中山大學
SUN YAT-SEN UNIVERSITY

中山大學
SUN YAT-SEN UNIVERSITY

指针与字符串进阶与练习

中山大学计算机学院

讲课人：课题组

中山大學
SUN YAT-SEN UNIVERSITY

目录

CONTENTS

罗马数字转整数

题目描述：给定一个罗马数字，输出对应的整数。

示例1：
罗马数字：IV
对应整数：4

示例2：
罗马数字：LVIII
对应整数：58
解释：L=50, V=5, III=3

示例3：
罗马数字：MCMXCIV
对应整数：1994
解释：M=1000, CM=900, XC=90, IV=4

题目链接：13. 罗马数字转整数 - 力扣（Leetcode）

罗马数字转整数

解题思路：从左往右扫描字符串，优先匹配CM(900), XC(90)等两个字符的数字，再匹配单字符的数字，将匹配到的数字加起来就是答案。

M	C	M	X	C	I	V

第1次：MC无匹配数字，M匹配1000

第2次：CM匹配900

第3次：XC匹配90

第4次：IV匹配4

因此罗马数字对应整数为1000+900+90+4=1994

罗马数字转整数

解题方法1:

字符串字面量数组查表法。预先定义关键的罗马数字，如CM(900)，CD(400)，XC(90)等，按优先级匹配罗马数字。库函数strncmp(st1, st2, n)可用于判断字符串st1, st2的前n位是否相等

代码:

```
char Roman_table[13][10]={"CM", "CD", "XC", "XL", "IX", "IV", "M", "D", "C", "L", "X", "V", "I"};
int digit_table[13]={900, 400, 90, 40, 9, 4, 1000, 500, 100, 50, 10, 5, 1};
int roman2int_table(char *roman){
    int len=strlen(roman), num=0;
    for (int i=0; i<len; )
        for (int j=0; j<13; ++j)
            if (strncmp(roman+i, Roman_table[j], strlen(Roman_table[j]))==0){
                num+=digit_table[j];
                i+=strlen(Roman_table[j]);
                break;
            }
    return num;
}
```

罗马数字转整数

解题方法2:

控制结构。与方法1一样按优先级匹配罗马数字，只是用if语句进行判断，而非提前做好表格。

代码:

```
int roman2int_control(char *roman){
    int len=strlen(roman), num=0;
    for (int i=0; i<len; )
        if (roman[i]=='C' && roman[i+1]=='M') num+=900, i+=2;
        else if (roman[i]=='C' && roman[i+1]=='D') num+=400, i+=2;
        else if (roman[i]=='X' && roman[i+1]=='C') num+=90, i+=2;
        else if (roman[i]=='X' && roman[i+1]=='L') num+=40, i+=2;
        else if (roman[i]=='I' && roman[i+1]=='X') num+=9, i+=2;
        else if (roman[i]=='I' && roman[i+1]=='V') num+=4, i+=2;
        else if (roman[i]=='M') num+=1000, i++;
        else if (roman[i]=='D') num+=500, i++;
        else if (roman[i]=='C') num+=100, i++;
        else if (roman[i]=='L') num+=50, i++;
        else if (roman[i]=='X') num+=10, i++;
        else if (roman[i]=='V') num+=5, i++;
        else if (roman[i]=='I') num+=1, i++;
    return num;
}
```

验证回文串

题目描述：如果在将所有大写字符转换为小写字符、并移除所有非字母数字字符之后，短语正着读和反着读都一样。则可以认为该短语是一个 **回文串** 。给定一个字符串，判断是否是回文串。

示例1:
字符串：A man, a plan, a canal: Panama
答案：true
解释：amanaplanacanalpanama是回文串

示例2:
字符串：race a car
答案：false
解释：race a car不是回文串

题目链接：125. 验证回文串 - 力扣（Leetcode）

验证回文串

解法:

新建一个空字符串，逐一扫描原字符串的每一个字符，判断是否数字或字母，若是，则转为小写字母后添加到新字符串的末尾，最后得到处理后的新字符串，然后判断是否为回文串。库函数Isalnum(ch)可以判断字符ch是否是数字或字母，tolower(ch)可以返回字母的小写形式，非字母则直接返回ch。

代码:

```
int isPalindrome(char *st)
{
    char removed[maxlen]={'\0'};
    int len_st=strlen(st), len=0;
    for (int i=0; i<len_st; ++i)
        if (isalnum(st[i])) removed[len++]=tolower(st[i]);
    for (int i=0; i<len/2; ++i)
        if (removed[i]!=removed[len-1-i]) return 0;
    return 1;
}
```

杨辉三角

题目描述：给定一个非负索引n，返回杨辉三角的第n行。（从0开始编号）
编写函数int* getRow(int n, int* size), n为非负索引，size指针的值为返回数组的长度，为传出参数

示例1:
n：3
答案：1, 3, 3, 1

示例2:
n：0
答案：1

示例3:
n：1
答案：1, 1

主函数:

```
int main(){
    int n;
    scanf("%d", &n);
    int *size;
    int *ans=getRow(n, size);
    for (int i=0; i<*size; ++i)
        printf("%d ", ans[i]);
    free(size);
    free(ans);
    return 0;
}
```

题目链接：119. 杨辉三角 II - 力扣（Leetcode）

杨辉三角

解法：

可以先新建一个二维数组（局部变量）计算前n行杨辉三角，最后再使用malloc新建一个一维数组，复制杨辉三角第n行并返回。注意函数返回数组要用malloc分配内存，不能使用局部变量，因为局部变量退出函数后会被释放。

代码：

```
int* getRow(int n, int* size){
    *size=n+1;
    int *ret=malloc((n+1)*sizeof(int));
    if (n==0){
        ret[0]=1;
        return ret;
    }
    int a[n+1][n+1];
    memset(a, 0, sizeof(a));
    for (int i=0; i<=n; ++i){
        a[i][0]=1;
        for (int j=1; j<=i; ++j)
          a[i][j]=a[i-1][j-1]+a[i-1][j];
    }
    for (int i=0; i<=n; ++i) ret[i]=a[n][i];
    return ret;
}
```

二进制求和

题目描述：给定两个二进制字符串a, b, 以二进制字符串的形式返回它们的和。
编写函数char *addBinary(char *a, char *b)

示例1:
a：11
b：1
答案：100

示例2:
a：1010
b：1011
答案：10101

题目链接：67. 二进制求和 - 力扣（Leetcode）

主函数:

```
int main(){
    char *a=malloc(maxlen), *b=malloc(maxlen);
    scanf("%s", a);
    scanf("%s", b);
    char *sum=addBinary(a, b);
    puts(sum);
    free(sum);
    free(a);
    free(b);
    return 0;
}
```

二进制求和

解法：

预先新建存放答案的字符数组，注意与整数数组不同的是，字符数组的末尾还需要添加'\0'表示结束，因此分配内存的大小要在最大长度的情况下加1.

存放答案时可以先将最低位放在数组的第0位，方便处理最高位的进位，最后再反转。

代码：

```
char *addBinary(char *a, char *b){
    int len_a=strlen(a), len_b=strlen(b), carry=0;
    int len_sum=len_a>len_b? len_a:len_b;
    char *sum=malloc(len_sum+2); //预留进位和结束符'\0'
    for (int i=len_sum; i<len_sum+2; ++i) sum[i]='\0';
    for (int i=0; i<len_sum; ++i){
        int s=carry+(i<len_a? a[len_a-1-i]-'0':0)+(i<len_b?
b[len_b-1-i]-'0':0);
        carry=s>>1;
        s&=1;
        sum[i]='0'+s;
    }
    if (carry) sum[len_sum++]='1';
    for (int i=0; i<len_sum/2; ++i){
        char ch=sum[i];
        sum[i]=sum[len_sum-1-i];
        sum[len_sum-1-i]=ch;
    }
    return sum;
}
```

中山大學
SUN YAT-SEN UNIVERSITY

中山大學
SUN YAT-SEN UNIVERSITY

谢 谢

中山大学计算机学院

编制人：课题组