程序设计语言与方法(C语言)

第八章 数组

求5个整数的平均值

- 定义（声明）变量
  - x1,x2,x3,x4,x5(待统计的5个数据)
  - x(平均值)
  - 过程
    - 输入数据；/* 能用循环输入这5个数据么？ */
    - x = (x1 + x2 + x3 + x4 + x5) / 5;
    - 输出x;

问题扩展

- 求用户输入的100个整数的平均值
- 问题描述
  - 数据：100个整数，平均值
  - 过程：
    - 输入100个整数
    - 计算100个整数的平均值
    - 输出结果（平均值）

数据分析

- 数据个数
  - 100
- 每个数据的类型一致
  - 整型
  - 数据类型相同

一维数组

- 一组具有相同类型的变量的集合
- int x[100]; /*数组的类型是其中的数据的类型*/
  - 数据基类型 数组名称 [数组的元素个数];
- 声明一维数组时，必需明确指定元素个数

定义数组

- 定义方式1
- int array[100];　　/*定义一个包含100个元素的一维数组，元素无缺省值*/
- 定义方式2
  - #define SIZE 10
  - ……
  - int array[SIZE]; /*定义一个包含SIZE(100)个元素的一维数组，元素无缺省值*/
- 定义方式3
  - int x[5] = {1,2,3,4,5}; /*定义一维数组，每个元素有缺省值*/
  - int x[] = {1,2,3,4,5};
    - 定义一维数组（数组的元素个数等于后面的数据个数），
    - 每个元素有缺省值

使用数组（数组访问）

- 按下标访问数组中的元素（确定位置的变量）
  - 数组名称[下标]； 下标就是变量在数组中的序号
  - x[10]
- 注意：
  - 下标仅仅是表示一个变量在数组中的位置（第几个）
  - 下标是从0开始
  - 最后一个变量的下标 等于 数组的元素个数–1
  - 100个变量的数组x
    - x[0] x[1] x[50] x[99]

对数组赋值

- 对数据赋值实质上是对数组中的每个变量进行赋值
- 使用循环赋值
  - #define SIZE 50
  - ……
  - int x[SIZE];
  - ……
  - for(i = 0; i < SIZE; i++) {　　//从第0个变量开始赋值（正向）
  - 　x[i] = i;
  - }
  - 或
  - for(i = SIZE – 1; i >= 0; i--) {　　//从最后一个变量开始赋值（逆向）
  - 　x[i] = i;
  - }

解决问题：100个整数的平均值

```
#include <stdio.h>

#define SIZE 100

int mian()
{
int x[SIZE]; // int x[100]
double r;

printf("please input 100 integers: ")
for(i = 0; i < SIZE; i++) {   //for(i = 0; i < 100; i++) {
  scanf("%d", &x[i]);
}
r = 0;
for(i = 0; i < SIZE; i++) {
  r += x[i];
}
printf("%lf\n", r / SIZE);
  return 0;
}
```

向函数传递一维数组

- 求给定整形数组的所有数据的均值
  - 输入（形参）：一个整型数组
    - int x[?]
  - 输出（返回值类型）：平均值
    - double

求均值

```
#include <stdlib.h>
#include <time.h>
#include <stdio.h>

#define Size 32
double mean(int x[]);
int main()
{
  int i, datas[Size];
  double avg;
  srand(time(NULL));
  for(i = 0; i < Size; i++) {
    datas[i] = rand() % 100 + 1;
  }
  avg = mean(datas);
  printf("avg: %ld\n", avg);
  return 0;
}
```

```
double mean(int x[])
{
  double sumx = 0.0;
  int i;
  for(i = 0; i < Size; i++) {
    sumx += x[i];
  }
  return sumx / Size;
}
```

排序和查找

- 排序
  - 将数组的变量按值的大小进行序列规整
- 查找
  - 从数组中查找某个数据的位置
    - 查找的是数据，而不是变量
    - 结果是值与给定数据相等的变量的位置

- 无序数组
- 有序数组

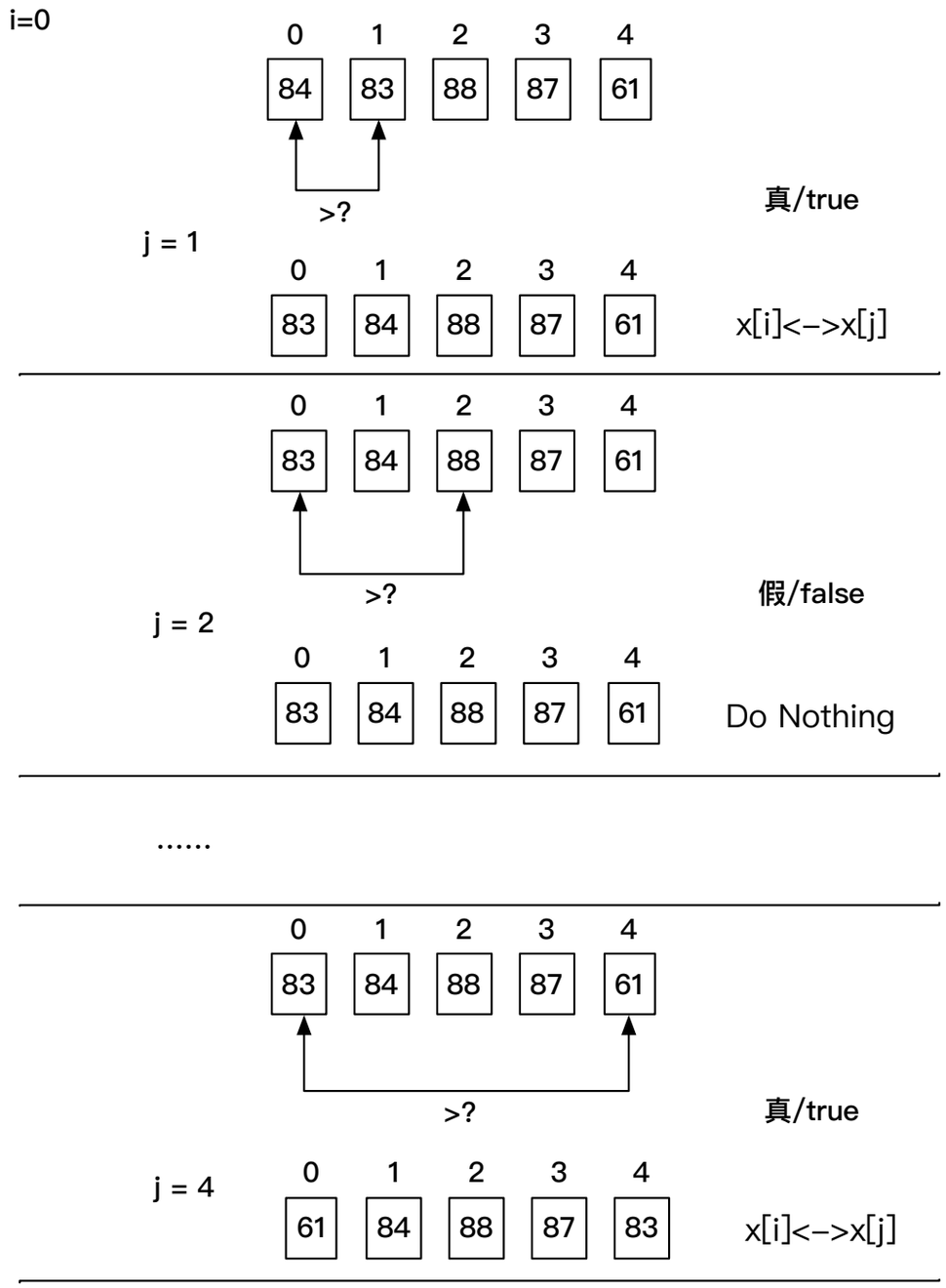
排序

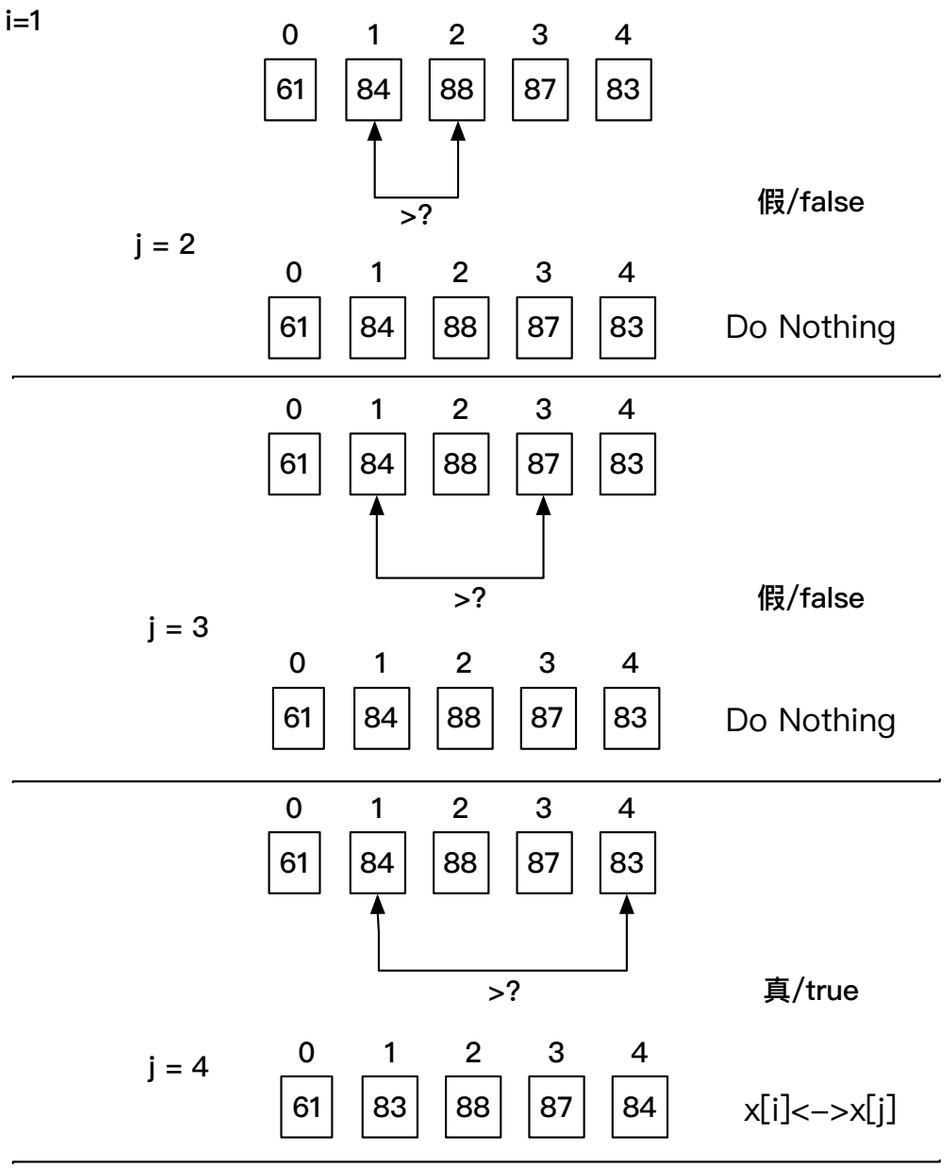
- 将100个整型数据按从小到大的顺序进行排列
  - 生成模拟数据
    - time(NULL); //获取系统时间，time.h，time_t time(time_t * arg);
    - srand(time(NULL)); //设置随机数种子，
      - stdlib.h, void srand(unsigned int seed);
    - rand(); //生成一个随机数，stdlib.h，int rand();
  - 对数组中的变量进行赋值
    - 逐个赋值
    - int ArrayName[ArraySize]; //int x[100]; x->ArrayName; 100->ArraySize
    - srand(time(NULL)); //也可以不设置生成随机数的种子
    - for(i = 0; i < ArraySize; i++) {
      - ArrayName[i] = rand() % 100 + 1; // 会得到什么样的一组数据?
    - }

排序

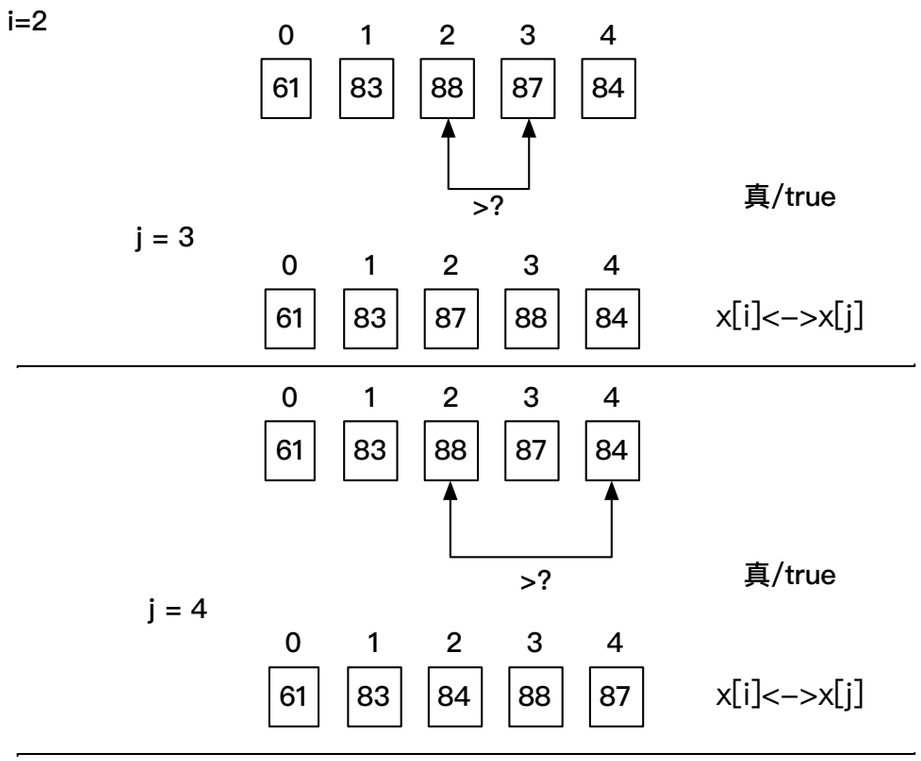
- 将100个整型数据按从小到大的顺序进行排列
  - 交换/冒泡排序法
  - 选择排序法
  - 插入排序法
  - 堆排序法
  - ……

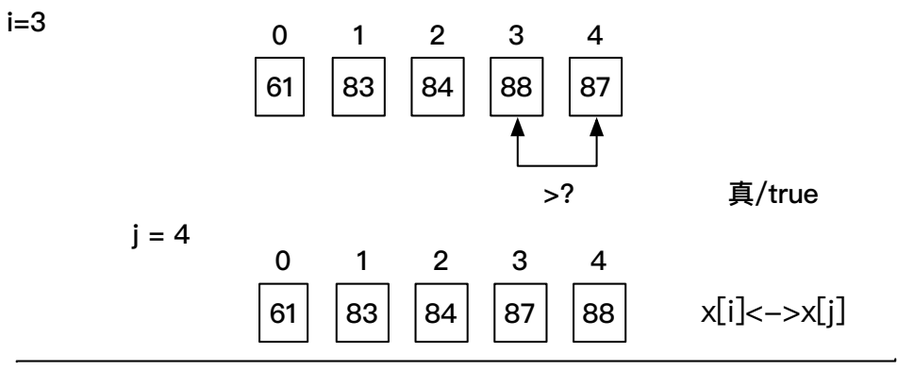
交换/冒泡排序


i=0
0 1 2 3 4
84 83 88 87 61
>?
真/true
j = 1
0 1 2 3 4
83 84 88 87 61
x[i]<->x[j]
0 1 2 3 4
83 84 88 87 61
>?
假/false
j = 2
0 1 2 3 4
83 84 88 87 61
Do Nothing
.....
0 1 2 3 4
83 84 88 87 61
>?
真/true
j = 4
0 1 2 3 4
61 84 88 87 83
x[i]<->x[j]


i=1
0 1 2 3 4
61 84 88 87 83
>?
假/false
j = 2
0 1 2 3 4
61 84 88 87 83
Do Nothing
0 1 2 3 4
61 84 88 87 83
>?
假/false
j = 3
0 1 2 3 4
61 84 88 87 83
Do Nothing
0 1 2 3 4
61 84 88 87 83
>?
真/true
j = 4
0 1 2 3 4
61 83 88 87 84
x[i]<->x[j]

交换/冒泡排序

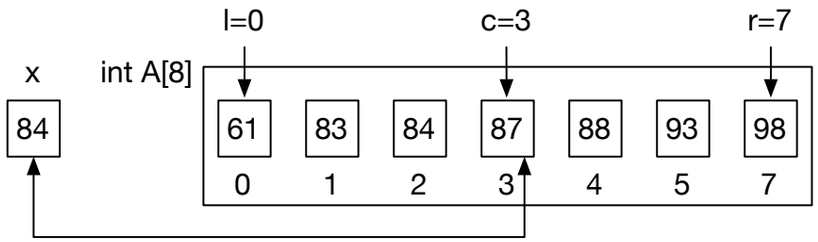

i=2
0 1 2 3 4
61 83 88 87 84
>?
真/true
j = 3
0 1 2 3 4
61 83 87 88 84
x[i]<–>x[j]
0 1 2 3 4
61 83 88 87 84
>?
真/true
j = 4
0 1 2 3 4
61 83 84 88 87
x[i]<–>x[j]


i=3
0 1 2 3 4
61 83 84 88 87
>?
真/true
j = 4
0 1 2 3 4
61 83 84 87 88
x[i]<–>x[j]

查找

- 找到第一个 VS 未找到
- 顺序查找
- 折半查找/二分法

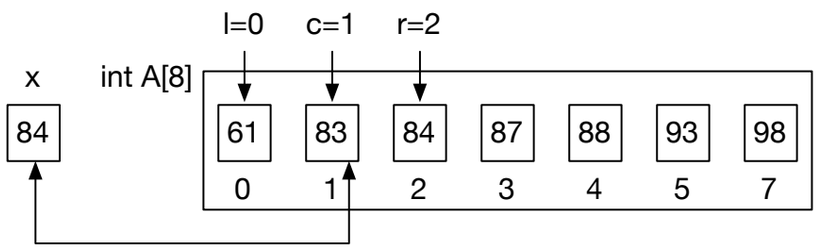
折半查找



x == A[c]? false

x > A[c]? false

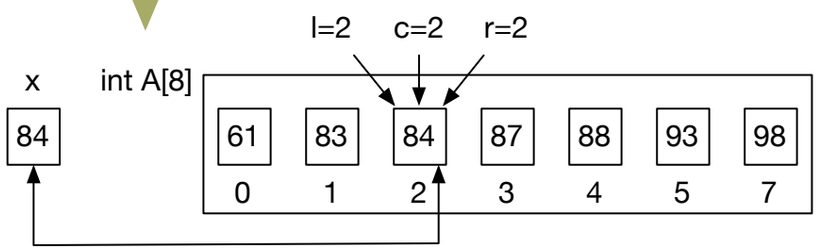
所以 找左半部分，调整右边界r=c−1



x == A[c]? false

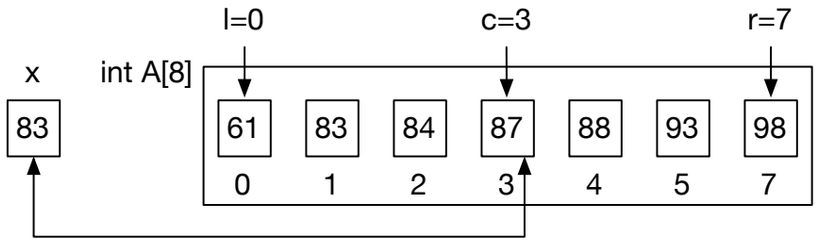
x > A[c]? true

所以 找右半部分，调整左边界l=c+1



x == A[c]? true

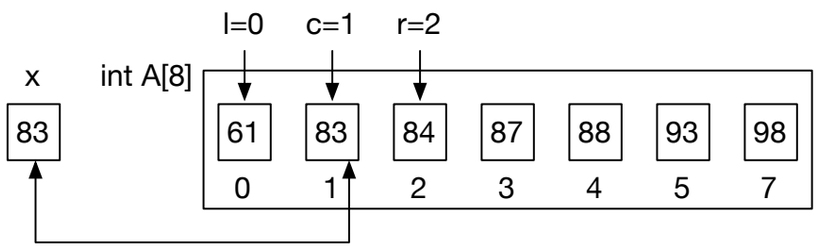
找到，返回下标2



x == A[c]? false

x > A[c]? false

所以 找左半部分，调整右边界r=c−1

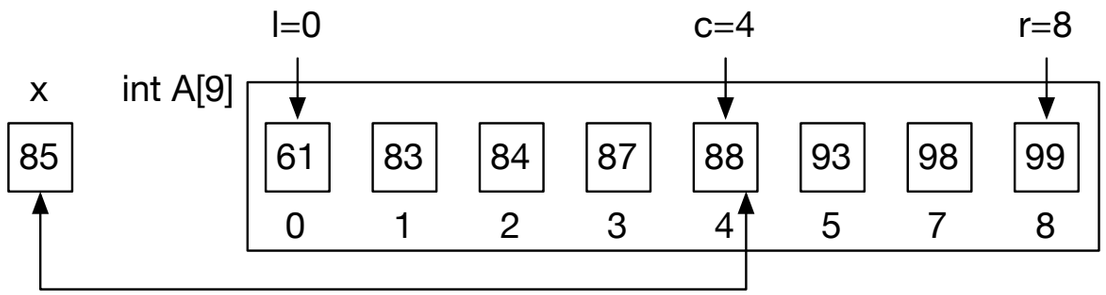


x == A[c]? true

找到，返回下标1

$$c=(l+r)/2$$

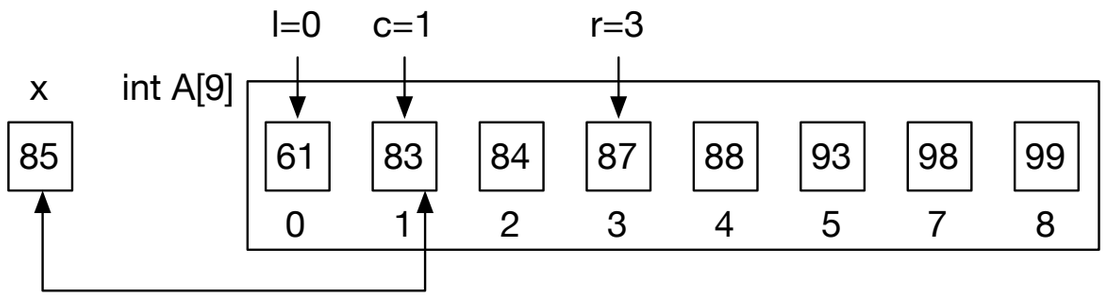
折半查找示例–续



x == A[c]? false

x > A[c]? false

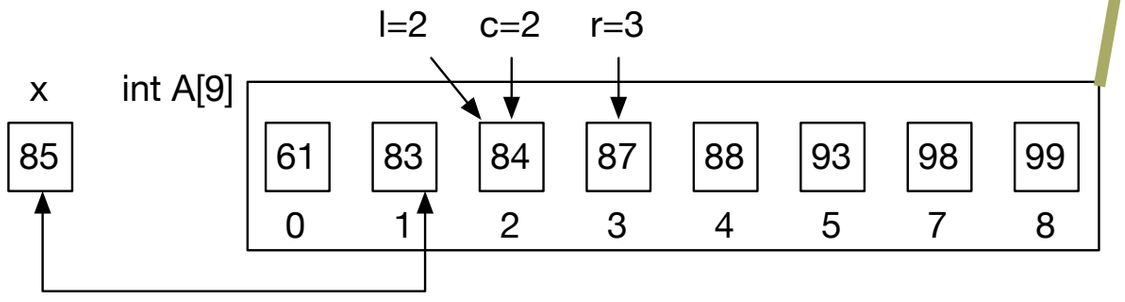
所以 找左半部分，调整右边界r=c–1



x == A[c]? false

x > A[c]? true

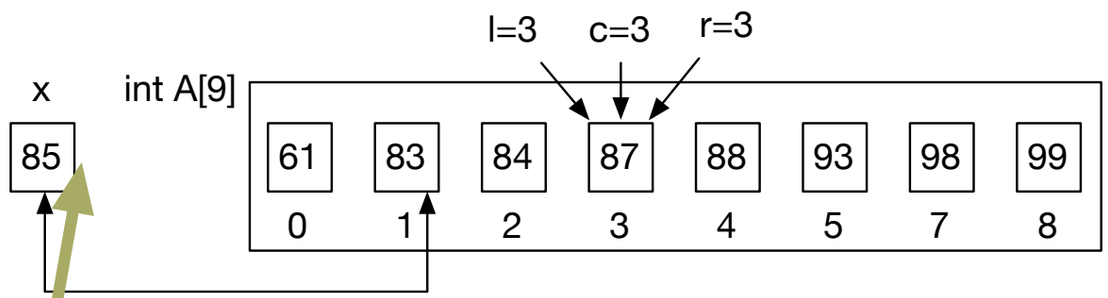
所以 找右半部分，调整左边界l=c+1



x == A[c]? false

x > A[c]? true

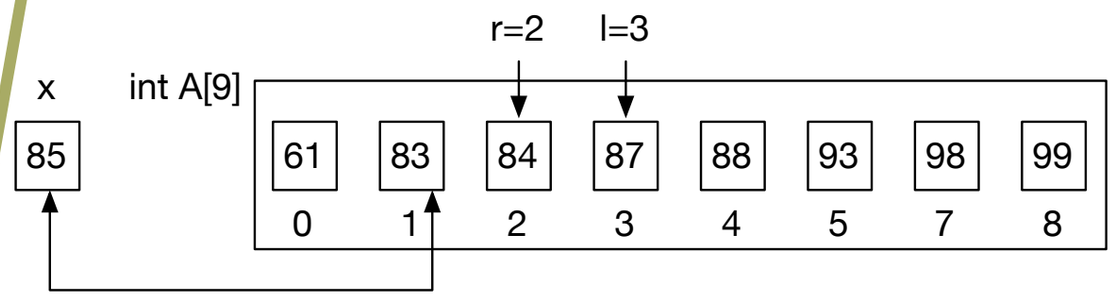
所以 找右半部分，调整左边界l=c+1



x == A[c]? false

x > A[c]? false

所以 找左半部分，调整左边界r=c–1



r 已经到了l的左边，不合逻辑

找不到要查找的数据，结束

$c=(l+r)/2$

折半查找

```
/*
数组名称：ArrayName
数组中变量的个数：ArraySize
待查找数据：x
*/
int l, r, c;
l = 0;
r = ArraySize - 1;
while(r >= l) {
    c = (l + r) / 2;
    if(ArrayName[c] == x) {
        break;
    } else if(ArrayName[c] > x)  {
        l = c + 1;
    } else {
        r = c - 1;
    }
}
if(r >= l) {
    printf("%d\n", c);
} else {
    printf("not found!\n");
}
```

向函数传递一维数组

- 需要传递什么?
- 全局变量的形式
- 参数形式
- 为数组中的变量赋值
- 输出数组中的变量
- 数组数据的处理
  - 排序、查找
  - 求和、平均数、最大数、最小数
- 函数能返回数组么?

整型数组排序函数

- /*
- 在一维整型数组中查找给定的数据
- 找到了给下标，找不到给-1（结果表示）
- 参数
  - 整型数组，名称 aArray; 大小 aSize（变量个数）
  - 待查找的数据，x，int
- 返回值
  - 下标/-1，int
- */
- **int findInArray(int x, int aArray[], int aSize);**

整型数组排序函数

```
int findInArray(int x, int aArray[], int aSize)
{
  int l, r, c;
  l = 0;
  r = aSize - 1;
  while(r >= l) {
    c = (l + r) / 2;
    if(aArray[c] == x) {
      break;
    } else if(aArray[c] > x) {
      r = c - 1;
    } else {
      l = c + 1;
    }
  }
  return (r >= l) ? c : -1;
}
```

多维数组

- 二维数组
  - 一维数组，数组中的变量还是一个一维数组
  - int ArrayA[9][9];
    - 9个变量的一维数组ArraytA
      - 这9个变量中的每个变量还是一9个整型变量的一维数组
  - int ArrayB[5][10];
    - 5个变量的一维数组ArraytB
    - 这5个变量中的每个变量都是一10个整型变量的一维数组
- 三维数组?

二维数组的定义和访问

- 定义
  - 数组基类型 数组名称[变量/元素个数][变量/元素个数];
    第1维(行)　　第2维(列)
  - int Array[2][2]={{1,2},{3,4}};
  - int Array[ ][2]={{1,2},{3,4}}; //与前一个等价
  - int Array[3][3]={1,2,3,4,5,6,7,8,9};//一般不建议使用
- 访问
  - 访问的是某个具体的变量
  - 用下标来确定
    - 数组名称[第1维的下标][第2维的下标]

二维数组

Array99	第0列	第1列	第2列	第3列	第4列	第5列	第6列	第7列	第8列
第0行	*Array99[0][0]* 1	*Array99[0][1]* 2	*Array99[0][2]* 3	4	5	6	7	8	*Array99[0][8]* 9
第1行	*Array99[1][0]* 2	4	6	8	10	12	14	16	18
第2行	*Array99[2][0]* 3	6	9	12	15	18	21	24	27
第3行	4	8	12	16	20	24	28	32	36
第4行	5	10	15	20	25	30	35	40	45
第5行	6	12	18	24	30	36	42	48	54
第6行	7	14	21	28	35	42	49	56	63
第7行	8	16	24	32	40	48	56	64	72
第8行	*Array99[8][0]* 9	18	27	36	45	54	63	72	*Array99[8][8]* 81

生成和输出九九乘法表

```
int Array99[9][9];
/* 按行按列顺序访问，即先访问低维（行），再顺序访问高维（列），[行][列] 确定数组中的具体变量 */
int i, j;
for(i = 0; i < 9; i++) {
  for(j = 0; j < 9; j++) {
    Array99[i][j] = i * j;
  }
}
```

```
for(i = 0; i < 9; i++) {
  for(j = 0; j < 9; j++) {
    if(j > 0) {
      printf("  ");
    }
    printf("%d*%d=%2d", i, j, Array99[i][j]);
  }
  printf("\n");
}
```

生成和输出九九乘法表

```
int Array99[9][9];
/* 按列按行顺序访问，即先访问高维（列），再顺序访问低维（行），[行][列] 确定数组中的具体变量 */
int i, j;
for(j = 0; j < 9; j++) {
  for(i = 0; i < 9; i++) {
    Array99[i][j] = i * j;
  }
}
```

```
for(i = 0; i < 9; i++) {
  for(j = 0; j < 9; j++) {
    if(j > 0) {
      printf("  ");
    }
    printf("%d*%d=%2d", i, j, Array99[i][j]);
  }
  printf("\n");
}
```

向函数传递二维数组

- 课本例8.2（P214）

- 习题8.2（4）（P222）

数组的初始化的简单用法

- int Array[5]={5};
- int Array[5]={1,2,3};
- int Array[10]={0};

- int Array[2][10]={0};
- int Array[2][4]={{1,2}, {1,2,3}};
- int Array[2][4]={{1,2,0,0}, {1,2,3,0}};

- static int Array[5]; /* 每个变量的值都会被初始化为0 */