学习目标

- 理解一维数组、二维数组的特点
- 能够定义和引用一维数组、二维数组

7.1 排序问题

输入一个正整数n （1<n≤10），再输入n个整数，用选择法将它们从小到大排序后输出。

1. 程序解析
2. 一维数组的定义和引用
3. 一维数组的初始化
4. 使用一维数组编程

7.1.1 程序解析—排序

```
#include <stdio.h>
int main(void)
{   int i, index, k, n, temp;
    int a[10];          /* 定义1个数组a，它有10个整型元素*/
    printf(“Enter n: ");
    scanf("%d”, &n);
    printf(“Enter %d integers: ",
    for(i = 0; i < n; i++)
        scanf("%d", &a[i]);
    for(k = 0; k < n-1; k++){   /* 对n个数排序 */
        index = k;
        for(i = k + 1; i < n; i++)
            if(a[i] < a[index])  index = i;
        temp = a[index]; a[index] = a[k]; a[k] = temp;
    }
    printf("After sorted: ");
    for(i = 0; i < n; i++)      /* 输出n个数组元素的值 */
        printf("%d ", a[i]);
    return 0;
}
```

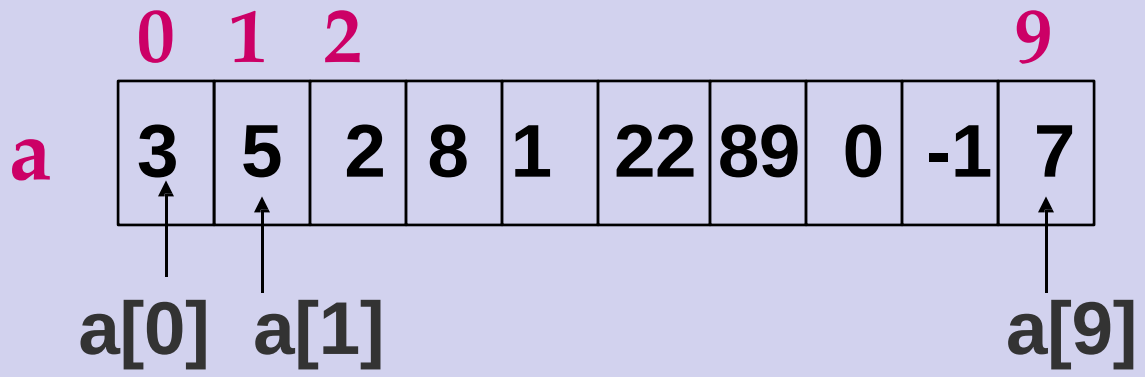
Enter n: 10

Enter 10 integers: 3 5 2 8 1 22 89 0 -1 7

After sorted: -1 0 1 2 3 5 7 8 22 89

数组

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

7.1.2 一维数组的定义和引用

1、定义

类型名 数组名[数组长度]

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、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]

下标不要越界
不能使用a[10]

下标：整型表达式

取值范围：[0，数组长度-1]

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]);
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]

数组长度为常量

引用数组元素

数组名[下标]

下标不要越界

```
int a[10];

a[0] = a[9] = 0;

a[k] = temp;
```

7.1.3 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

- 静态数组、动态数组的初始化

  static int b[5] = {1, 2, 3, 4, 5};

  静态存储的数组如果没有初始化，所有元素自动赋0

  static int b[5];

  动态存储的数组如果没有初始化，所有元素为随机值

  auto int c[5];

针对部分元素的初始化

static int b[5] = {1, 2, 3};

b[0] = 1, b[1] = 2, b[2] = 3, b[3] = 0, b[4] = 0

auto 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

7.1.4 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 7-2 用数组计算fibonacci数列的前20个数，并按每行打印5个数的格式输出。1, 1, 2, 3, 5, ……

例7-3 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”。

例 7-4 输入n(n<10)，再输入n个数

⑴求最小值

⑵求最小值和它所对应的下标

⑶将最小值与第一个数交换，输出交换后的n个数

例 7-1 输入n(n<10)，再输入n个数，用选择法将它们从小到大排序后输出。

例 7-2 计算fibonacci数列

用数组计算fibonacci数列的前20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20个数

f[0] = f[1] = 1

f[n] = f[n-1] + f[n-2]　2≤n≤19

例 7-2 源程序

```
#include <stdio.h>
int main(void)
{
    int i;
    int fib[20] = {1, 1};   /* 数组初始化 */

    for(i = 2; i < 20; i++)
        fib[i] = fib[i - 1] + fib[i - 2];

    for(i = 0; i < 20; 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**
89	**144**	**233**	**377**	**610**
987	**1597**	**2584**	**4181**	**6765**

例7-3 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 9 6

9

输出：1

输入：2 9 8 9 6

7

输出：Not Found

例 7-3 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 7-3 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;
}
```

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

例 7-3 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", i);
    else printf("Not Found\n");
    return 0;
}
```

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

例 7-4(1) 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

以上内容仅为本文档的试下载部分，为可阅读页数的一半内容。如要下载或阅读全文，请访问：https://d.book118.com/468013013053006047