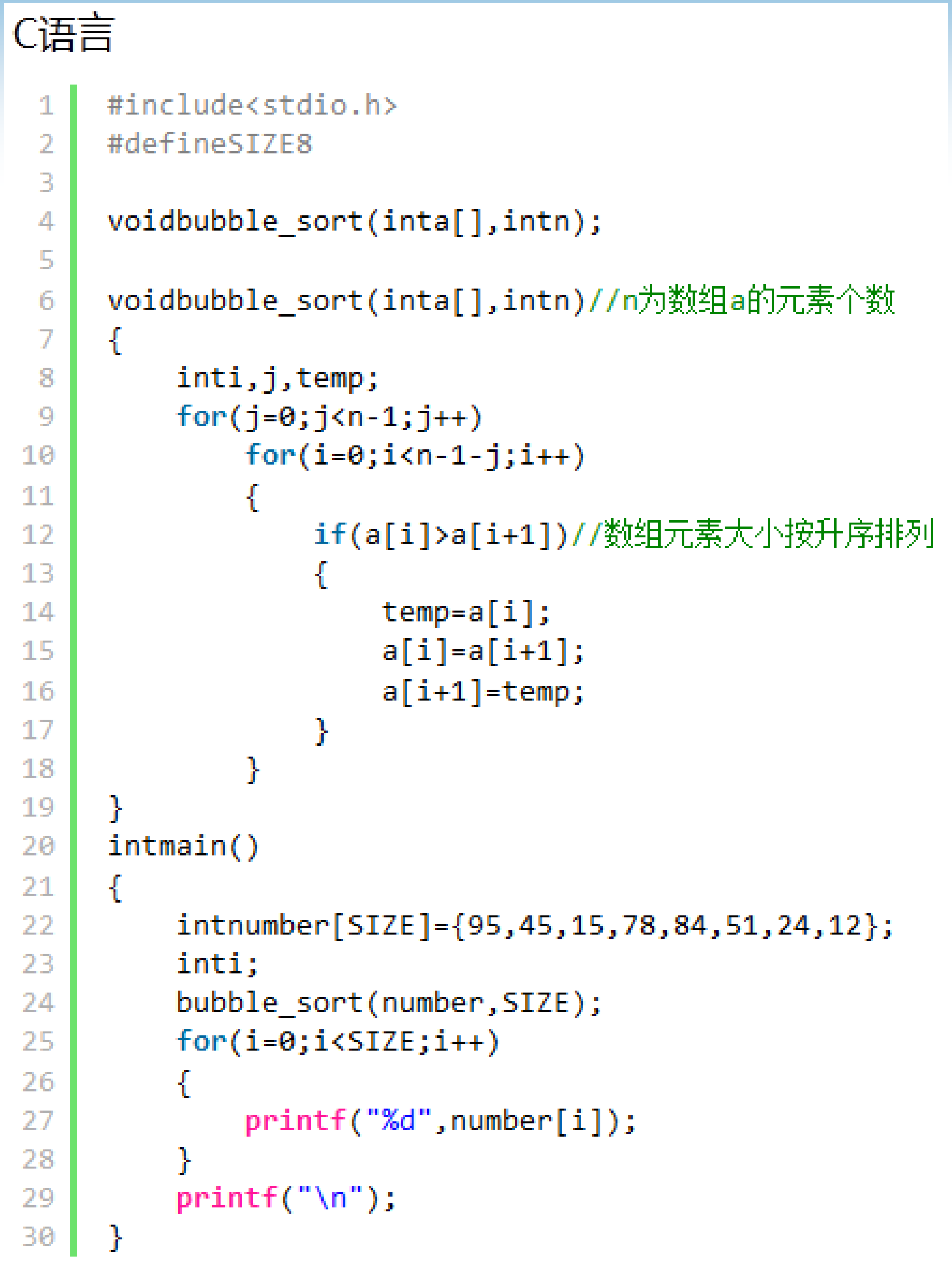
共需要**n-1**遍处理即完成排序。



Bubble Sort!

- Can be explained to a child.
- More or less efficient.

冒泡排序

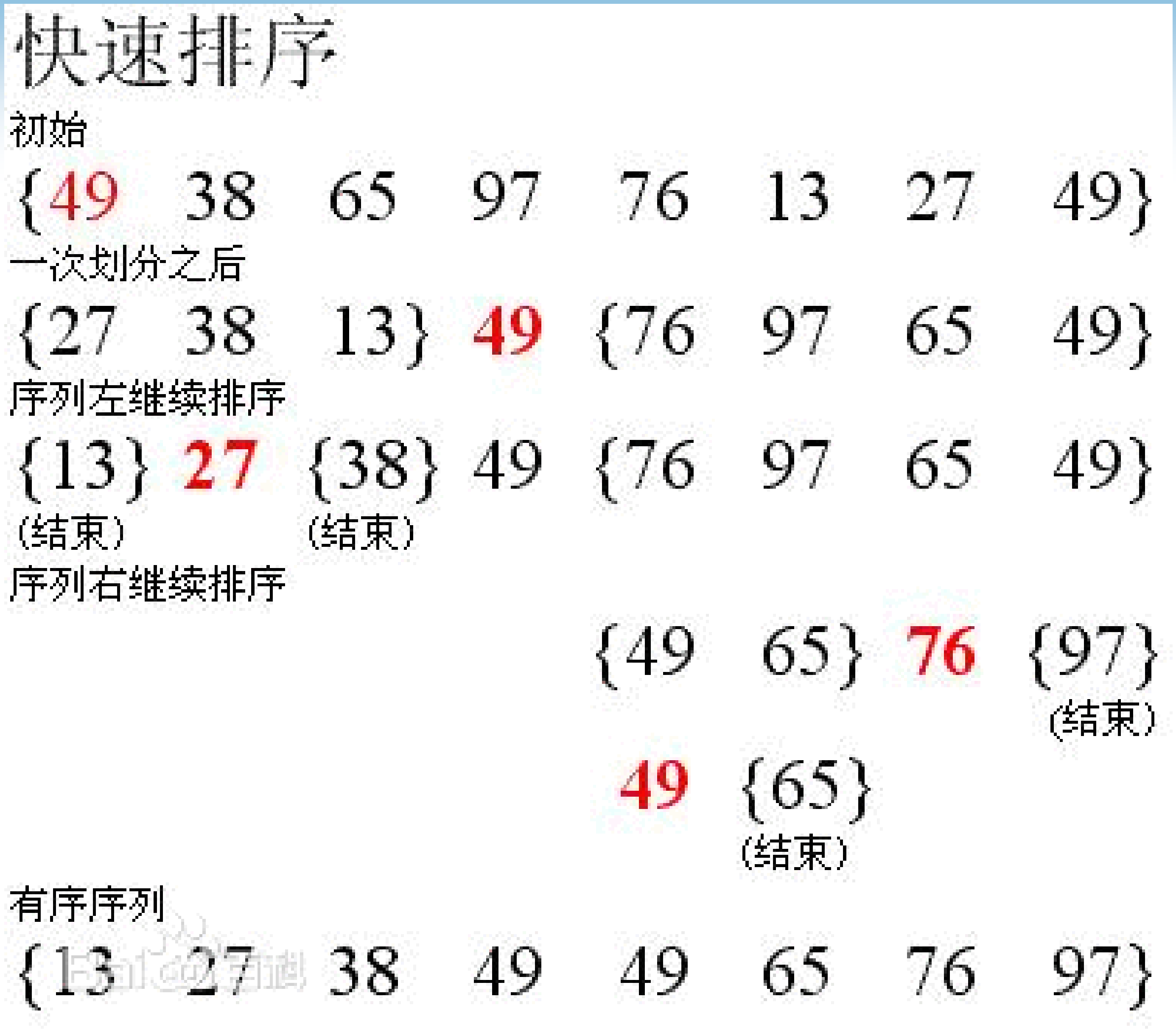


C语言

```
#include<stdio.h>
#defineSIZE8

voidbubble_sort(inta[],intn);

voidbubble_sort(inta[],intn)//n为数组a的元素个数
{
    inti,j,temp;
    for(j=0;j<n-1;j++)
        for(i=0;i<n-1-j;i++)
        {
            if(a[i]>a[i+1])//数组元素大小按升序排列
            {
                temp=a[i];
                a[i]=a[i+1];
                a[i+1]=temp;
            }
        }
}
intmain()
{
    intnumber[SIZE]={95,45,15,78,84,51,24,12};
    inti;
    bubble_sort(number,SIZE);
    for(i=0;i<SIZE;i++)
    {
        printf("%d",number[i]);
    }
    printf("\n");
}
```

快速排序

- 高效
- 分治思想：
  - 先保证列表的前半部分都小于后半部分，然后分别对前半部分和后半部分排序
  - 按此方法对这两部分数据分别进行快速排序，整个排序过程可以递归进行
- 算法的高效与否与列表中数字间的比较次数有直接的关系
- 前半部分的任何一个数不再跟后半部分的数进行比较



快速排序

初始

{49 38 65 97 76 13 27 49}

一次划分之后

{27 38 13} 49 {76 97 65 49}

序列左继续排序

{13} 27 {38} 49 {76 97 65 49}

(结束) (结束)

序列右继续排序

{49 65} 76 {97}

(结束)

49 {65}

(结束)

有序序列

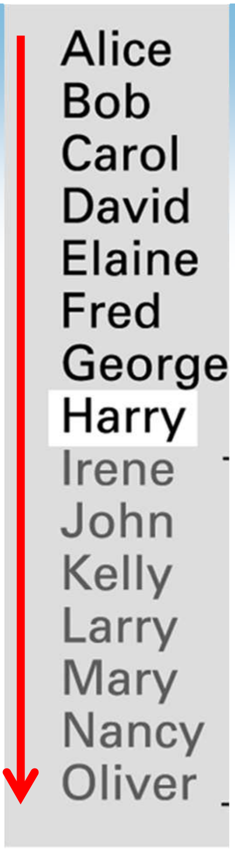
{13 27 38 49 49 65 76 97}

Video

Search algorithms

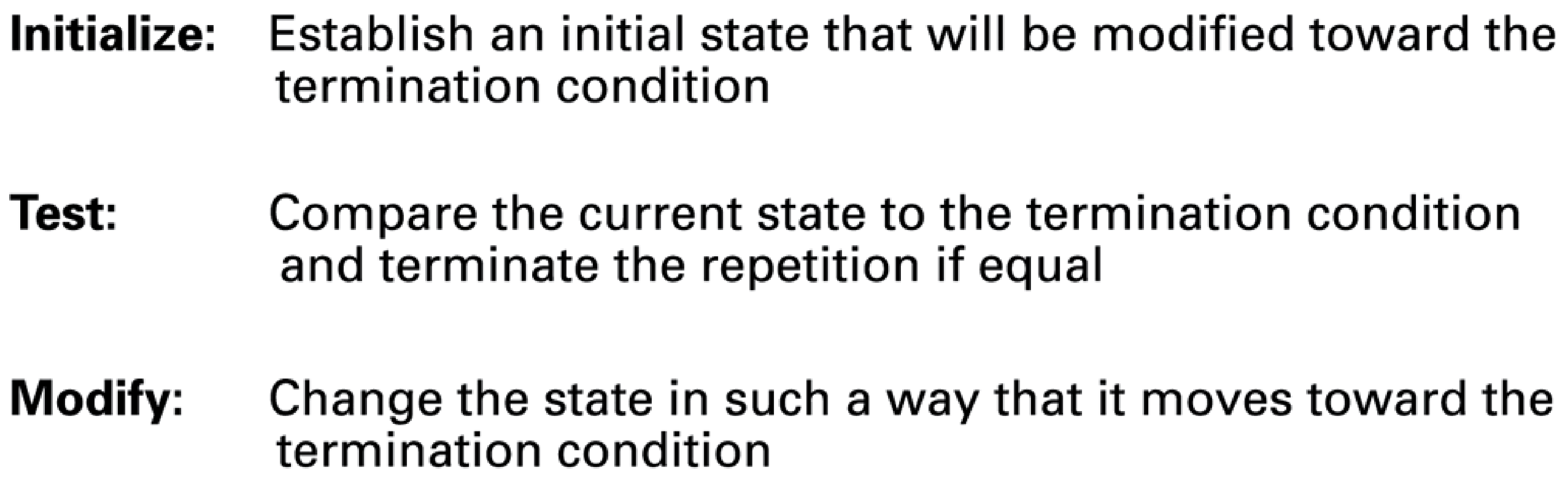
- Sequential search (顺序查找)
- Binary search (二分查找)

Sequential search algorithm in pseudocode



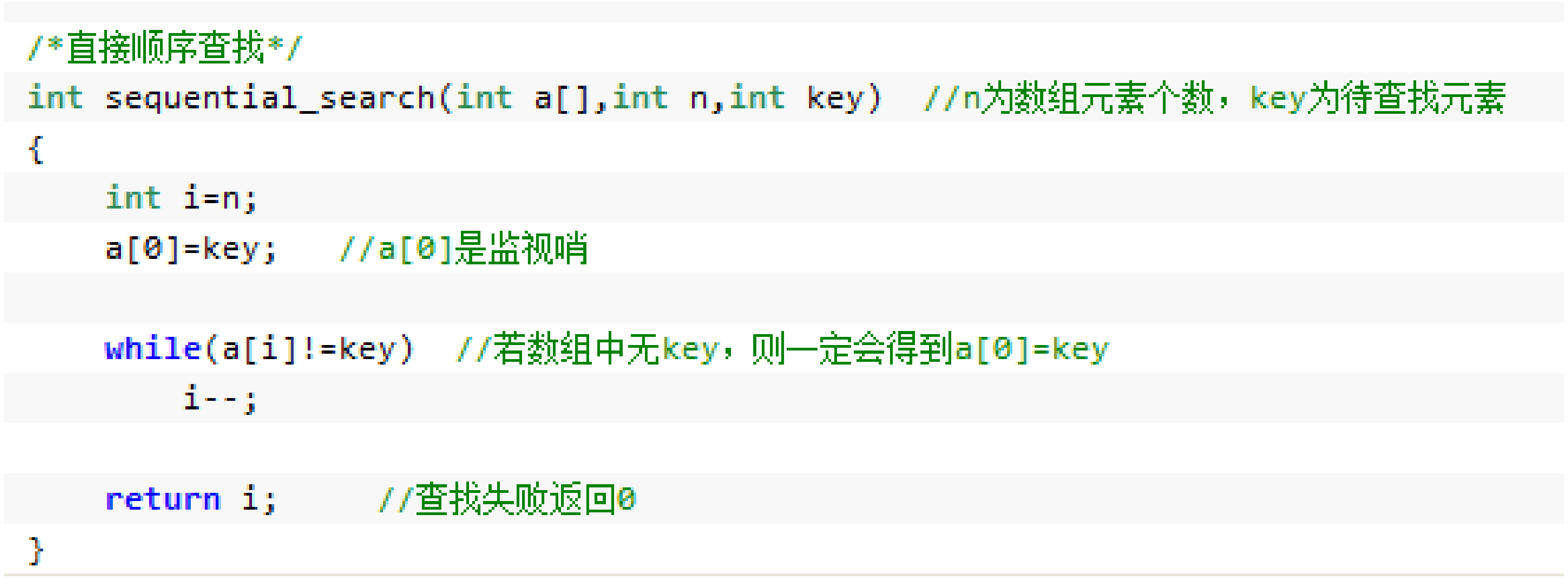
```
procedure Search (List, TargetValue)
if (List empty)
      then
         (Declare search a failure)
      else
         (Select the first entry in List to be TestEntry;
          while (TargetValue > TestEntry and
                           there remain entries to be considered)
                do (Select the next entry in List as TestEntry.);
          if (TargetValue = TestEntry)
                then (Declare search a success.)
                else (Declare search a failure.)
         ) end if
```

Figure 5.7 **Components of repetitive control**



Initialize:	Establish an initial state that will be modified toward the termination condition
Test:	Compare the current state to the termination condition and terminate the repetition if equal
Modify:	Change the state in such a way that it moves toward the termination condition

顺序查找

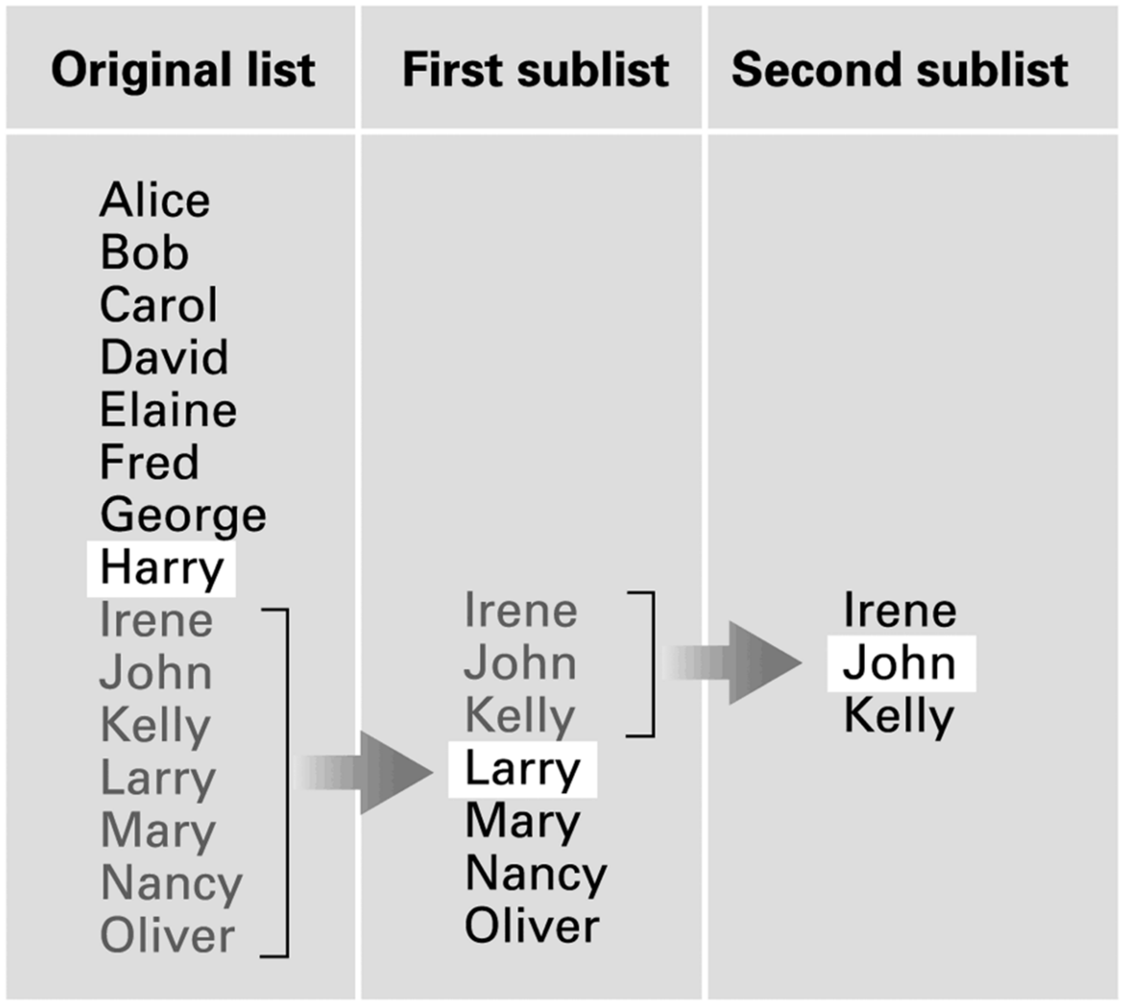


```
/*直接顺序查找*/
int sequential_search(int a[],int n,int key)  //n为数组元素个数，key为待查找元素
{
    int i=n;
    a[0]=key;   //a[0]是监视哨

    while(a[i]!=key)  //若数组中无key，则一定会得到a[0]=key
        i--;

    return i;     //查找失败返回0
}
```

二分查找

- 二分*查找*又称折半查找
- 优点：比较次数少，查找速度快，平均性能好；
- 缺点：要求待查表为有序表，且插入删除困难。因此，折半查找方法适用于不经常变动而查找频繁的有序列表

Figure 5.12 **Applying our strategy to search a list for the entry John**



Binary search technique

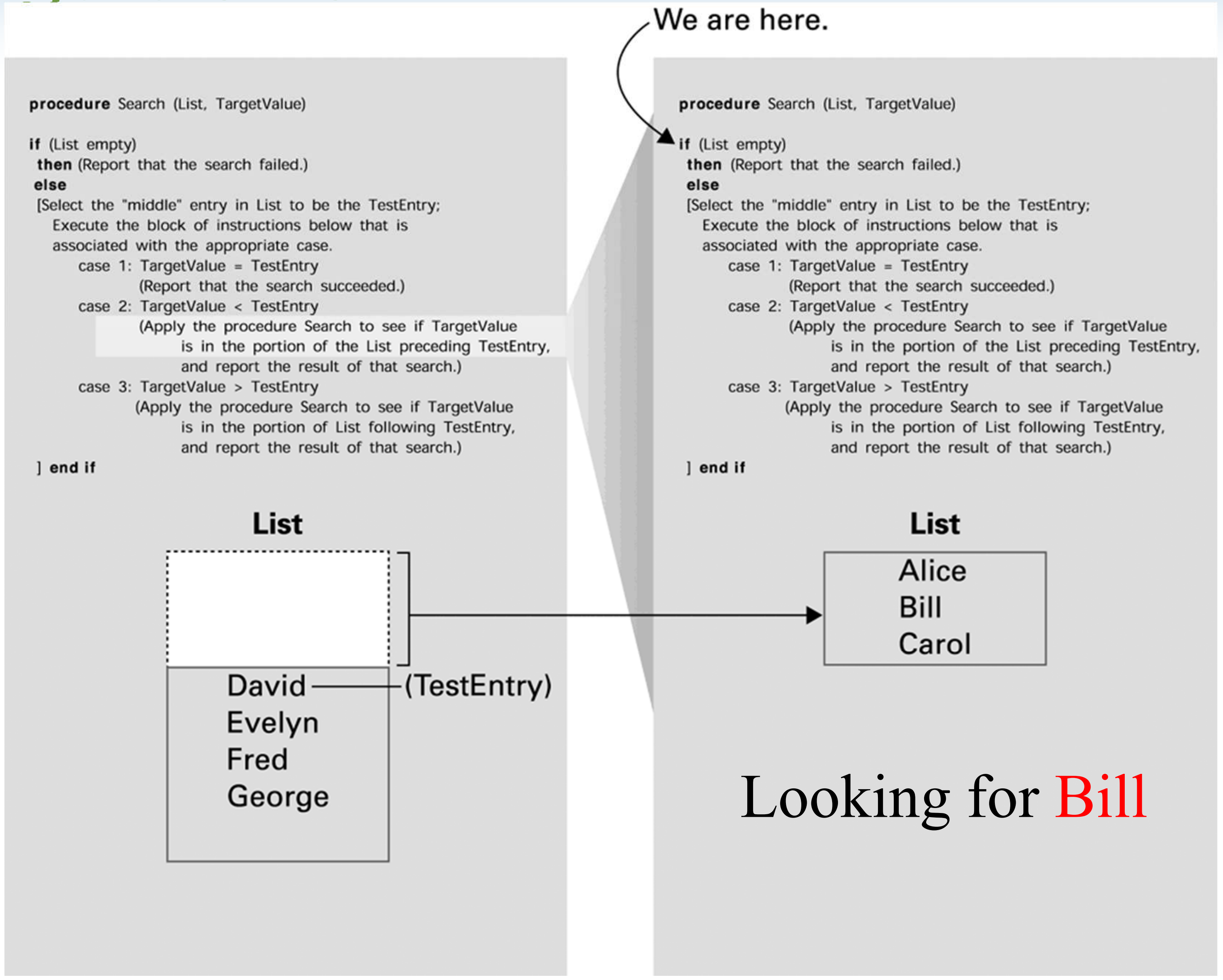
```
if (List empty)
 then
  (Report that the search failed.)
 else
   [Select the “middle” entry in the List to be the TestEntry;
    Execute the block of instructions below that is
      associated with the appropriate case.
        case 1: TargetValue = TestEntry
                (Report that the search succeeded.)
        case 2: TargetValue < TestEntry
                (Search the portion of List preceding TestEntry for
                     TargetValue, and report the result of that search.)
        case 3: TargetValue > TestEntry
                (Search the portion of List following TestEntry for
                     TargetValue, and report the result of that search.)
   ] end if
```

Figure 5.14 **The binary search algorithm in pseudocode**

```
procedure Search (List, TargetValue)
if (List empty)
  then
     (Report that the search failed.)
   else
     [Select the "middle" entry in List to be the TestEntry;
      Execute the block of instructions below that is
         associated with the appropriate case.
            case 1: TargetValue = TestEntry
                    (Report that the search succeeded.)
            case 2: TargetValue < TestEntry
                    (Apply the procedure Search to see if TargetValue
                         is in the portion of the List preceding TestEntry,
                         and report the result of that search.)
            case 3: TargetValue > TestEntry
                   (Apply the procedure Search to see if TargetValue
                        is in the portion of List following TestEntry,
                        and report the result of that search.)
     ] end if
```

Figure 5.15

Recursion structure



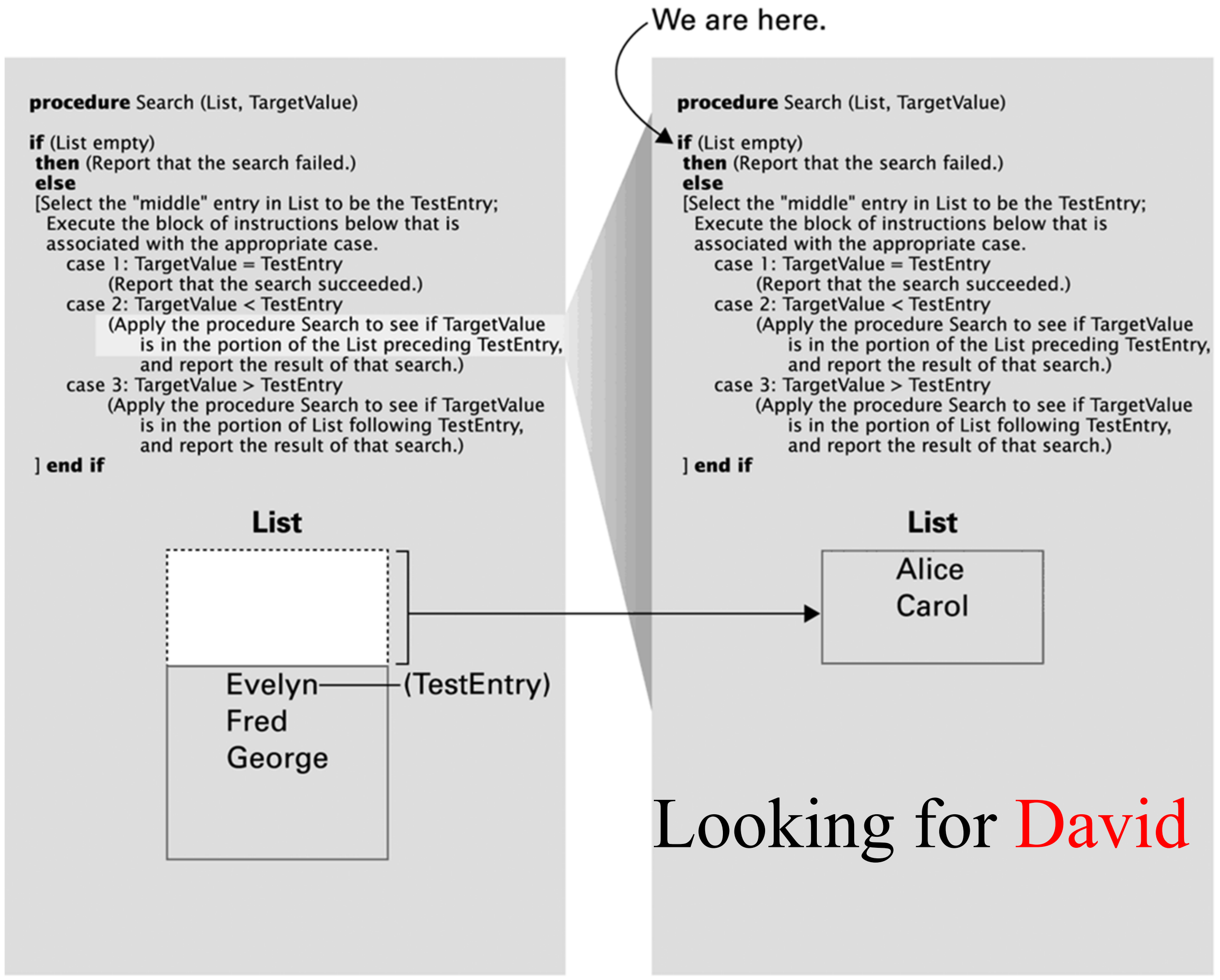
```
procedure Search (List, TargetValue)

if (List empty)
 then (Report that the search failed.)
 else
 [Select the "middle" entry in List to be the TestEntry;
   Execute the block of instructions below that is
   associated with the appropriate case.
      case 1: TargetValue = TestEntry
              (Report that the search succeeded.)
      case 2: TargetValue < TestEntry
              (Apply the procedure Search to see if TargetValue
                  is in the portion of the List preceding TestEntry,
                  and report the result of that search.)
      case 3: TargetValue > TestEntry
              (Apply the procedure Search to see if TargetValue
                  is in the portion of List following TestEntry,
                  and report the result of that search.)
 ] end if
```

Alice
Bill
Carol
David
Evelyn
Fred
George

Looking for Bill

Figure 5.16

Alice
Carol
Evelyn
Fred
George

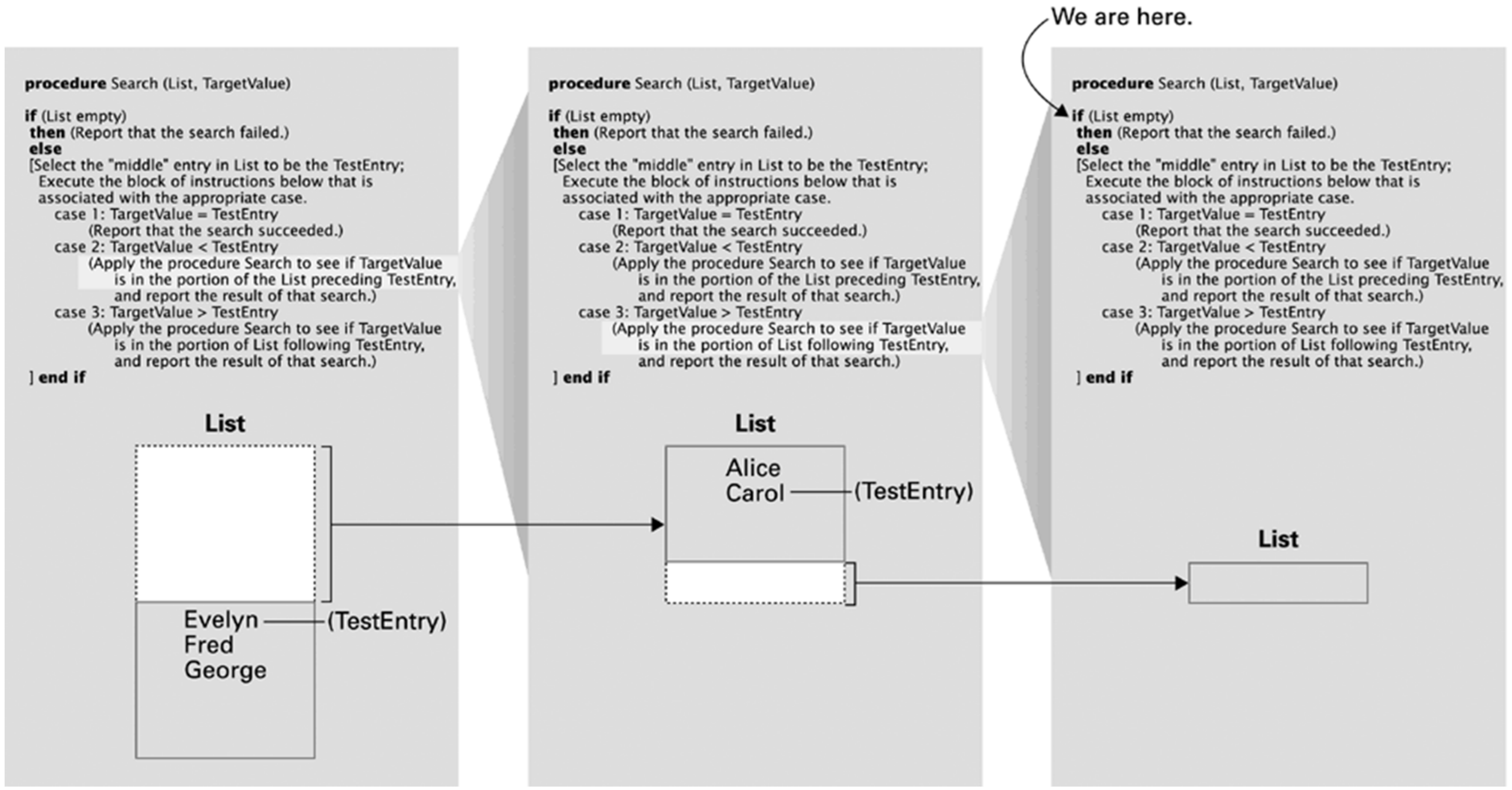


```
procedure Search (List, TargetValue)

if (List empty)
 then (Report that the search failed.)
 else
 [Select the "middle" entry in List to be the TestEntry;
   Execute the block of instructions below that is
   associated with the appropriate case.
      case 1: TargetValue = TestEntry
            (Report that the search succeeded.)
      case 2: TargetValue < TestEntry
            (Apply the procedure Search to see if TargetValue
                 is in the portion of the List preceding TestEntry,
                 and report the result of that search.)
      case 3: TargetValue > TestEntry
            (Apply the procedure Search to see if TargetValue
                 is in the portion of List following TestEntry,
                 and report the result of that search.)
 ] end if
```

```
procedure Search (List, TargetValue)

if (List empty)
 then (Report that the search failed.)
 else
 [Select the "middle" entry in List to be the TestEntry;
   Execute the block of instructions below that is
   associated with the appropriate case.
      case 1: TargetValue = TestEntry
            (Report that the search succeeded.)
      case 2: TargetValue < TestEntry
            (Apply the procedure Search to see if TargetValue
                 is in the portion of the List preceding TestEntry,
                 and report the result of that search.)
      case 3: TargetValue > TestEntry
            (Apply the procedure Search to see if TargetValue
                 is in the portion of List following TestEntry,
                 and report the result of that search.)
 ] end if
```

Looking for David

Figure 5.17



```
procedure Search (List, TargetValue)

if (List empty)
 then (Report that the search failed.)
 else
 [Select the "middle" entry in List to be the TestEntry;
  Execute the block of instructions below that is
  associated with the appropriate case.
    case 1: TargetValue = TestEntry
           (Report that the search succeeded.)
    case 2: TargetValue < TestEntry
           (Apply the procedure Search to see if TargetValue
              is in the portion of the List preceding TestEntry,
              and report the result of that search.)
    case 3: TargetValue > TestEntry
           (Apply the procedure Search to see if TargetValue
              is in the portion of List following TestEntry,
              and report the result of that search.)
 ] end if
```

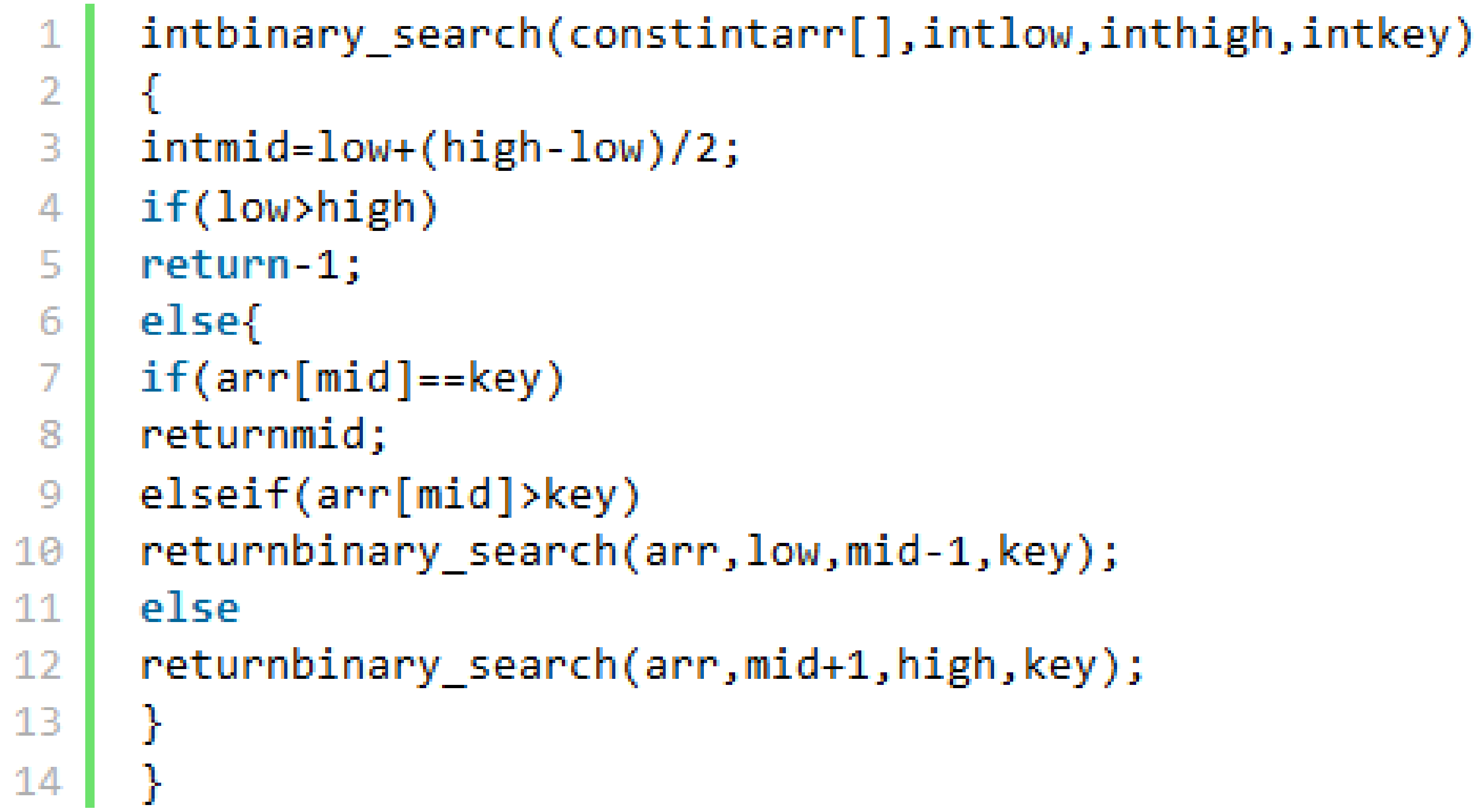
```
procedure Search (List, TargetValue)

if (List empty)
 then (Report that the search failed.)
 else
 [Select the "middle" entry in List to be the TestEntry;
  Execute the block of instructions below that is
  associated with the appropriate case.
    case 1: TargetValue = TestEntry
           (Report that the search succeeded.)
    case 2: TargetValue < TestEntry
           (Apply the procedure Search to see if TargetValue
              is in the portion of the List preceding TestEntry,
              and report the result of that search.)
    case 3: TargetValue > TestEntry
           (Apply the procedure Search to see if TargetValue
              is in the portion of List following TestEntry,
              and report the result of that search.)
 ] end if
```

```
procedure Search (List, TargetValue)

if (List empty)
 then (Report that the search failed.)
 else
 [Select the "middle" entry in List to be the TestEntry;
  Execute the block of instructions below that is
  associated with the appropriate case.
    case 1: TargetValue = TestEntry
           (Report that the search succeeded.)
    case 2: TargetValue < TestEntry
           (Apply the procedure Search to see if TargetValue
              is in the portion of the List preceding TestEntry,
              and report the result of that search.)
    case 3: TargetValue > TestEntry
           (Apply the procedure Search to see if TargetValue
              is in the portion of List following TestEntry,
              and report the result of that search.)
 ] end if
```

Looking for David

二分查找

C语言代码 递归实现



```
intbinary_search(constintarr[],intlow,inthigh,intkey)
{
intmid=low+(high-low)/2;
if(low>high)
return-1;
else{
if(arr[mid]==key)
returnmid;
elseif(arr[mid]>key)
returnbinary_search(arr,low,mid-1,key);
else
returnbinary_search(arr,mid+1,high,key);
}
}
```