一维数组与字符串的指针实现

2022 年 8 月 16 日

1. 将数组的第一个元素和第二个元素互换位置，第三个和第四个元素互换位置，第五个和第六个元素互换位置……然后输出。要求数组的长度为 12，元素由用户输入，并且都是整数。

```c
#include <stdio.h>
int main()
{
    int a[12];
    for(int i = 0; i < 12; i++)
    {
        printf("Input the %d-th element:\n", i + 1);
        // scanf后面的参数，本质上是内存地址，指明了输入的数据即将写入的位置
        scanf("%d", a + i); // 数组名a自动转化为首元素地址，加i就是i号元素的地址
    }

    printf("The input array is below:\n");
    for(int i = 0; i < 12; i++)
    {
        printf("%d\t", *(a + i)); //
            数组a的i号元素的地址，进行*运算后，得到该地址所存储的数据，*(a+i)相当于a[i]
    }
    printf("\n");

    for(int i = 0; i <= 10; i += 2)
    {
        // 交换
        int temp = *(a + i);
        *(a + i) = *(a + i + 1);
        *(a + i + 1) = temp; // 可见，*既可以用于读取也可以用于写入
```

```
    }

    printf("The output array is below:\n");
    for(int i = 0; i < 12; i++)
    {
        printf("%d\t", *(a + i));
    }
    printf("\n");
    return 0;
}
```

2. 设有一个数组，长度为 12，并且数组元素由用户输入。现在要求，把 7 号位置的元素取出来，放到 0 号位置，并且把原来的 1 号到 6 号位置的元素相应后移，然后输出调整后的数组。例如：当输入数组为 1,2,3,-4,5, 6, 7, 8, 9, 10,-11,12 的时候，输出数组为 8,1,2,3,-4,5,6,7,9,10,-11,12。

```
#include <stdio.h>
int main()
{
    int a[12];
    //
        注意到数组名a自动转化为首元素地址，a+12表示数组的12号单元所在位置（恰好越界的位置）
    for(int *addr = a; addr < a + 12; addr++)
    // addr<a+12表示数组不越界
    // addr++表示指向下一个元素
    {
        printf("Input the %d-th integer:\n", addr - a + 1); //
            地址（指针减法），就看相差多少个元素的位置
        scanf("%d", addr); //
            scanf的后面参数，本质上是地址，指明了输入的数据即将写入的位置
    }

    printf("The input array is:\n");
    for(int *addr = a; p < a + 12; addr++)
    // p<a+12表示在数组范围内
    // addr++表示指向下一个元素
    {
        printf("%d\t", *addr);
    }
```

```
    printf("\n");

    // 移动元素
    int temp = *(a + 7);
    for(int i = 7; i > 0; i--)
    {
        *(a + i) = *(a + i - 1);
    }
    (*a) = temp;

    printf("The output array is:\n");
    for(int *addr = a; addr < a + 12; addr++)
    {
        printf("%d\t", *addr);
    }
    printf("\n");

    return 0;
}
```

3. 设有一个数组，长度为 12，并且数组元素由用户输入。现在要求，把 7 号位置的元素取出来，放到 11 号位置，并且把原来的 8 号到 11 号位置的元素相应前移，然后输出调整后的数组。例如：当输入数组为 1,2,3,-4,5, 6, 7, 8, 9, 10,-11,12 的时候，输出数组为 1,2,3,-4,5, 6, 7, 9, 10,-11,12, 8。

```
#include <stdio.h>
int main()
{
    int a[12];
    for(int i = 0; i < 12; i++)
    {
        printf("Input the %d-th integer:\n", i + 1);
        scanf("%d", a + i);
    }

    printf("The input array is:\n");
    for(int i = 0; i < 12; i++)
    {
        printf("%d\t", a[i]);
```

```
    }
    printf("\n");

    // 移动元素
    int temp = *(a + 7);
    for(int i = 7; i < 11; i++)
    {
        *(a + i) = *(a + i + 1);
    }
    *(a + 11) = temp;

    printf("The output array is:\n");
    for(int i = 0; i < 12; i++)
    {
        printf("%d\t", a[i]);
    }
    printf("\n");

    return 0;
}
```

4. 输入 5 个整数，求其中的最大值和最小值，并把这些整数按照与输入相反的顺序输出。

```
#include <stdio.h>
int main()
{
    int a[5];
    for(int i = 1; i <= 5; i++)
    {
        printf("Input %d-th integer:\n", i);
        scanf("%d", a + i - 1); // 相当于scanf("%d", &a[i-1]);
    }

    // 数组名a自动转化为首元素地址，因此(*a)表示首元素
    int max = (*a), min = (*a);
    // a+5表示数组a的5号元素所在的地址
    for(int *p = a; p < a + 5; i++) // p<a+5表示不越界
    {
        if((*p) > max)
```

```
        {
            max = (*p);
        }
        if((*p) < min)
        {
            min = (*p);
        }
    }
    printf("The maximum is: %d.\n", max);
    printf("The minimum is: %d.\n", min);

    int b[5] = {0};
    for(int i = 0; i < 5; i++)
    {
        *(b + 4 - i) = *(a + i); // 把元素从数组a复制到数组b
    }

    for(int i = 0; i < 5; i++)
    {
        printf("%d\t", *(b + i));
    }
    printf("\n");
    return 0;
}
```

5. 设一个数组包含 12 个元素，每一个元素都由用户输入。返回并输出最大元素所在的位置的下标和最小元素所在的位置的下标。如果出现多个并列最大元素或者并列最小元素，只取它们第一次出现的位置的下标。

```
#include <stdio.h>
int main()
{
    int a[12];
    for(int *addr = 0; addr < a + 12; addr++)
    {
        printf("Input the %d-th element:\n", addr - a + 1);
        // scanf的后面参数的本质是地址
        scanf("%d", addr); // 用户输入的数据将写入到地址为addr的内存单元中
    }
```

```
11
12      printf("The input array is:\n");
13      for(int *addr = a; addr < a + 12; addr++)
14      {
15          printf("%d\t", *addr);
16      }
17      printf("\n");
18
19      int maxi_at_loc = 0, mini_at_loc = 0; // 先假定最大值和最小值都在0号单元
20      for(int i = 1; i < 12; i++)
21      {
22          if(*(a + i) > *(a + maxi_at_loc)) // 找到更大的
23          {
24              maxi_at_loc = i; // 更新下标
25          }
26          if(*(a + i) < *(a + mini_at_loc)) // 找到更小的
27          {
28              mini_at_loc = i; // 更新下标
29          }
30      }
31
32      printf("The first maximum element is at Location %d and its value is %d\n",
            maxi_at_loc, *(a + maxi_at_loc));
33      printf("The first minimum element is at Location %d and its value is %d\n",
            mini_at_loc, *(a + mini_at_loc));
34
35      return 0;
36  }
```

6. 设一个数组包含 12 个元素，每一个元素都由用户输入。先把数组的最小元素与第一个元素交换，然后把数组的最大元素与最后一个元素交换。如果出现多个并列最大元素或者并列最小元素，只取第一个最大或者最小元素来交换。

```
1   #include <stdio.h>
2   int main()
3   {
4       int a[12];
5       for(int i = 0; i < 12; i++)
6       {
```

```
7           printf("Input the %d-th element:\n", i + 1);
8           int *addr = &(a[i]); // &在这里是取值运算符，就是获取a[i]的地址
9           scanf("%d", addr); // scanf的后面参数应该是地址
10      }
11
12      printf("The input array is:\n");
13      for(int i = 0; i < 12; i++)
14      {
15          printf("%d\t", a[i]);
16      }
17      printf("\n");
18
19      int mini_at_loc = 0; // 假定最小元素就在0号单元
20      for(int i = 1; i < 12; i++)
21      {
22          if(*(a + i) < *(a + mini_at_loc))
23          {
24              mini_at_loc = i;
25          }
26      }
27
28      // 交换
29      int temp = (*a);
30      (*a) = *(a + mini_at_loc);
31      *(a + mini_at_loc) = temp;
32
33      int maxi_at_loc = 0; // 假定最大元素在0号单元
34      for(int i = 1; i < 12; i++)
35      {
36          if(*(a + i) > *(a + maxi_at_loc))
37          {
38              maxi_at_loc = i;
39          }
40      }
41
42      // 交换
43      temp = *(a + 11);
44      *(a + 11) = *(a + maxi_at_loc);
45      *(a + maxi_at_loc) = temp;
```

```
46
47      printf("The adjusted array is:\n");
48      for(int i = 0; i < 12; i++)
49      {
50          printf("%d\t", a[i]);
51      }
52      printf("\n");
53
54      return 0;
55  }
```

7. 输入一个正整数，求它用到的不同数字（0-9）出现的次数。例如给定 n=100311，那么它有 2 个 0，3 个 1 和 1 个 3。

```
1   #include <stdio.h>
2   int main()
3   {
4       printf("Input an integer:\n");
5       int n = 0;
6       scanf("%d", &n);
7       int occur_times[10] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
8       while(n != 0)
9       {
10          int r = n % 10;
11          (*(occur_times + r))++; // 注意小括号指定了运算优先级
12          n /= 10;
13      }
14      printf("statistics info:\n");
15      for(int i = 0; i < 10; i++)
16      {
17          if(*(occur_times + i) != 0)
18          {
19              printf("%d: %d\n", i, *(occur_times + i));
20          }
21      }
22      return 0;
23  }
```

8. 输入一个字符串，然后程序按逆序输出它。

```c
#include <stdio.h>
#include <string.h>
int main()
{
    char str[1024];
    printf("Input a string:\n");
    gets(str);
    // --p即p=p-1，也就是指针前移一个单元
    for(int *p = str + strlen(str) - 1; p >= str; --p)
    {
        putchar(*p); // *号运算符获取了指针所指的对象
    }
    putchar('\n');
    return 0;
}
```

9. 输入一个字符串，将串中大写字母变为小写，小写字母变为大写，其他符号不变。

```c
#include <stdio.h>
#include <string.h>

#define MAX_LEN 1024
const int DIFF_BETWEEN_UPPER_AND_LOWER = 'a' - 'A';

int main()
{
    printf("Please input a string:\n");
    char str[MAX_LEN];
    gets(str);
    char *p = str; // p指向了str的首元素
    // 注意到0，0.0，'\0'的都代表假
    // 注意到字符串末端的字符为'\0'，一旦p指向这个字符，(*p)的值为0
    while(*p) // p未指向字符串的末端
    {
        if((*p) >= 'A' && (*p) <= 'Z') // p指向大写字母
        {
            (*p) += DIFF_BETWEEN_UPPER_AND_LOWER; // (*p)的值增大，得到了对应的小写字母
        }
```

```
        else if((*p) >= 'a' && (*p) <= 'z') // p指向小写字母
        {
            (*p) -= DIFF_BETWEEN_UPPER_AND_LOWER; // 得到了对应的大写字母
        }
        p++; // 后移一个单元，即一个字节
    }
    printf("After alternating cases, the input string has become: %s\n", str);
    return 0;
}
```

10. 求一个输入字符串的有效长度。如果字符串不含有空格，则有效长度为它本身地长度; 如果字符串含有空格，则有效长度为第一个空格之前的字符串（不包括空格本身）的长度。

```
#include <stdio.h>
#define MAX_LEN 1024
int main()
{
    char str[MAX_LEN];
    printf("Input a string:\n");
    gets(str);
    char *p = str; // 数组名str自动转化为首元素地址，因此p被赋值为字符串首元素地址
    while((*p) != '\0' && (*p) != ' ') // p为指向字符串末端，并且未指向空格
    {
        p++; // 后移一个元素（字符）
    }
    printf("The valid len is: %d.\n", p - str); //
        指针相加得到相差多少个单元（字符），也就是题意要求的长度
    return 0;
}
```