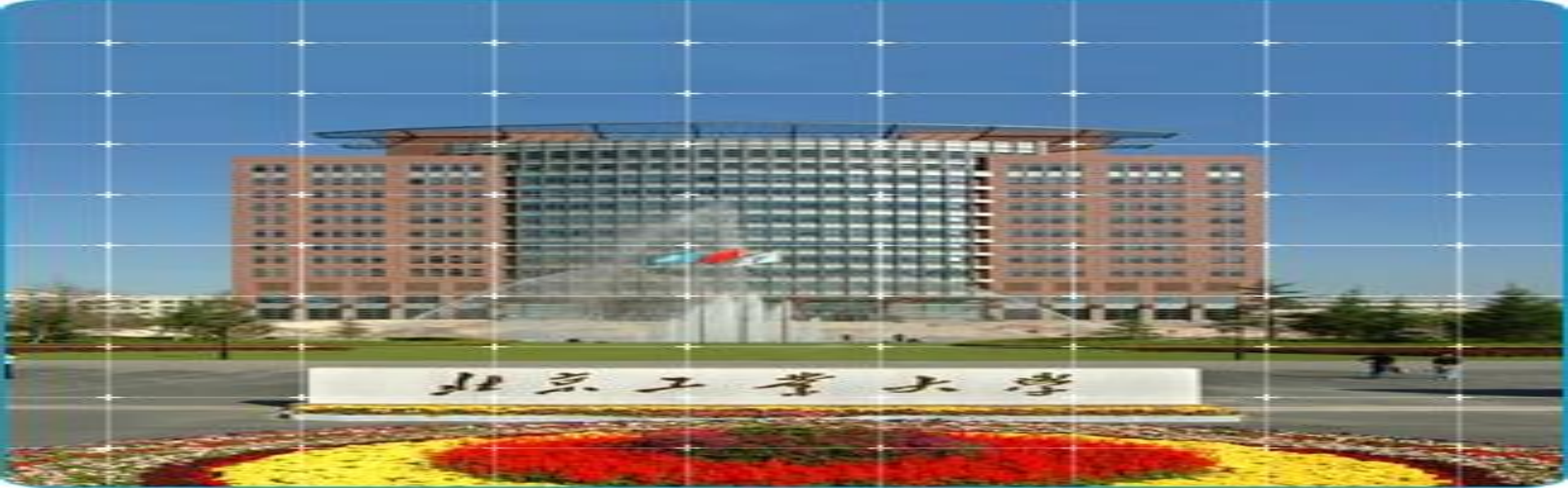


C语言程序设计

5.1 数组

- 了解数组含义及功能
- 理解数组变量在内存中的分配和使用方式
- 掌握数组的定义、初始化及引用方法
- 掌握字符数组及字符串的用法及基本操作

5.1.1 导例：如何存储和操作某班C语言课程的成绩

➢ 问题描述

一个班（假设有10名同学）C语言课程考试后，如何利用C程序来显示这些同学的成绩？如果有的同学的成绩统计错了，如何将它改正过来？如果有2个同学的成绩统计混淆了，如何将他们的成绩交换过来？

5.1.1 导例：如何存储和操作某班C语言课程的成绩

➢ 问题分析

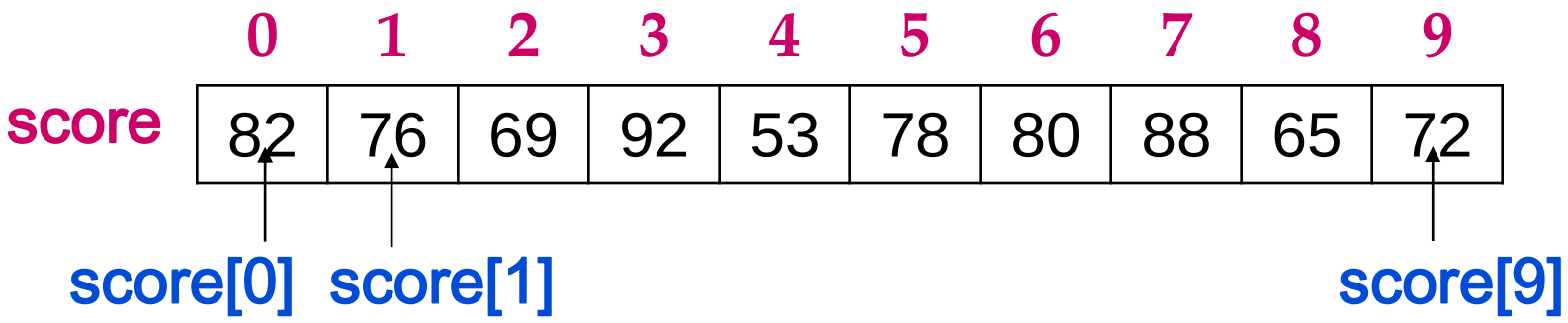
如何存储学生成绩？

- 变量
- 数组

5.1 数组

```
#include <stdio.h>
#define N 10
void main ( )
{       /*定义一维数组并初始化,长度必须为常量*/
        int score[N]={82,76,69,92,53,78,80,88,65,72};
        int i, t;
        for(i=0;i<N;i++)       /*显示所有学生的成绩*/
              printf("%d  ",score[i]);
        printf("\n");
        score[4]=60;     /*修改第5名同学的成绩*/
        t=score[1];    /*互换第2和第7名同学的成绩*/
        score[1]=score[6];
        score[6]=t;
        for(i=0;i<N;i++)       /*再次显示所有学生的成绩*/
              printf("%d  ",score[i]);
}
```

```
for(i=0;i<N;i++)
    printf("%d  ",score[i]);
```



- 数组：相同类型数据的有序集合，在内存中连续存放。
  - 由数组名和下标惟一地确定每个数组元素
  - 每个元素都属于同一类型
- 一批相同类型的变量使用同一个数组变量名，用下标来相互区分。
  - 优点：表述简洁，可读性高；便于使用循环结构

一维数组的定义和引用

1、定义

类型名　数组名[数组长度]

数组长度为常量

类型名：数组元素的类型

数组名：数组（变量）的名称，标识符

数组长度：常量表达式，给定数组的大小

```
int a[10];
```

定义一个含有10个整型元素的数组 a

```
char c[200];
```

定义一个含有200个字符元素的数组 c

```
float f[5];
```

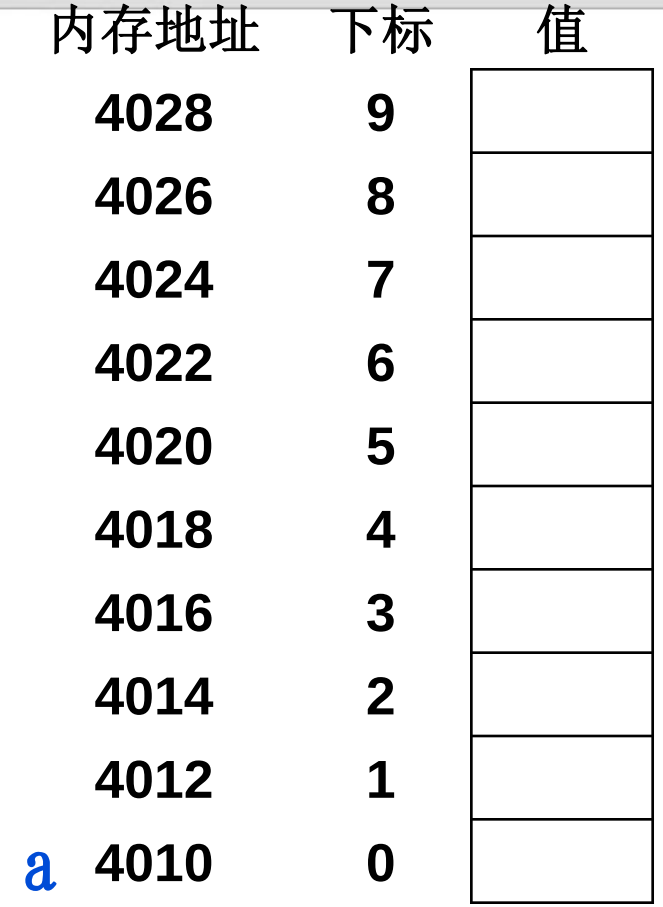
定义一个含有5个浮点型元素的数组 f

2、数组的内存结构

```
int a[10];
```

假设系统规定int类型占用2个字节，则对于数组a，其内存分配形式

只要知道了数组第一个元素的地址以及每个元素所需的字节数，其余各个元素的存储地址均可计算得到。



数组名是一个地址常量，存放数组内存空间的首地址。不允许被修改。

3、引用

- 先定义，后使用
- 只能引用单个的数组元素，不能一次引用整个数组

  数组名[下标]

  下标：整型表达式

  取值范围：[0，数组长度-1]

下标不要越界
不能使用a[10]

```
int  a[10];
```

10个元素：a[0]、a[1]、…… a[9]

- 数组元素的使用方法与同类型的变量相同

```
scanf("%d", &a[i]);
count[i]++;
temp = a[index]; a[index]= a[k]; a[k] = temp;
printf("%d ", a[i]);
```

区分数组的定义和数组元素的引用

定义数组

类型名　数组名[数组长度]

引用数组元素

数组名[下标]

数组长度为常量

下标不要越界

```
int a[10];
a[0] = a[9] = 0;
a[k] = temp;
```

一维数组的初始化

- 定义数组时，对数组元素赋初值

  类型名 数组名[数组长度] = {初值表};

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

  a[0]=1, a[1]=2, . . . … a[9]=10

- 数组的初始化

  数组如果没有初始化，所有元素为随机值

针对部分元素的初始化

```
int fib[20] = {0, 1};
```

fib[0] = 0, fib[1] = 1, 其余元素不确定

- 如果对全部元素都赋初值，可以省略数组长度

```
int a[ 10 ] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
```

建议不要省略数组长度

使用一维数组编程

数组和循环

```
for(i = 0; i < n; i++)
        printf("%d ", a[i]);
```

数组下标作为循环变量，通过循环，逐个处理数组元素

一维数组示例

例5-1 用数组计算fibonacci数列的前10个数，并按每行打印5个数的格式输出。1，1，2，3，5，……

例5-2 顺序查找法。输入5个整数，将它们存入数组a中，再输入1个数x，然后在数组中查找x，如果找到，输出相应的最小下标，否则，输出“Not Found”。

例5-3 输入n(n<10)，再输入n个数

(1) 输出最小值和它所对应的下标

(2) 将最小值与第一个数交换，输出交换后的n个数

例5-4 选择排序法。

例5-5 二分查找法。

例 5-1 计算fibonacci数列

用数组计算fibonacci数列的前10个数，并按每行打印5个数的格式输出。

1，1，2，3，5，8，13，……

用数组计算并存放fibonacci数列的前10个数

f[0] = f[1] = 1

f[n] = f[n-1] + f[n-2]　　2≤n≤9

例 5-1 源程序

```
#include <stdio.h>
int main(void)
{
    int i;
    int fib[10] = {1, 1};        /* 数组初始化 */
    for(i = 2; i < 10; i++)
        fib[i] = fib[i - 1] + fib[i - 2];
    for(i = 0; i < 10; i++) {
        printf("%6d", fib[i]);
        if((i + 1) % 5 == 0)    /* 5个数换行 */
            printf("\n");
    }
    return 0;
}
```

```
     1     1     2     3     5
     8    13    21    34    55
```

例5-2在数组中查找一个给定的数

输入5个整数，将它们存入数组a中，再输入1个数x，然后在数组中查找x，如果找到，输出相应的下标，否则，输出“Not Found”。

输入：2 9 8 1 9

9

输出：1

输入：2 9 8 1 6

7

输出：Not Found

例 5-2 源程序

```
#include <stdio.h>
int main(void)
{   int i, flag, x;
    int a[5];
    printf(“Enter 5 integers: ");
    for(i = 0; i < 5; i++)
        scanf("%d", &a[i]);
    printf(“Enter x: ");
    scanf("%d", &x);
    flag = 0;
    for(i = 0; i < 5; i++)
        if(a[i] == x){
            printf("Index is %d\n", i);
            flag = 1;
            break;
        }
    if(flag == 0)     printf("Not Found\n");
    return 0;
}
```

```
Enter 5 integers: 2 9 8 1 9
Enter x: 9
Index is 1
```

```
Enter 5 integers: 2 9 8 1 9
Enter x: 7
Not Found
```

例 5-2思考(1)

```
#include <stdio.h>
int main(void)
{   int i, flag, x;   int a[5];
    printf(“Enter 5 integers: ");
    for(i = 0; i < 5; i++)
        scanf("%d", &a[i]);
    printf(“Enter x: ");
    scanf("%d", &x);
    flag = 0;
    for(i = 0; i < 5; i++)
        if(a[i] == x){
            printf("Index is %d\n", i);
            flag = 1;
            break;
        }
    if(flag == 0)     printf("Not Found\n");
    return 0;
}
```

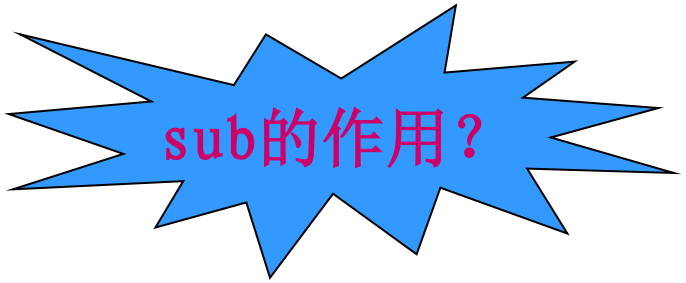
去掉break语句，结果？

```
Enter 5 integers: 2 9 8 1 9
Enter x: 9
Index is 1
Index is 4
```

例 5-2 思考(2)

```
#include <stdio.h>
int main(void)
{   int i, sub, x;
    int a[5];
    printf(“Enter 5 integers: ");
    for(i = 0; i < 5; i++)
        scanf("%d", &a[i]);
    printf(“Enter x: ");
    scanf("%d", &x);
    sub = -1;
    for(i = 0; i < 5; i++)
        if(a[i] == x)
            sub = i;
    if(sub != -1)  printf("Index is %d\n", sub);
    else printf("Not Found\n");
    return 0;
}
```

Enter 5 integers: 2 9 8 1 9
Enter x: 9
Index is 4



例 5-3 求最小值

```
#include <stdio.h>
int main(void)
{    int i, min, n;
     int a[10];
     printf(“Enter n: ");
     scanf("%d", &n);
     printf(“Enter %d integers: ", n);
     for(i = 0; i < n; i++)
          scanf("%d", &a[i]);
     min = a[0];
     for(i = 1; i < n; i++)
          if(a[i] < min)  min = a[i];
     printf("min is %d \n", min);
     return 0;
}
```

```
Enter n: 6
Enter 6 integers: 2  9  -1  8  1  6
min is -1
```

虽得到了最小值，但不能确定最小值所在下标！

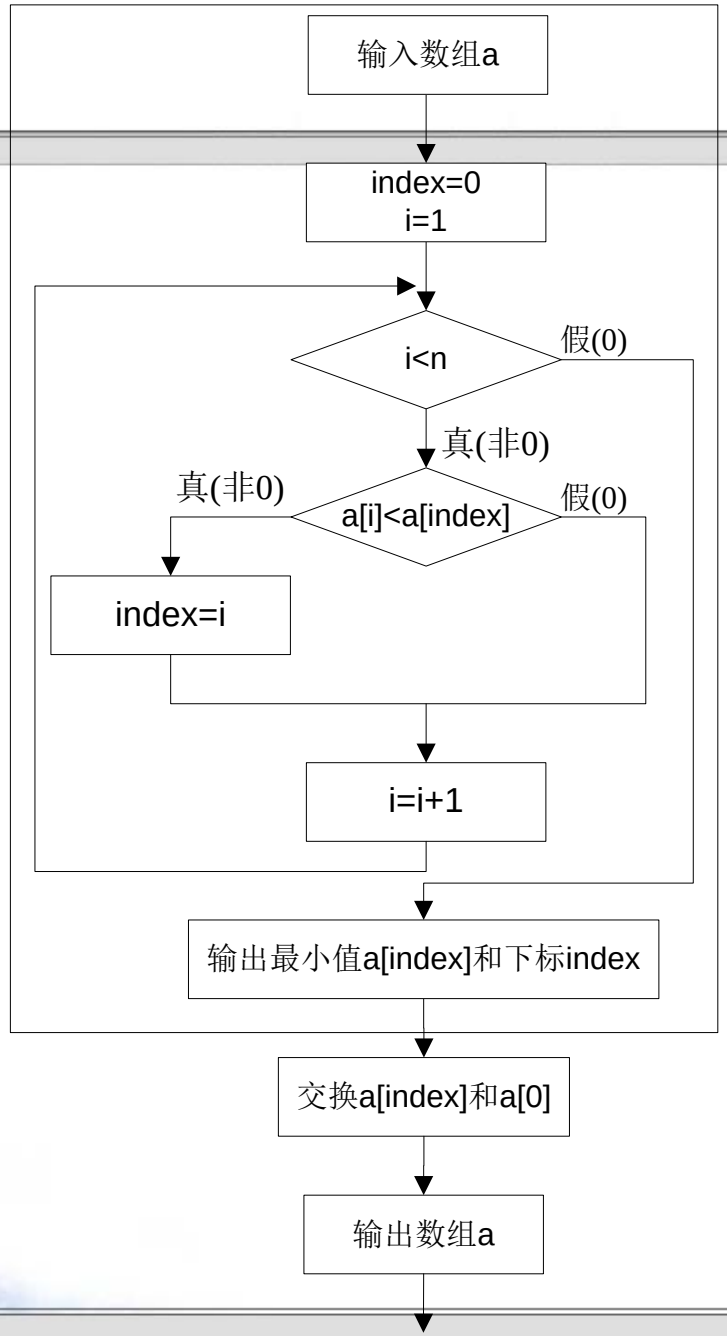
例 5-3(1) 求最小值及其下标

输入n(n<10)，再输入n个数，输出最小值和它所对应的下标。

用index记录最小值对应的下标

a[index]就是最小值

流程图


输入数组a
index=0
i=1
i<n
假(0)
真(非0)
真(非0)
a[i]<a[index]
假(0)
index=i
i=i+1
输出最小值a[index]和下标index
交换a[index]和a[0]
输出数组a

求最小值及下标

```
#include <stdio.h>
int main(void)
{    int i, index, n;
    int a[10];
    printf(“Enter n: ");
    scanf("%d", &n);
    printf(“Enter %d integers: ", n);
    for(i = 0; i < n; i++)
        scanf("%d", &a[i]);
    index = 0;
    for(i = 1; i < n; i++)
        if(a[i] < a[index])  index = i;
    printf("min is %d\tsub is %d\n", a[index], index);
    return 0;
}
```

```
Enter n: 6
Enter 6 integers: 2  9  -1  8  1  6
min is -1       sub is 2
```

例 5-5(2) 交换最小值

输入n(n<10)，再输入n个数,将最小值与第一个数交换，输出交换后的n个数。

用index记录最小值对应的下标

a[index]就是最小值

最小值与第一个数交换

a[index] <==> a[0]

导例5.1.2 选择法排序

利用选择排序方法将5.1.1导例中全班同学C语言课程成绩按照从低到高的顺序排列。

下标值	0	1	2	3	4	5	6	7	8	9
	82	76	69	92	53	78	80	88	65	72

第1次：	53	76	69	92	82	78	80	88	65	72
第2次：		65	69	92	82	78	80	88	76	72
第3次：			69	92	82	78	80	88	76	72
第4次：				72	82	78	80	88	76	92
第5次：					76	78	80	88	82	92
第6次：						78	80	88	82	92

选择法(1)

82 76 69 92 53 78 80 88 65 72(n=10)

10个数(a[0]~a[9])中找最小数，与a[0]交换

(1) 53 76 69 92 82 78 80 88 65 72 a[4] <==> a[0]

9个数(a[1]~a[9])中找最小数，与a[1]交换

(2) 53 65 69 92 82 78 80 88 76 72 a[8] <==> a[1]

8个数(a[2]~a[9])中找最小数，与a[2]交换

(3) 53 65 69 92 82 78 80 88 76 72 a[2] <==> a[2]

7个数(a[3]~a[9])中找最小数，与a[3]交换

(4) 53 65 69 72 82 78 80 88 76 92 a[9] <==> a[3]

6个数(a[4]~a[9])中找最小数，与a[4]交换

(5) 53 65 69 72 76 78 80 88 82 92 a[8] <==> a[4]

5个数(a[5]~a[9])中找最小数，与a[5]交换

(6) 53 65 69 72 76 78 80 88 82 92 a[5] <==> a[5]

选择法(1)

82 76 69 92 53 78 80 88 65 72(n=10)

4个数(a[6]~a[9])中找最小数，与a[6]交换

(7) 53 65 69 72 76 78 80 88 82 92 a[6] <==> a[6]

3个数(a[7]~a[9])中找最小数，与a[7]交换

(8) 53 65 69 72 76 78 80 82 88 92 a[8] <==> a[7]

2个数(a[8]~a[9])中找最小数，与a[8]交换

(9) 53 65 69 72 76 78 80 82 88 92 a[8] <==> a[8]

选择法(2)

(1) 10个数 (a[0]~a[9]) 中找最小数，与 a[0] 交换

(2) 9个数 (a[1]~a[9]) 中找最小数，与 a[1] 交换

(3) 8个数 (a[2]~a[9]) 中找最小数，与 a[2] 交换

(4) 7个数 (a[3]~a[0]) 中找最小数，与 a[3] 交换

---

(1) n个数 (a[0]~a[n-1]) 中找最小数，与 a[0] 交换

(2) n-1个数 (a[1]~a[n-1]) 中找最小数，与 a[1] 交换

……

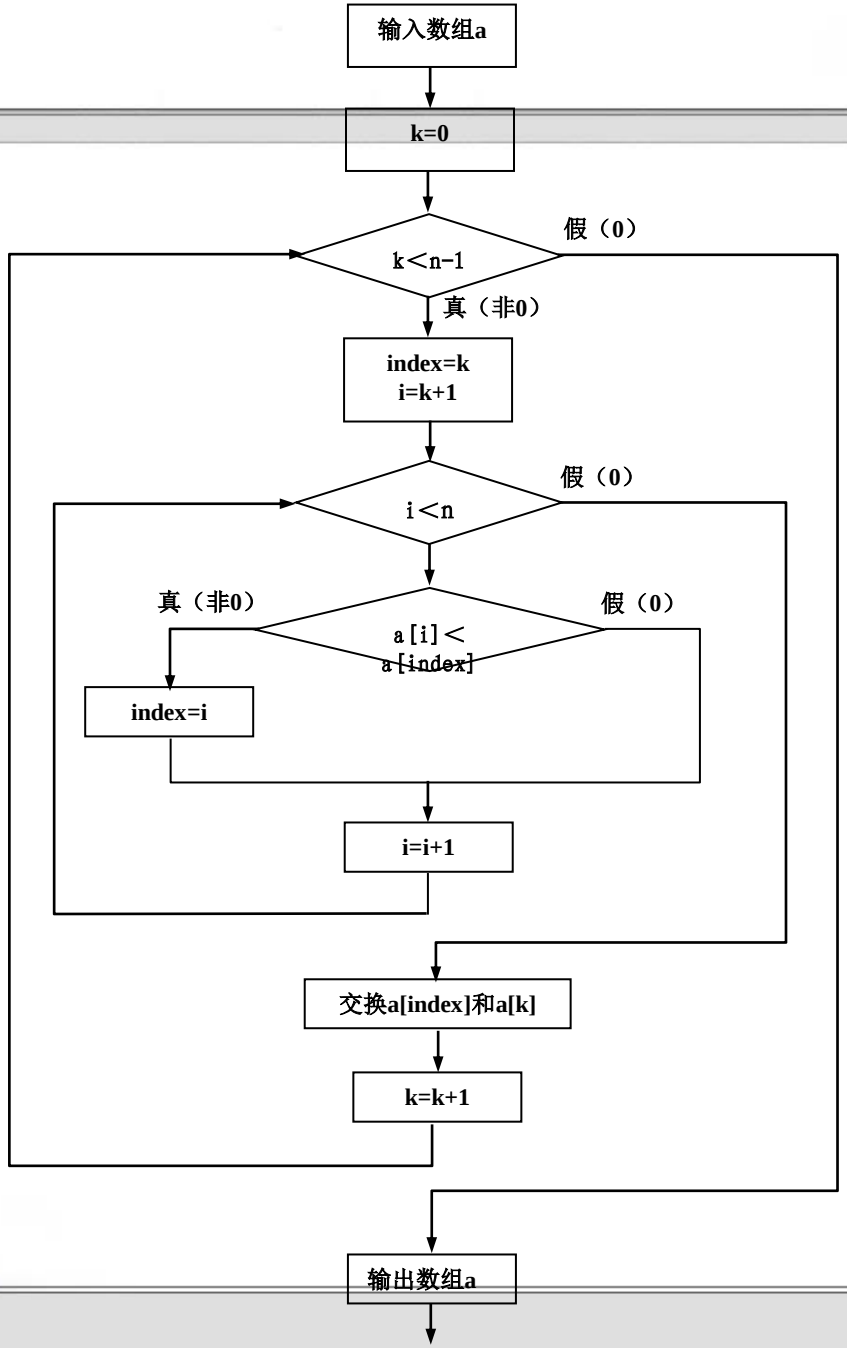
(n-1) 2个数 (a[n-2]~a[n-1]) 中找最小数，与 a[n-2] 交换

n个数重复n-1次

流程图

外循环控制：

n 个数选择排序共需要n-1次



内循环控制：

在下标范围[k, n-1]内找最小值所在位置index

选择法排序（程序段）

```
for( i =0; i < n -1; i ++)
    {
        k = i;              /*查找最小元素的下标*/
        for( j = i +1; j < n; j ++)
            if( a[j]< a[k])
                k = j;
        if( k != i)         /*将a[k]和a[i]交换*/
        {
            t = a[k];
            a[k] = a[i];
            a[i] = t;
        }
    }
```

排序前学生成绩：

82 76 69 92 53 78 80 88 65 72

排序后学生成绩：

53 65 69 72 76 78 80 82 88 92

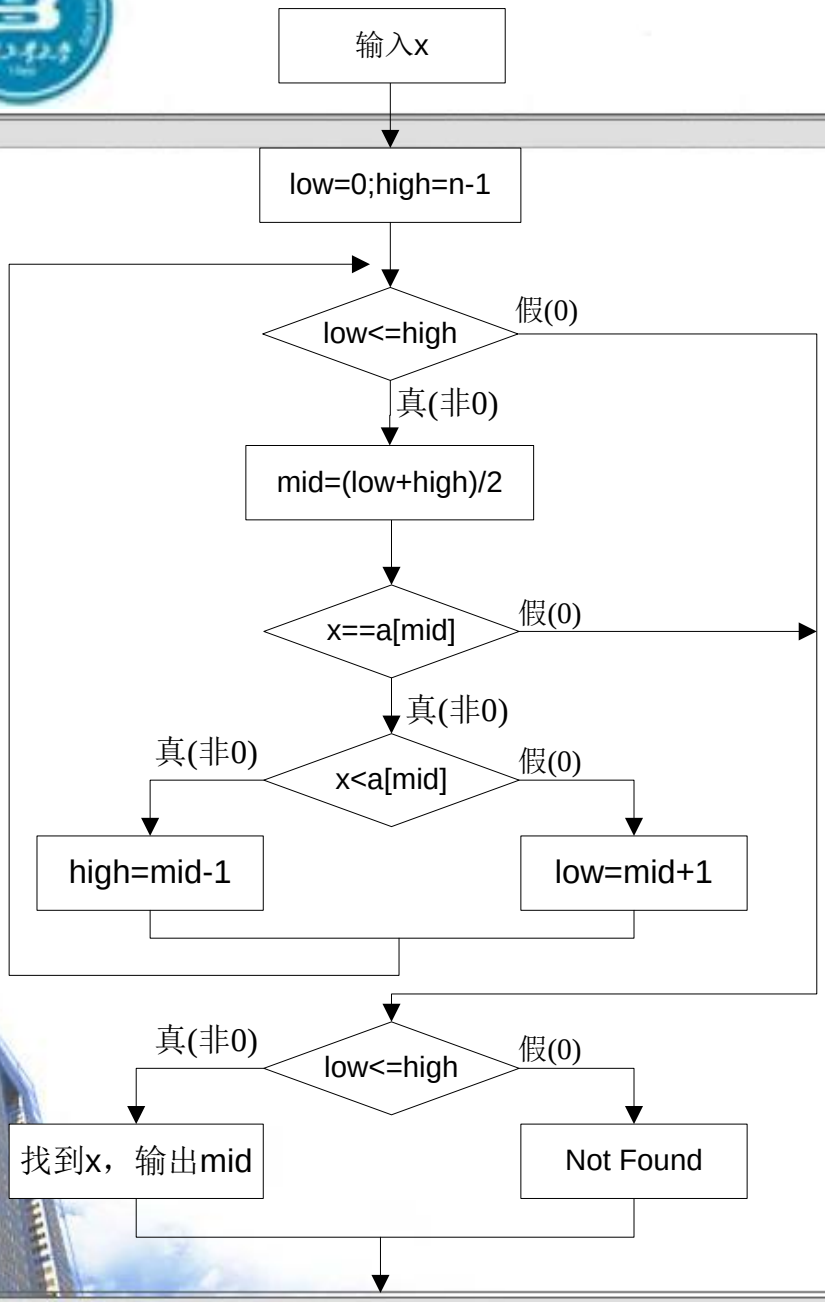
5.1.4导例：二分查找算法

已知某个成绩，查找该成绩在班级中的排名，即返回该成绩下标。

5.1.3导例顺序查找是最简单明了的一种，其查找过程就是对数组元素从头到尾的遍历过程。但是一旦数组元素量很大的情况下，其查找的效率不高。

二分查找是查找效率较高的一种，但前提是数组元素必须是有序的。

二分查找流程图


输入x
low=0;high=n-1
low<=high
假(0)
真(非0)
mid=(low+high)/2
x==a[mid]
假(0)
真(非0)
真(非0)
x<a[mid]
假(0)
high=mid-1
low=mid+1
真(非0)
low<=high
假(0)
找到x，输出mid
Not Found

二分法查找(程序段)

```
int binarySearch(int a[], int n, int x)
{
    int low = 0;
    int high = n - 1;
    while(low <= high)
    {
        int mid = (low + high)/2;
        if(x==a[mid])      return mid;      /*mid为x在数组中的下标*/
        else if(x<a[mid])      /*在左半边*/
            high = mid - 1;
        else                        /*在右半边*/
            low = mid + 1;
    }
    return -1;    /*没找到*/
}
```

例 统计数字字符个数

输入一个以回车结束的字符串(少于80个字符)，统计其中数字字符的个数。

分析：

数组长度取上限80

以 '\n' 做为输入结束符

```
Enter a string: It's 512
count = 3
```

```c
#include <stdio.h>
int main(void)
{   int count, i;
    char str[80];
    printf( "Enter a string: ");
    i = 0;
    while((str[i] = getchar( )) != '\n')
        i++;
    str[i] = '\0';
    count = 0;
    for(i = 0; str[i] != '\0'; i++)
        if(str[i] <= '9' && str[i] >= '0')
            count++;
    printf("count = %d\n", count);
    return 0;
}
```

一维字符数组

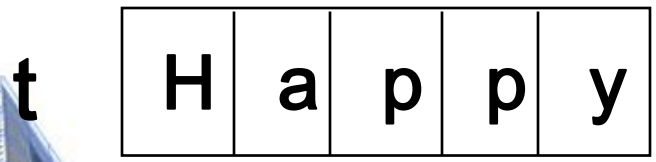
- 字符串的存储和运算可以用一维字符数组实现
- 一维字符数组的定义、引用、初始化与其他类型的一维数组一样。

```
char   str[80];
```

定义一个含有80个字符型元素的数组str

```
char t[5]={'H', 'a', 'p', 'p', 'y'};
```

初始化数组 t



输出数组 t 的所有元素

```
for(i=0; i<5; i++)
    putchar( t[i] );
```

字符串的存储

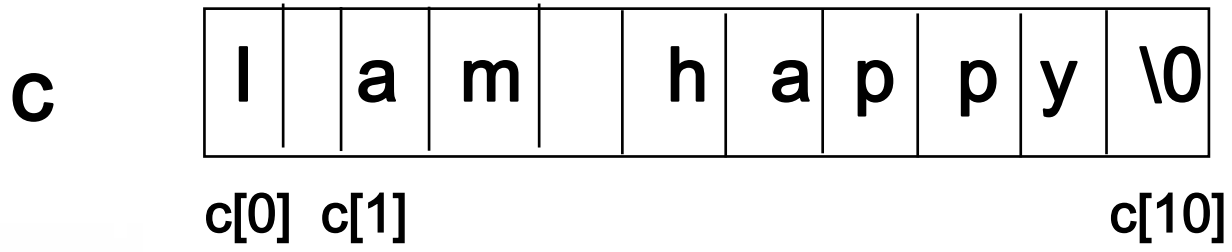
char c[11]=“I am happy”;

字符串遇 '\0' 结束

第一个 ‘\0’ 前面的所有字符和 ‘\0’ 一起构成了字符串 “I am happy”

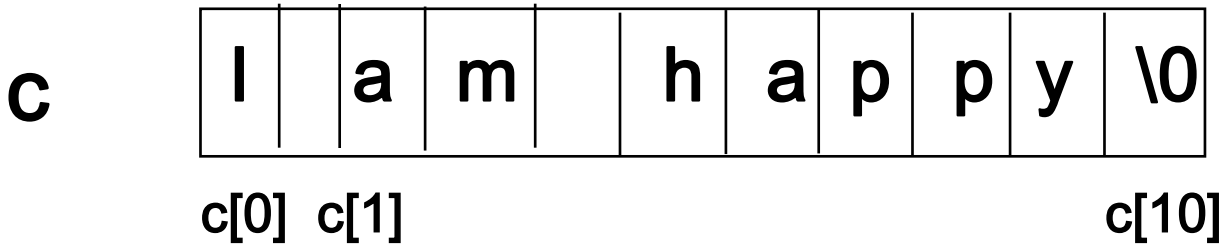
'\0' 之后的其他数组元素与该字符串无关

字符串由有效字符和字符串结束符 '\0' 组成



一维字符数组

```
char c[11]=" I am happy" ;
```



当用字符串初始化字符数组时，必须在定义字符数组的同时进行初始化，而不能定义完字符数组后再将字符串赋值给字符数组

```
char c[11];
c="I am happy"; /*错误*/
```

字符串

字符串常量

用一对双引号括起来的字符序列

一个字符串结束符 '\0'

"Happy"

字符串结束符

6个字符 'H' 'a' 'p' 'p' 'y' '\0'

有效字符

字符串的有效长度：有效字符的个数

字符串与一维字符数组

字符串：一个特殊的一维字符数组

- 把字符串放入一维字符数组（存储）
- 对字符串的操作 ===> 对字符数组的操作

2. 对字符串的操作

- 把字符串放入一维字符数组（存储）
- 对字符串的操作 ===> 对字符数组的操作

普通字符数组：数组元素的个数是确定的，一般用下标控制循环

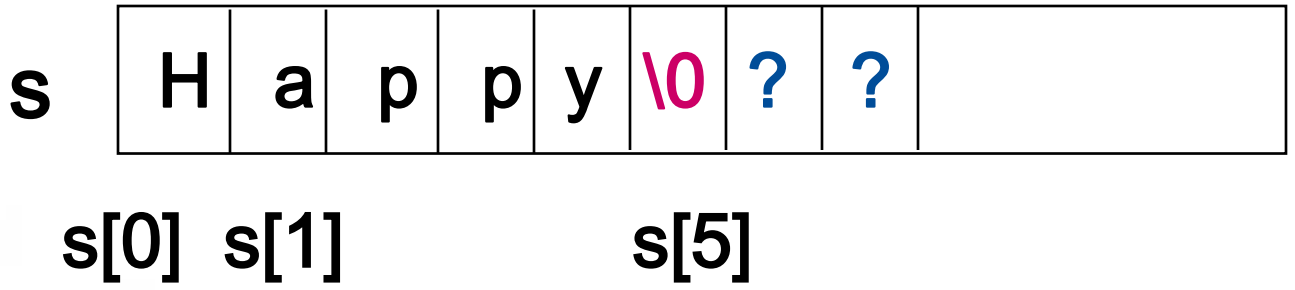
字符串：没有显式地给出有效字符的个数，只规定在字符串结束符 '\0' 之前的字符都是字符串的有效字符，一般用结束符 '\0' 来控制循环

循环条件：`s[i] != '\0'`

输出字符串

```
for(i = 0; s[i] != '\0'; i++)
    putchar(s[i]);
for(i = 0; i < 80; i++)
    putchar(s[i]);
for(i = 0; i < len; i++)
    putchar(s[i]);
```

输出？



3. 字符串的存储－赋值和输入

- 把字符串放入一维字符数组（存储）
- 对字符串的操作 ===> 对字符数组的操作

存储

数组初始化

```
char s[6]= "Hello";
```

区分"a" 和 'a'

赋值

```
s[0]='a'; s[1]='\0';
```

或者

```
char s[6]= "a";
```

"a"	2 个字符 'a' 和 '\0'
'a'	1 个字符常量

字符串的输入

字符串的存储

- 字符数组初始化： char s[6] = "Happy";
- 赋值：s[0] = 'a'; s[1] = '\0';
- 输入

  '\0' 代表空操作，无法输入

  输入时，设定一个输入结束符

  将输入结束符转换为字符串结束符 '\0'

5.1.6 统计字符串中字符的信息

- 问题描述

  从键盘输入一个字符串，统计出该字符串中数字、大写字母、小写字母以及其他字符的数量。

- 问题分析

  如何表示字符串？

```
#include <stdio.h>
#include <string.h>
void main ( )
{   char s[50];
    int dig=0,up=0,lw=0,other=0;
    int i,n;
    gets(s);                    /*通过键盘给s赋值*/
    n=strlen(s);                /*求s的长度并赋值给n*/
    for(i=0;i<n;i++){
        if(s[i]>='0' && s[i]<='9')              /*求 s中数字的个数*/
            dig++;
        else if(s[i]>='A' && s[i]<='Z')         /*求 s中大写字母的个数*/
            up++;
        else if(s[i]>='a' && s[i]<='z')         /*求 s中小写字母的个数*/
            lw++;
        else  other++;                  /*求 s中其他字符的个数*/
    }
    printf("dig= %d up= %d lw= %d other= %d\n",dig,up,lw,other);
}
```

Hi, Mr Li, Are you 25 years old?

dig= 2 up= 4 lw= 16 other= 10