@[toc]

c程序设计总结（红皮书+真题）

1、将一个整数分解成质因数，相乘例如90打印出90=2 * 3 * 3 * 5

```
//将一个正整数分解成质因数
/*思路:
    1、从2开始的质因数不断除以该正整数n,能整除就输出，直到不能整除
    2、换下一个质因数去除，重复上述操作，直到输出最后一个质因数
*/
int main(){
    int i,n;
    printf("请输入n:");
    scanf("%d",&n);
    printf("%d=",n);
    for(int i = 2; i < n; i++){
        while(i){//一直循环到该质因数不能被整除后跳到else中
            if(n%i == 0){
            n=n/i;
            printf("%d*",i);
            }
            else{//该个质因数已经除完了，到下一个质因数
                break;
            }
        }
    }
    printf("%d",n);//只剩下最后一个质因数
    return 0;
}
```

2、编写函数判断两个整数是否互质，使用辗转相除法求两个整数M,N的最大约数

```
//辗转相除法,m%n（m>n），m = n, n = m%n 直到n为0
int prime(int m, int n){
    int temp;
    while(n!=0){
        temp = m%n;
        m = n;
        n = temp;
    }
    if(m ==1) return 1;
    else return 0;
}
```

3、给出年月日计算该日是该年的第几天

```
//判断是否是闰年：能被4整除但是不能被100整除，或者是能被400整除
int leap_year(int year){
    if((year%4 == 0  && year%100 != 0) || year%400 == 0){
        return 1;
```

```
    }
    else{
        return 0;
    }
}

int main(){
    //给出年月日，计算该日是该年的第几天
    int months[13] = {0,31,28,31,30,31,30,31,31,30,31,30,31};     //12月各月份天数
    int year,month,day,count = 0;
    scanf("%d%d%d",&year,&month,&day);
    if(leap_year(year)){//闰年的二月多一天，不是闰年就28天
        months[2] += 1;
    }
    for(int i = 1; i < month; i++){//统计每个月的天数
        count+=months[i];
    }
    count+=day;                         //加上最后一天的天数
    printf("有%d天",count);
    return 0;
}
```

4、(二进制枚举)对输入的任意正整数，打印出几何{0,1，....,n-1}的所有子集 例如输入3 输出{}、{0}， {1}，{0,1},{2}，{0,2}，{1,2}，{0,1,2}

```
#include<stdio.h>


//对输入的任意正整数，打印出几何{0,1，....,n-1}的所有子集
//例如输入3 输出{}、{0}， {1}，{0,1},{2}，{0,2}，{1,2}， {0,1,2}
int main()
{
    //思路：二进制枚举，从0到2^n-1个数中，转换为二进制数后，什么位置是1组成了上述子集
    //{}、{0}， {1}，{0,1},{2}，{0,2}，{1,2}， {0,1,2}  代表 0 1 10 11 110 101 110
111
    int n;
    int flag;
    scanf("%d",&n);
    for(int i = 0; i < 1 << n; i++){        //总共2^n个数，从0~2^n-1
        flag = 1;
        printf("{");
        for(int j = 0; j < n; j++){
            if(i & (1 << j)){               //判断第j位是不是0  1<<j 表示第j位为1 & i
判断第j位是否为1
                if(flag){                   //判断是否为第一位
                    printf("%d",j);
                    flag = 0;
                }
                else{
                    printf(",%d",j);
                }
            }
        }
```

```
        printf("}\n");
    }
    return 0;
}
```

5、将a进制的数n 转化为b进制数，并输出

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

//对输入a 进制的数 n 转换为 b进制的数
int main()
{
    //算法思想，先把a进制的数转换为10进制的数，再把十进制的数转换为b进制的数
     int a,b;
     char str[40];
     while(scanf("%d%s%d",&a,str,&b)!=EOF){
        int tmp = 0, lenth = strlen(str);
        int weights = 1;
        for(int i = lenth - 1; i >=0; i--){//从高位到低位计算十进制的数
            int x;
            if(str[i] >= '0' && str[i] <= '9'){//如果是0-9的数字
                x = str[i] - '0';
             }
             else if(str[i] >= 'a' && str[i] <= 'z'){//如果是小写字母
                x = str[i] - 'a' + 10;
             }
             else{//剩下的就是大写字母
                x = str[i] - 'A' + 10;
             }
             tmp += x*weights;
             weights *= a;           //计算下一轮权重的值
         }
         char ans[40],size = 0;      //ans保存转换进制后的各位数
         while(tmp != 0){            //将求得的十进制数转换为b进制数
            int x = tmp%b;
            ans[size++] = x < 10 ? x+'0' : x-10+'A';
            tmp = tmp/b;
         }
         //输出
         printf("%s 的%d进制数转换为%d数为：",str,a,b);
         for(int i = size - 1; i >= 0; i--){
            printf("%c",ans[i]);
         }
         printf("\n");
     }

    return 0;
}
```

6、设计一个数组，将数组a[n]分为两部分，左边为奇数，右边的为偶数

```
//设计一个数组，将数组a[n]左边为奇数，右边的为偶数
void divide(int ans[], int n){
    //采用双指针解法，一个指针指向数组开头，一个指向末尾，向中间找，找到了就交换，类似于快速排序
    int i = 0,j = n-1;
    while(i < j){
        while(i < j && ans[i]%2 != 0){
            i++;
        }
        while(i < j && ans[j]%2 == 0){
            j--;
        }
        if(i < j){
            int temp = ans[i];
            ans[i] =ans[j];
            ans[j] = temp;
        }
    }
}
```

7、一个m行n列的整型矩阵A，编写一个swap函数，使得对A元素进行交换，第一个元素和倒数第一个元素交换，第二个元素和倒数第二个元素交换 ,不另外设置矩阵

```
#include<stdio.h>
#define m 3
#define n 4
void swap(int a[m][n]){
    //思路：对于前半部分的值与后半部分的值进行交换
    //添加一个计数器，计数到一半时，终止
    int i,j,temp,count = 0;
    for(i = 0; i < m; i++){
        for(j = 0; j < n; j++){
            if(count == m*n/2) return;
            temp = a[i][j];
            a[i][j] = a[m-i-1][n-j-1];
            a[m-i-1][n-j-1] = temp;
            count++;
        }
    }
}
```

8、两个升序数组a[m] b[n] 合并成一个升序数组c

```
#include<stdio.h>
#define m 6
#define n 5
//两个升序数组a[m] b[n] 合并成一个升序数组c
void fun(int a[], int b[], int c[]){
```

```
    //思路：双指针，看哪个先遍历完，遍历完之后，只剩下一个数组，再将剩下的数组放到c中
    int i=0,j=0,d=0, k=0;
    while(i < m && j < n){
        if(a[i]>b[j]){
            c[k++] = b[j++];
        }
        else{
            c[k++] = a[i++];
        }
    }
    if(i == m){
        for(d = j; j < n; j++){//之前写的是j-1,这里为j是因为j-1已经遍历过了，j是下一个元素 ，只需要对下一个元素进行操作即可
            c[k++] = b[j];
        }
    }
    if(j == n){
        for(d = i; i < m; i++){
            c[k++] = a[i];
        }
    }
}
```

9、1个整数数组，所有的偶数从小到大放在数组的前半部分，所有的奇数按从小到大存放在数组的后半部分

```
#include<stdio.h>
#include<string.h>
#define N 7

void Sort(int *a, int m, int n){//冒泡排序进行排序
    int temp;
    for(int i = m; i < n-1; i++){
        for(int j = m; j< n-1-i+m;j++){//这里为 j < n-1  如果为j < n-i-1  那么j无法到达a数组末端，因为i不是从0开始的,所以要加上一个 m
            if(a[j] > a[j+1]){
                temp = a[j+1];
                a[j+1]= a[j];
                a[j] = temp;
            }
        }
    }
}

int main()
{
    int i = 0, j = N-1, p = 0, q = 0;
    int a[N] = {7,9,3,2,5,4,1};
    while(i < j){
        while(a[i]%2==0){
            i++;
            p++;
        }
        while(a[j]%2!=0){
```

```
            j--;
            q++;
        }
        if(i < j){
            int temp = a[i];
            a[i] = a[j];
            a[j] = temp;

        }
    }
//  Sort(a,0,p-1);  Sort(1,p,p+q-1) p-1的话，会剩下一个数没被遍历到 ，不会被排序
    Sort(a,0,p);
    Sort(a,p,p+q);
    for(int i = 0; i < N; i++){
        printf("%d  ",a[i]);
    }
    return 0;
}
```

10、对数组进行右移操作

```
#include<stdio.h>
#define N 20

void move(int a[], int n, int offset){//直接模拟
    while(offset--){//右移次数
        int temp = a[n-1];
        for(int i = n-1; i > 0; i--){
            a[i] = a[i-1];
        }
        a[0]=temp;
    }
}
int main()
{
    int arr[N] = {1,2,3,4,5,6,7,8,9,10};
    move(arr,10,2);
    for(int i = 0; i < 10; i++){
        printf("%d  ",arr[i]);
    }
    return 0;
}
//其实也是之前做的一个交换向量的题 a[a1,a2,an,b1,b2,bm],交换成a[b1,b2,bm,a1,a2,an],就是右移m的操作
//思路：对数组最后offset个数数组逆转，前半部分逆转，然后整个数组逆转，也可得到
//eg a[1,2,3,4,5,6,7] 右移三位 a[1,2,3,4,7,6,5] a[4,3,2,1,7,6,5] a[5,6,7,1,2,3,4]
```

11、（不会）两个聊天机器人对话

1、M1只会说'Y','N','2'

2、M2只会说'y','n','1'

3、M1主动说话

4、当一个机器人说话讲的不是数字时，它必须继续说话，对方不能不说

5、当一个人说出数字时，它自己停止说话，对方可以接着说，也可以不说从而结束对话

```
#include<stdio.h>

int main(){
    //思路：首先判断是否为M1说话，如果不是则退出
    //然后继续判断是否是遇到数字后变成另外一个人说话， 期间只能是一个人说YN,如果不是，则进行判断是否结束，未结束，第二个人说话，同样判断条件，然后while循环，跳出循环后，如果str[i-1]不是数字字符，语法错，如果i不是结束字符，那么也错。
    int i = 0;
    char str[100];
    scanf("%s",str);
    while(str[i] != '\0'){
        //M1先说，如果没说，不符合规则就break
        while(str[i] == 'Y' || str[i] == 'N') i++;      //M1一直说话
        if(str[i] == '2') i++;
        else break;

        //M1说完，M2说
        while(str[i] == 'y' || str[i] == 'n') i++;
        if(str[i] == '1') i++;
        else break;
    }
    if(str[i-1] == '1' || str[i-1] == '2'){
        if(str[i] == '\0'){//已经结束对话
            printf("是机器人对话\n");
        }
        else{
            printf("不是机器人对话\n");
        }
    }
    else{
        printf("不是机器人对话\n");
    }
    return 0;
}
```

12、从给定的向量中删除元素值为x到y之间的所有元素（向量要求元素之间不能有间断）

```
int del(int A[], int n, int x, int y){
    //思路：通过k统计删除的个数，然后这就是其下一个元素需要前移的位数
    int i,k=0;
    for(i = 0; i < n; i++){
        if(A[i] >= x && A[i] <= y){
            k++;
        }
        else{
            A[i-k] = A[i]; //后面元素前移的位数
        }
    }
    return n-k;
```

```
}
```

13、把整数数组中值相同的元素删除只剩下最后一个，并把剩余元素全部前移到前面

```
//和前面思路类似，利用k来统计前移次数，但是得逐个比较，逐个前移
void fun(int a[], int n){
    int i,j,k = 0;
    for(i = 0; i < n; i++){
        for(j = i+1; j < n; j++){
            if(a[i] == a[j]) k++;
            else             a[j-k] = a[j];
        }
        k=0; //统计个数重新清零
    }
}
```

14,a,b严格递增数组合并，b数组 合并在a数组中，不能申请额外的数组,保证合并后仍为严格递增，函数返回合并后a数组元素

```
 int sort(int a[], int m,int b[], int n){
    int i,j=0,k=0,len=m;                    //需要一个新的len存储当前数组长度
    for(i = 0; i < n; i++){
        while(j < len && a[j] < b[i]) j++;      //找到插入位置，j<len和当前数组长度进行比较

        if(a[j] == b[i]) continue;              //因为需要合并后仍然为严格递增，所以相等的情况忽略

        for(k = len-1; k >= j; k--){//后移
            a[k+1] = a[k];
        }
        a[j] = b[i];
        len++;          //记录后移的最后一位
    }
    return len;             //len表示当前数组长度
}
```

15、使用数组精确计算M/N(0<M<N<=100)的各位小数的值。如果M/N是无限不循环小数，则计算并输出它的第一循环小节，同时要求输出循环节的起止位置

```
#include<stdio.h>


int main(){
    int num, den,i=0,j=0,k = 0;
    int a[100];         //存放小数部分
    int b[100];         //存放余数部分
    scanf("%d%d", &num, &den);
```

```
    int s = 0;
    while(num > den){
        s = num/den;          //整数部分
        num = num%den;        //更小的分子
    }

    while(num != 0){
        num = num*10;
        a[i] = num/den;       //更新小数部分
        num = num%den;
        b[i] = num;       //更新余数部分
        for(j = 0; j < i; j++){
            if(b[j] == num){//出现余数相同，即开始出现循环小数
                printf("从小数点%d位开始循环，到%d位结束",j+1, i);
                num = 0;
                break;
            }
        }
        i++;
    }
    printf("%d.",s);
    for(j = 0; j < i; j++){
        printf("%d",a[j]);
    }
    return 0;
}
```

16、编写程序生成和输出10*10螺旋矩阵，例如5 * 5的矩阵如下

```
#include<stdio.h>
#define N 10

//生成螺旋矩阵
int main(){
    //思路：模拟
    int a[10][10];
    int i=0, j=0, k=0, num=1;

    for(k=0; k<= N/2; k++){          //赋值N/2次，每一次都是一个矩阵环，奇数偶数一致

        for(i = k; i <= N-k-1; i++){//最上一行赋值    这里赋值十个，后面 9个依次递减
            a[k][i] = num++;
        }
        for(j = k+1; j < N-k-1; j++){//最右一行赋值 这里要从k+1开始赋值，然后赋值 8个
            a[j][N-1-k] = num++;
        }
        for(i = N-k-1; i > k; i--){//最下一行赋值 这里注意要位置递减赋值 ，赋值9个
            a[N-1-k][i] = num++;
        }
        for(j = N-k-1; j > k; j--){//最左一行赋值 这里注意要位置递减赋值 ，赋值9个
            a[j][k] = num++;
        }
    }
    for(i = 0; i < N; i++){
        for(j = 0; j < N; j++){
```

```
            printf("%-5d", a[i][j]);
        }
        printf("\n");
    }
    return 0;

}
```

17、输出s的值，精度为1e-6

```
int main(){
    float term = 1,n = 0;
    int i;
    for(i = 1; term > 1e-6; i++){
        //这里需要注意c语言中的float与int类型的转换，必须写1.0才表示为一个float型，不能写出1
        term = 1.0/(2*i-1) * 2.0*i /(2.0*i-1);
        n += term;
    }
    printf("%f ", n);
    return 0;
}
```

18、输出魔方阵

```
//输出魔方阵
/*
1、第一行的中间一列为1，用j = n/2+1 确定a[1][j] = 1
2、每一个数存放的行比前一个数的行数减1， 列数加1
3、一个数行数为第一行，下一个数行数为最后一行
4、一个列数为最后一列，下一个数列数为第一列
5、如果按上面的规则所确定的位置上已有数，或上一个数是第一行最后一列，则把下一个数放在上一个数的下面
*/
void OutMagicCube(){
    int a[N][N]={0},i,j,k,n;
    n = 4;

    i = n+1;            //从最后一行开始存储
    j = n/2+1;
    a[1][j] = 1; //第一行中间一列的值为1

    for(k = 2; k <= n*n; k++){
        i = i-1;
        j = j+1;                //每一个数为 行数减1，列数加1
        if((i<1) && (j>n)){
            i = i+2;            //如果为最后一列并且为第一行
            j = j-1;
        }
        else{
            if(i<1) i = n;      //上个数第一行，下个数第n行
            if(j>n) j = 1;      //上个数第n列，下个数第一列
        }
        if(a[i][j] == 0) a[i][j] = k;
```

```
        else{                    //有数，放在下面 i+2本来是-1，然后行数要+2才是下一行，列不变，之前+1，现在-1
            i = i+2;
            j = j-1;
            a[i][j] = k;
        }
    }

    for(i = 1; i <= n; i++){
        for(j = 1; j<=n; j++){
            printf("%-4d", a[i][j]);
        }
        printf("\n");
    }
}
```

19、m行n列矩阵，编写程序将矩阵中值小于0的元所在行与列上的所有元素置为0，并输出

```
//思路：不能直接赋值为0，得先把0放到第一行或者第一列，然后还需要两个变量来保存第一行或者第一列来保存最开始的结果

void outQuadrix(int a[][N], int m, int n){
    int row = 0, col = 0;                //判断第一行或者第一列是否为0的标志
    for(int i = 0; i < m; i++){
        for(int j = 0; j < n; j++){
            if(a[i][j]<=0){
                if(i==0){
                    row = 1;
                }
                if(j==0){
                    col = 1;
                }
                a[0][j]=0;           //第一行的第j列置为0
                a[i][0]=0;           //第一列的第i行置为0
            }
        }
    }

    for(int i = 1; i < m; i++){
        for(int j = 1; j < n; j++){
            if(a[i][0]==0 || a[0][j]==0){        //如果该行或者该列存在0，则该行或者该列的元素全部置为0
                a[i][j] = 0;
            }
        }
    }

    if(col){
        for(int i = 0; i < m; i++)
            a[i][0] = 0;
    }
    if(row){
        for(int i=0; i < n; i++){
```

```
            a[0][i] = 0;
        }
    }
    for(int i = 0; i < m; i++){
        for(int j = 0; j < n; j++){
            printf("%-4d",a[i][j]);
        }
        printf("\n");
    }
}
```

20、设计字符串S以及长度为n的字符型一维数组a，编写一个函数，统计a中每个字符在字符串S中出现的次数，要求该函数以s,a,n为形参，一维整型数组为返回值

```
//思路：先统计S中每个字符出现的次数，然后再遍历a，通过hash表反应a中每个字符在s中出现的次数
int *count(char *s, char a[], int n){
    int *hash = (int *)malloc(sizeof(int)*255);      //C语言中char类型默认是有符号类型,字符串的ASCII码范围是0-255
    int *result = (int *)malloc(sizeof(int)*n);      //保存结果
    memset(hash,0,sizeof(int)*255);                  //初始化赋值
    int i = 0;
    while(s[i]!='\0'){
        hash[s[i++]]++;
    }
    for(i=0; i < n; i++){
        printf("%c在S中出现的次数为%d\n",a[i],hash[a[i]]);
        result[i] = hash[a[i]];
    }
    return result;
}
```