数组与字符串综合

2022 年 8 月 16 日

1. 编写一个程序，计算一个整型数组中最大的和最小的数之间的差值。假设数组的长度为 12，元素由用户输入。

```c
#include <stdio.h>
int main()
{
    int a[12] = {0};
    for(int i = 0; i < 12; i++)
    {
        printf("Please input the %d-th element:\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");

    int maxi = a[0], mini = a[0]; // 假定首元素为最大和最小值
    for(int i = 1; i < 12; i++)
    {
        if(a[i] < mini)
        {
            mini = a[i];
        }

        if(a[i] > maxi)
```

```
        {
            maxi = a[i];
        }
    }

    int max_diff = maxi - mini;

    printf("max_diff: %d\n", max_diff);

    return 0;
}
```

2. 随机生成 10 个互不相同的两位正整数，并输出其中的素数。

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
int main()
{
    int a[10];
    int i, j, s;
    srand((unsigned)time(NULL)); // 布置随机数种子
    for(i = 0; i < 10; i++)
    {
        // 以下的rand()将根据srand()布置的种子，产生一个（伪）随机正整数
        a[i] = rand() % 90 + 10; //
            rand()%90得到0~89范围内的数，加10就恰好包含全部两位正整数

        // 检查是否之前已经生成过
        for(j = 0; j < i; j++)
        {
            if(a[j] == a[i]) // 出现过
            {
                i--; // 不算数
                break; // 跳出内层循环
            }
        }
    }
```

```
    printf("The 10 input numbers:\n");
    for(i = 0; i < 10; i++)
    {
        printf("%d\t", a[i]);
    }
    printf("\n");

    for(i = 0; i < 10; i++)
    {
        s = 2;
        // 以下判定素数的算法利用了技巧，只要枚举到平方根（向下取整）即可
        while(s <= (int)sqrt(a[i]) && a[i] % s != 0)
        {
            s++;
        }
        // 退出循环有两种情况，一是找到约数，二是已经枚举到超出平方根

        // 以下情况是说枚举到超出平方根
        if(s > (int)sqrt(a[i])) // 可判定是素数
        {
            printf("%d is a prime number.\n", a[i]);
        }
    }
    return 0;
}
```

3. 输入一个十进制整数，转化为 d 进制表示（d 大于或等于 2，小于或等于 9）。

```
#include <stdio.h>

int main()
{
    // 输入十进制数
    int dec_num = 0;
    printf("Input a decimal number:\n");
    scanf("%d", &dec_num);

    // 输入进制数
    int base = 0;
```

```c
    printf("Input the base for a counting system (must between 2 and 9):\n");
    scanf("%d", &base);

    // 进制数不符合要求
    if(base > 9 || base < 2)
    {
        printf("The base is not between 2 and 9.\n");
        return 0;
    }

    int derived_bits[1024] = {0}; // 用于保存余数
    int bit_count = 0;
    int shrunk_num = dec_num;
    while(shrunk_num) // 非0值表示真，0值表示假
    {
        derived_bits[bit_count++] = shrunk_num % base; // 存入一个余数
        shrunk_num /= base; // 求商
    }

    printf("The dicimal %d is equal to ", dec_num);
    for(int i = bit_count - 1; i >= 0; i--) // 倒序输出
    {
        printf("%d", derived_bits[i]);
    }
    printf(" with base %d.\n", base);
    return 0;
}
```

4. 将串 t 复制到串 s 中，复制时要求将 t 中第 n 个字符之后的字符移到串 s 的开头，其它字符顺序后移。如：ABCD123 换为：123ABCD（n=4）。

```c
#include <stdio.h>
#include <string.h> // 包含字符串操作的库

int main()
{
    char input_str[1024];

    // 每一个单元都初始化为'\0'
```

```
    for(int i = 0; i < 1024 - 1; i++)
    {
        input_str[i] = '\0'; // 也可以写成input_str[i] = 0;
    }
    printf("Input a string:\n");
    gets(input_str); // 输入字符串

    int n = 0;
    printf("Input the number of characters after which movement occurs:\n");
    scanf("%d", &n);

    //
    if(n < 0 || n > strlen(input_str))
    {
        printf("The input location value is invalid.\n");
        return 0;
    }

    char output_str[1024];
    int i = 0;
    // 从字符串input_str中，下标为n的字符开始复制，直到末端
    for(i = n; i < strlen(input_str); i++) //
        strlen(s)表示从s开始，直到'\0'为止，所包含的字符的数目（'\0'不算在内）
    {
        output_str[strlen(output_str)] = input_str[i]; //
            将input_str[i]插入到output_str的末端
    }

    // 从input_str的首字符开始复制，直到下标为n-1的字符
    for(i = 0; i < n; i++)
    {
        output_str[strlen(output_str)] = input_str[i];
    }
    output_str[strlen(output_str)] = '\0'; // 补上'\0'作为字符串末端

    printf("The generated string is: %s.\n", output_str);
    return 0;
}
```

5. 判断一个字符串是否是回文。如：”level” 是回文.

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

# define MAX_LEN 1024

int main()
{
    char str[MAX_LEN];
    printf("Please input a string:\n");
    scanf("%s", str);

    // 逐对逐对检查，只需要检查到字符串的一半长度位置即可
    // 要求顺数第1个和倒数第1个，顺数第2个和倒数第2个...
    for(int i = 0; i < strlen(str) / 2; i++)
    {
        if(str[i] != str[strlen(str) - 1 - i])
        {
            printf("'%s' is not a palindrome.\n");
            return 0;
        }
    }

    printf("'%s' is a palindrome.\n");
    return 0;
}
```

6. 将字符串 str1 插入到字符串 str2 中下标为 pos 的位置处。例如，”abcd” 插入到”12345” 中，下标为 2 的位置，得到”12abcd345”。

```c
#include <stdio.h>
#include <string.h>
int main()
{
    printf("Input a string which will receive another string:\n");
    char base_str[1024];
    gets(base_str);

```

```
    printf("Input a string which will be inserted:\n");
    char inserted_str[1024];
    gets(inserted_str);

    printf("Input an index at which insertion occurs:\n");
    int pos = 0;
    scanf("%d", &pos);

    if(pos < 0 || pos > strlen(base_str))
    {
        printf("The place for insertion is invalid.\n");
        return 0;
    }

    char output_str[1024];
    int i = 0;

    // 先复制前一半
    for(i = 0; i < pos; i++)
    {
        output_str[i] = base_str[i];
    }

    // 然后复制待插入的字符串
    for(; i < pos + strlen(inserted_str); i++)
    {
        output_str[i] = inserted_str[i - pos];
    }

    // 最后复制剩下的一半
    for(; i < strlen(inserted_str) + strlen(base_str); i++)
    {
        output_str[i] = base_str[i - strlen(inserted_str)];
    }

    // 补上末端的'\0'
    output_str[strlen(output_str)] = '\0';

    printf("after insertion, we have '%s'", output_str);
```

```
    return 0;
}
```

7. 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);
    int valid_len = 0;
    for(int i = 0; str[i] != '\0' && str[i] != ' '; i++)
    {
        valid_len++;
    }
    printf("The valid len is: %d.\n", valid_len);
    return 0;
}
```

8. 随机生成 10 个不同的两位正整数。

```
/*****************************解法一*******************************/
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main()
{
    srand(time(NULL)); // 布置随机数种子
    int exist[100]; // exist[i]=1当且仅当i已经被产生过
    for(int i = 0; i < 100; i++)
        exist[i] = 0; // 一开始的时候，每一个数都未被产生过

    int num[10]; // 用于保存被产生的数
    for(int j = 0; j < 10; j++) // 依次做10次
    {
        int rand_num;
```

```
        do{
            rand_num = rand() % 90 + 10; // 随机生成一个两位数
        }while(exist[rand_num] == 1); // 如果之前已经产生过就重来
        exist[rand_num] = 1; // 标记为已经产生过
        num[j] = rand_num; // 保存在num数组中
    }

    for(int j = 0; j < 10; j++)
        printf("%d\t", num[j]);
    printf("\n");

    return 0;
}
```

```
/*****************************解法二*******************************/
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define RAND_NUM_COUNT_REQUIRED 10 // 需要被产生的随机数的个数
#define INTERESTING_START 10 // 从这个位置开始，往后的元素是我们感兴趣的

int main()
{
    srand(time(NULL));
    int num[100]; // 用于保存所有可能被选的数，100足够大
    for(int i = 0; i < 100; i++)
        num[i] = i; // 用于生成等待被选的数，多生成其他一些数不影响

    int cursor = INTERESTING_START; //
        cursor表示当前的关注点（已经选出的数和有待选择的数的边界）

    for(int j = 0; j < RAND_NUM_COUNT_REQUIRED; j++) // 依次做10次
    {
        int rand_location = cursor + rand() % (100 - cursor); //
            选择一个存放两位数的随机位置
        // below swap the numbers at cursor and rand_location respectively
        int temp = num[cursor];
        num[cursor] = num[rand_location];
```

```
24          num[rand_location] = temp;
25          //
26          cursor++; // 游标左边的数已经定下来了，以后只在右边选
27      }
28
29      for(int j = INTERESTING_START; j < INTERESTING_START + RAND_NUM_COUNT_REQUIRED;
            j++)
30      {
31          printf("%d\t", num[j]);
32      }
33      printf("\n");
34
35      return 0;
36  }
```