西北工业大学
NORTHWESTERN POLYTECHNICAL UNIVERSITY

C程序设计

Programming in C

1011014

主讲：姜学锋，计算机学院

实现查找算法

1、顺序查找法

2、二分查找法

3、插值查找法

4、斐氏查找法

6.5 数组应用程序举例

- （1）顺序查找法
- 顺序查找的基本思想是让关键字与序列中的数逐个比较，直到找出与给定关键字相同的数为止或序列结束，一般应用于无序序列查找。

6.5 数组应用程序举例

【例6.14】

编写顺序查找函数Search，从一个无序数组中查找数据的位置。

6.5 数组应用程序举例

例6.14

```
 1 #include <stdio.h>
 2 int Search(int A[],int n,int find)
 3 { //顺序查找 n=序列元素个数 find=欲查找数据
 4   int i;
 5   for (i=0; i<n ; i++) if (A[i]==find) return i;
 6   return -1; //未找到
 7 }
 8 #define N 10
 9 int main()
10 {
11   int A[N]={18,-3,-12,34,101,211,12,90,77,45}, i,find;
12   scanf("%d",&find);
13   i=Search(A,N,find);
14   if(i>=0) printf("A[%d]=%d\n",i,find);
15   else printf("not found\n");
```

6.5 数组应用程序举例

例6.14

```
    return 0;
}
```

6.5 数组应用程序举例

例6.14

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

程序运行屏幕

```
A[4]=101
```

```
101↙
```

- （2）二分查找法
- 对于有序序列，可以采用二分查找法进行查找。它的基本思想是：升序排列的n个元素集合A分成个数大致相同的两部分，取A[n/2]与欲查找的find作比较，如果相等则表示找到find，算法终止。如果find<A[n/2]，则在A的前半部继续搜索find，如果find>A[n/2]，则在A的后半部继续搜索find。

6.5 数组应用程序举例

【例6.15】

编写二分查找函数BinarySearch，从一个有序数组中查找数据的位置。

6.5 数组应用程序举例

例6.15

```
 1 #include <stdio.h>
 2 int BinarySearch(int A[],int n,int find)
 3 { //二分查找 n=序列元素个数 find=欲查找数据
 4   int low,upper,mid;
 5   low=0 , upper=n-1; //左右两部分
 6   while(low<=upper) {
 7     mid = low + (upper-low)/2; //不用(upper+low)/2，避免upper+low溢出
 8     if( A[mid] < find) low = mid+1; //右半部分
 9     else if (A[mid] > find) upper = mid - 1; //左半部分
10     else return mid; //找到
11   }
12   return -1; //未找到
13 }
14 #define N 10
```

6.5 数组应用程序举例

例6.15

```
15 int main()
16 {
17    int A[N]={8,24,30,47,62,68,83,90,92,95},i,find;
18    scanf("%d",&find);
19    i=BinarySearch(A,N,find);
20    if(i >= 0) printf("A[%d]=%d\n",i,find);
21    else printf("not found\n");
22    return 0;
23 }
```

6.5 数组应用程序举例

例6.15

程序运行屏幕

```
A[8]=92
```

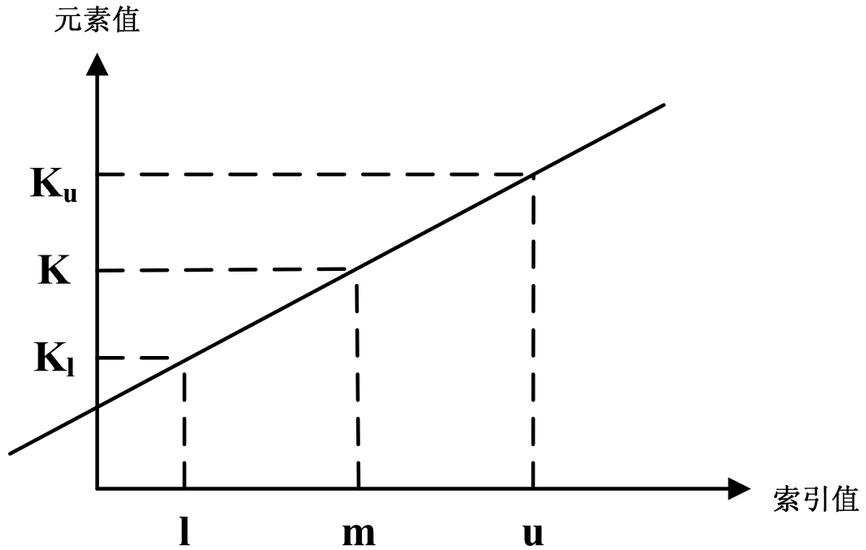
92↙

- （3）插值查找法
- 如果欲查找的数据分布平均时，可以使用插值法（Interpolation），或称为插补法来进行查找，在查找的数据对象大于500时，插值查找法会比二分查找法来得快速。

6.5 数组应用程序举例

- 插值查找法是以数据分布的近似直线来作比例运算，以求出中间的索引并进行数据比对，如果取出的值小于要寻找的值，则提高下界，如果取出的值大于要寻找的值，则降低下界，如此不断地减少查找的范围。

- 所以插值查找法的原理与二分查找法是相同的，至于中间值的寻找是通过比例运算。如下所示，其中K是指定要寻找的对象，而m则是可能的索引值：



$$\frac{K-K_l}{K_u-K_l}=\frac{m-l}{u-l}$$

$$m=(u-l)\frac{K-K_1}{K_u-K_1}+1$$

6.5 数组应用程序举例

【插值查找法举例】

编写查找函数InterpolationSearch，从一个数组中查找数据的位置。

6.5 数组应用程序举例

例6.61

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <time.h>
4  int InterpolationSearch(int A[],int n, int find)
5  {
6    int low,upper,mid;
7    low = 0;
8    upper = n - 1;
9    while(low <= upper) {
10     mid=(upper-low)*(find-A[low])/(A[upper]-A[low])+low;
11     if(mid < low || mid > upper) break;
12     if(find < A[mid]) upper=mid - 1;
13     else if(find > A[mid]) low=mid + 1;
14     else return mid;
15   }
```

6.5 数组应用程序举例

例6.61

```
  return -1;
}
void QuickSort(int A[],int n,int left,int right)
{ //快速排序 n为数组元素个数 left=数组左边界 right=数组右边界
  int i,j,t;
  if(left<right) { //一趟快速排序
    i=left; j=right + 1;
    while(1) {
      while(i+1<n&& A[++i]<A[left]); //向后搜索 <升序 >降序
      while(j-1>-1&& A[--j]>A[left]); //向前搜索 >升序 <降序
      if(i>=j) break;
      t=A[i], A[i]=A[j], A[j]=t; //交换
    }
    t=A[left], A[left]=A[j], A[j]=t; //交换
    QuickSort(A, n, left, j-1); //关键数据左半部分递归
```

6.5 数组应用程序举例

例6.61

```
31      QuickSort(A, n, j+1, right); //关键数据右半部分递归
32    }
33 }
34 #define N 100
35 int main()
36 {
37    int A[N],i;
38    int find;
39    srand((unsigned int)time(0)); //设置随机数种子
40    for(i=0; i<N; i++) //随机产生N个数
41      A[i] = rand()%1000;
42    QuickSort(A,N,0,N-1);
43    for(i=0; i<N; i++) printf("%d ", A[i]); //输出排序结果
44    printf("\n");
45    scanf("%d", &find); //输入查找对象
```

6.5 数组应用程序举例

例6.61

```
46    if((i = InterpolationSearch(A, N, find)) >= 0)
47      printf("found:%d\n", i);
48    else
49      printf("not found\n");
50    return 0;
51  }
```

- （4）斐氏查找法
- 二分查找法每次查找时，都会将查找区间分为一半，所以其查找时间为O($\log_2 n$)，这里要介绍的斐氏查找，其利用斐波那契(Fibonacci)数列作为间隔来查找下一个数，所以区间收敛的速度更快，查找时间为O(logn)。

6.5 数组应用程序举例

- 斐氏查找使用斐波那契(Fibonacci)数列来决定下一个数的查找位置，所以必须先制作斐波那契数列；

- 1,1,2,3,5,8,13,21,34,55,89,144,233,377,610,987,1597,……

6.5 数组应用程序举例

- 第一个查找数的位置可以由下面公式求得，其中n为查找对象的个数，Fy为第y个斐式数，必须大于等于n，若算出x值，则使用Fx作为第一个查找下标，也就是第x个斐式数：
- Fy + m = n
- Fy >= n + 1
- x = y - 1

6.5 数组应用程序举例

- 以10个查找对象来说：
- Fy + m = 10
-
- 取Fy = 8, m = 2，对照斐波那契数列得到8是第六个斐式数，所以y=6，因而x得5，也就是使用第五个斐式数的值（也就是5）作为下标开始查找。

6.5 数组应用程序举例

- 如果数组在下标5处的值大于指定的查找值，则第一个查找位置就是下标5的位置，如果小于指定的查找值，则第一个查找位置必须加上m，也就是F5 + m = 5 + 2 = 7，也就是下标7的位置，其实加上m的原因，是为了要让下一个查找值刚好是数组的最后一个位置。

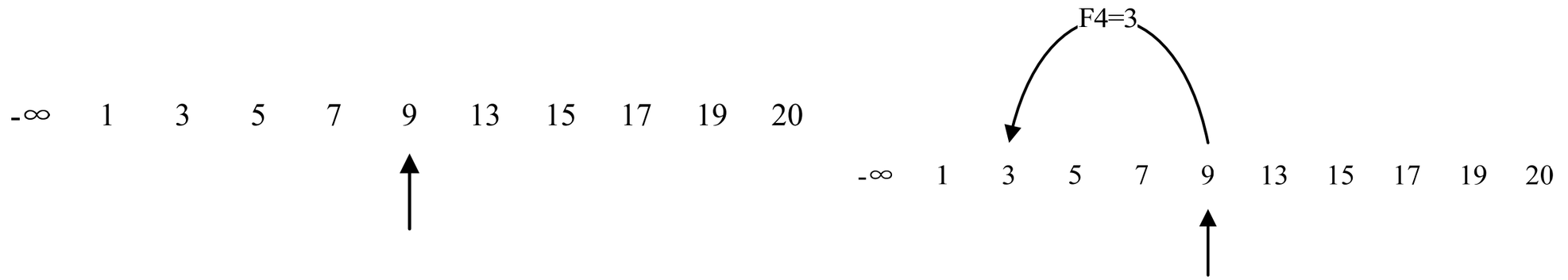
6.5 数组应用程序举例

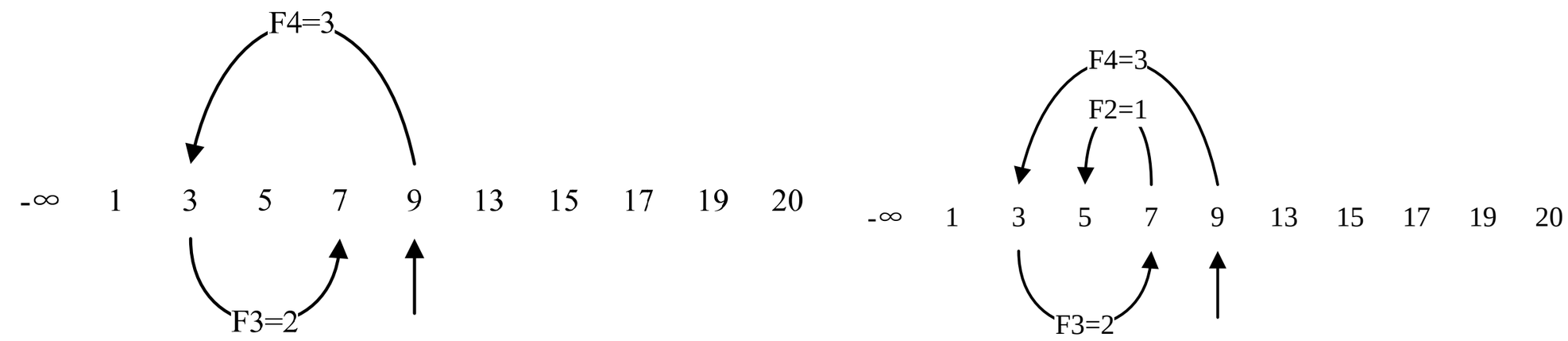
- 例如若在下面的数组中查找（为了计算方便， 通常会将下标0当作无限小的数，而数组由下标1开始）：

- -∞ 1 3 5 7 9 13 15 17 19 20

6.5 数组应用程序举例

- 如图所示：

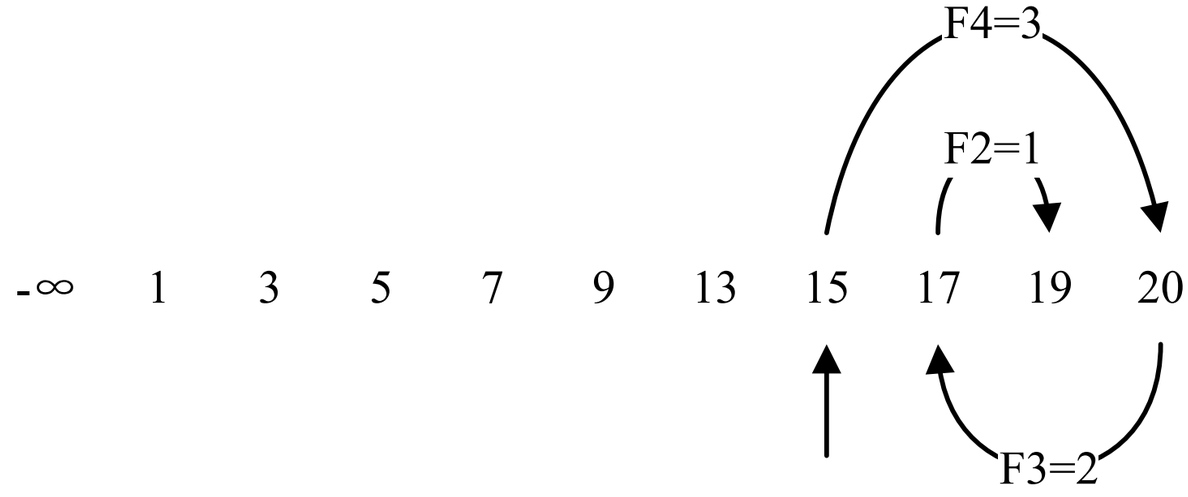




6.5 数组应用程序举例

- 如果要查找5的话，则由下标F5（F5表示第5个斐波那契数作为下标，也就是5）开始查找，接下来如果数列中的数大于指定查找值时，就往左找，小于时就向右，每次找的间隔是F4（第4个斐波那契数作为下标，也就是3）、F3（第3个斐波那契数作为下标，也就是2）、F2（第2个斐波那契数作为下标，也就是1）来寻找，当斐波那契数为0时还没找到，就表示寻找失败。

- 如果要查找19，由于第一个查找值下标F5处的值小于19，所以此时必须对齐数列右方，也就是将第一个查找值的下标改为F5+2 = 7，然后用上述的方式进行查找，如下所示：



6.5 数组应用程序举例

- 斐波那契查找除了收敛快速之外，由于其本身只会使用到加法与减法，在运算上也可以加快。

6.5 数组应用程序举例

【斐氏查找法举例】

编写查找函数FibSearch，从一个数组中查找数据的位置。

6.5 数组应用程序举例

例6.62

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <time.h>
4  #define N 10
5  int FibSearch(int A[], int find)
6  { //斐波那契查找
7    int i, x, m;
8    int Fib[N] = {-99999};
9    Fib[0]=0;
10   Fib[1]=1;
11   for(i=2; i<N; i++) //建立斐波那契数列
12     Fib[i] = Fib[i-1] + Fib[i-2];
13   x=0;
14   while(Fib[x]<=N+1) x++; //找x值
15   x--; //Fib不大于N+1的x
```

6.5 数组应用程序举例

例6.62

```
16    m=N-Fib[x]; //差值
17    x--; //左边界
18    i=x;
19    if(A[i]<find) i=i+m; //若大于斐氏数，则从右方开始查找
20    while(Fib[x]>0) {
21      if(A[i] < find) i=i+ Fib[--x]; //查找后半段
22      else if(A[i] > find) i =i- Fib[--x]; //查找前半段
23      else return i; //找到
24    }
25    return -1;
26  }
27  void QuickSort(int A[],int n,int left,int right)
28  { //快速排序 n为数组元素个数 left=数组左边界 right=数组右边界
29    int i,j,t;
30    if(left<right) { //一趟快速排序
```

6.5 数组应用程序举例

例6.62

```
31      i=left; j=right + 1;
32      while(1) {
33        while(i+1<n&&A[++i]<A[left]); //向后搜索 <升序 >降序
34        while(j-1>-1&&A[--j]>A[left]); //向前搜索 >升序 <降序
35        if(i>=j) break;
36        t=A[i], A[i]=A[j], A[j]=t; //交换
37      }
38      t=A[left], A[left]=A[j], A[j]=t; //交换
39      QuickSort(A, n, left, j-1); //关键数据左半部分递归
40      QuickSort(A, n, j+1, right); //关键数据右半部分递归
41    }
42  }
43  int main()
44  {
45    int A[N] = {0};
```

6.5 数组应用程序举例

例6.62

```
46    int i, find;
47    srand((unsigned int)time(0)); //设置随机数种子
48    for(i=0; i<N; i++) //随机产生N个数
49      A[i] = rand()%100;
50    QuickSort(A,N,0,N-1);
51    for(i=0; i<N; i++) printf("%d ", A[i]); //输出排序结果
52    printf("\n");
53    scanf("%d", &find); //输入查找对象
54    if((i=FibSearch(A, find)) >= 0)
55      printf("found:%d\n", i);
56    else
57      printf("not found\n");
58    return 0;
59  }
```


程序设计