《程序设计基础课程设计》实验报告

班级：1603019

姓名：张俊华

学号：16030199025

所选题目：1_1, 1_2, 1_3, 2_2, 3_2, 4_1, 5_1, 5_2

第 1_1 题

比较两个文本文件并打印出它们第一个不相同的行（文件每行字符数不多于 80）。

算法描述：

在未读到文件尾前，不断调用 fgets 函数，读取文件行到字符串，通过 strcmp 比较两个字符串是否相同，若字符串不同，则输出两个字符串，程序结束。

本程序进一步拓展了功能，在输出不同字符串的基础上，对字符串每一位分别判断，若发现这一位不同，调用 SetConsoleTextAttribute 函数，修改控制台字符颜色，实现了高亮输出不同字符的功能。

源程序：

no1_1.c

```
void findDifferent(FILE *fp1, FILE *fp2) {
    /**
     * @brief 比较两个文件指针对应的 txt 不同之处
     * @param fp1 第一个 txt 对应的文件指针
     * @param fp2 第二个 txt 对应的文件指针
     * @author 张俊华 16030199025
     */
    const int MAX_LENGTH = 80; //每行的最大长度
    char tempString1[MAX_LENGTH];   //保存文件一的字符串
    char tempString2[MAX_LENGTH];   //保存文件一的字符串
    HANDLE handle;
    handle = GetStdHandle(STD_OUTPUT_HANDLE);
    int j = 0; //保存行号
    int i = 0;
    while (1) {
        j++;//增加行号
        if ((fgets(tempString1, MAX_LENGTH, fp1) == NULL) &
            (fgets(tempString2, MAX_LENGTH, fp2) == NULL)) {
            puts("未检测到不同");
        };

        if (isspace(*(tempString1 + strlen(tempString1) - 1)))
            tempString1[strlen(tempString1) - 1] = '\0';
        if (isspace(*(tempString2 + strlen(tempString2) - 1)))
            tempString2[strlen(tempString2) - 1] = '\0';
        //去掉末尾的空白字符

        if (strcmp(tempString1, tempString2)) {
            printf("在第%d 行捕捉到不同\n",j);
            for (i = 0;
                 i <= (strlen(tempString1) < strlen(tempString2) ? strlen(tempString1) :
strlen(tempString2)); i++) {
                if (*(tempString1 + i) == *(tempString2 + i)) {
                    SetConsoleTextAttribute(handle, FOREGROUND_INTENSITY);
                    printf("%c", *(tempString1 + i));
                } else {
                    SetConsoleTextAttribute(handle, FOREGROUND_INTENSITY | FOREGROUND_RED);
                    printf("%c", *(tempString1 + i));
                    SetConsoleTextAttribute(handle, FOREGROUND_INTENSITY);
                }
            }
            if (i < strlen(tempString1)) {
                SetConsoleTextAttribute(handle, FOREGROUND_INTENSITY | FOREGROUND_RED);
                puts(tempString1 + i);
                SetConsoleTextAttribute(handle, FOREGROUND_INTENSITY);
            } else printf("\n");

            for (i = 0;
```

```
                i <= (strlen(tempString2) < strlen(tempString1) ? strlen(tempString2) :
strlen(tempString1)); i++) {
                if (*(tempString1 + i) == *(tempString2 + i)) {
                    SetConsoleTextAttribute(handle, FOREGROUND_INTENSITY);
                    printf("%c", *(tempString2 + i));
                } else {
                    SetConsoleTextAttribute(handle, FOREGROUND_INTENSITY | FOREGROUND_RED);
                    printf("%c", *(tempString2 + i));
                    SetConsoleTextAttribute(handle, FOREGROUND_INTENSITY);
                }
            }
            if (i < strlen(tempString2)) {
                SetConsoleTextAttribute(handle, FOREGROUND_INTENSITY | FOREGROUND_RED);
                puts(tempString2 + i);
                SetConsoleTextAttribute(handle, FOREGROUND_INTENSITY);
            } else printf("\n");

            return;
        }
    }


}

int main() {
    /**
     * @brief 比较两个 txt 的不同之处，打印第一个不同的行
     * @author 张俊华 16030199025
     */

    HANDLE handle;
    handle = GetStdHandle(STD_OUTPUT_HANDLE); //输出句柄
    puts("txt 文件比较工具");
    puts("请将需要比较的 txt 文件分别命名为\"txt1.txt\",\"txt2.txt\",和本程序放在相同目录下。");
    puts("准备好了吗？");
    getchar();

    FILE *fp1;
    FILE *fp2;
    fp1 = fopen("txt1.txt", "r");
    fp2 = fopen("txt2.txt", "r");
    if (fp1 == NULL || fp2 == NULL) {
        puts("读取文件出错，请检查！");
        getchar();
        exit(1);
    }
    //初始化文件指针_只读模式

    findDifferent(fp1, fp2); //调用 findDifferent 比较两个文件指针对应的 txt 不同之处
    fclose(fp1);
    fclose(fp2);
}
```

测试数据（输入、输出）：


E:\workspace\cfile\Compare2TXT\main2.exe
txt文件比较工具
请将需要比较的txt文件分别命名为"txt1.txt","txt2.txt",和本程序放在相同目录下。
准备好了吗?
在第3行捕捉到不同
- Super fast SAD powered premium servers!!!
- Super fast SSD powered premium servers.
Process exited after 3.399 seconds with return value 0
请按任意键继续. . .


E:\workspace\cfile\Compare2TXT\main2.exe
txt文件比较工具
请将需要比较的txt文件分别命名为"txt1.txt","txt2.txt",和本程序放在相同目录下。
准备好了吗?
在第2行捕捉到不同
Enwrought with golden and silver light,
Enwrought have golden and silver light,
Process exited after 1.174 seconds with return value 0
请按任意键继续. . .

第 1_2 题

文本文件 num1.txt 和 num2.txt 中各有一组用空格分隔的整数，将 num1.txt 和 num2.txt 联合排序，并将结果保存在 num3.txt 中。

算法描述：

首先新建文件指针，以只读模式打开两个 txt 文件，然后用 while 循环读入整数，读到文件尾结束。将两个 txt 中的数字保存在数组中。调用 qsort 进行排序，排序完成后循环输出，最后将排序好的数据输出到新的文件中。

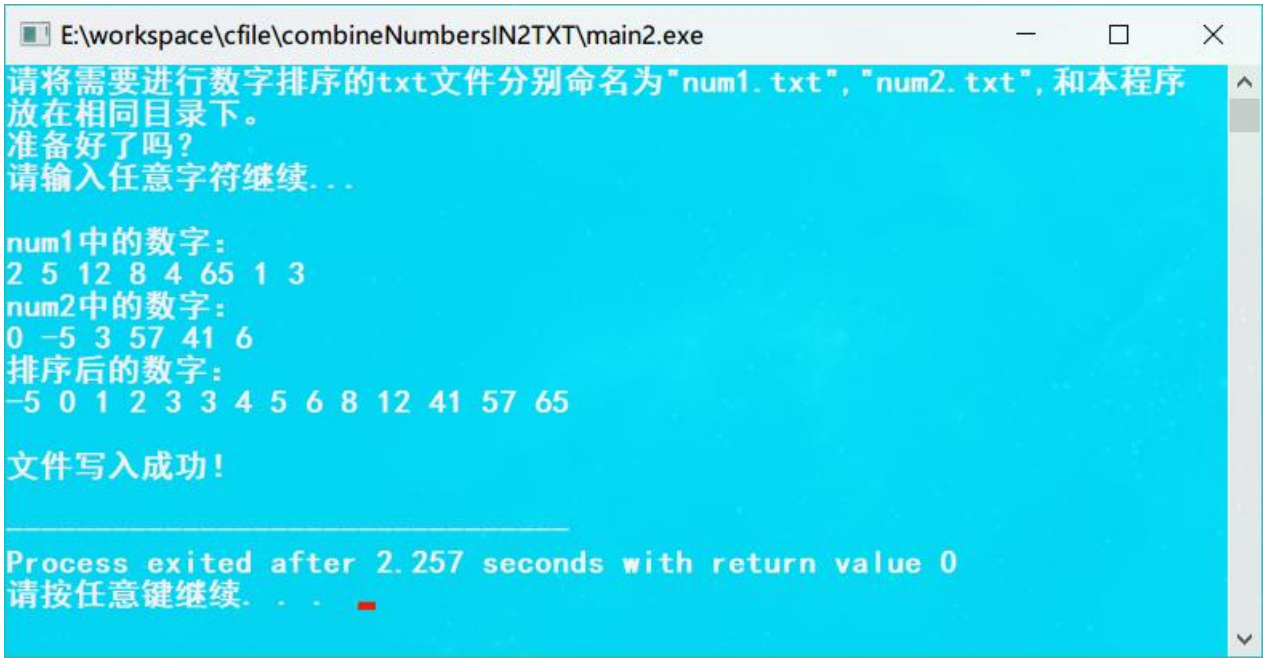
源程序：

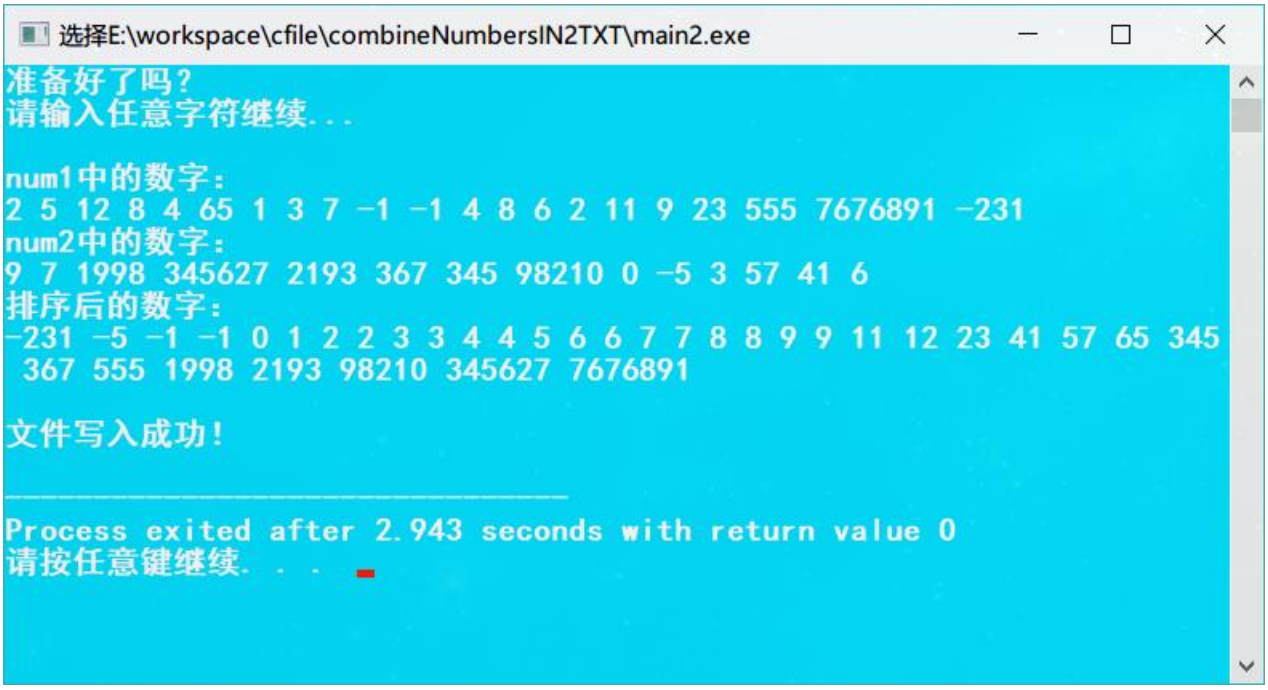
no1_2.c

```c
int compInc(const void *a, const void *b)
{
    return *(int *)a - *(int *)b;
}
int main() {
    /**
     * @brief  对两个 txt 中的数字进行排序并输出新的 txt
     * @author 张俊华 16030199025
     */

    const int MAX_COUNT = 1000;//最大排序数字数目
    puts("请将需要进行数字排序的 txt 文件分别命名为\"num1.txt\",\"num2.txt\",和本程序放在相同目录下。");
    puts("准备好了吗？");
    puts("请输入任意字符继续...");
    getchar();

    FILE * fp1;
    FILE * fp2;
    fp1=fopen("num1.txt","r");
    fp2 = fopen("num2.txt","r");
    if(fp1==NULL||fp2==NULL){
        puts("读取文件出错，请检查！");
        getchar();
        return 0;
    }
    //初始化文件指针_只读模式

    int numbers[MAX_COUNT];
    int i = 0; //numbers 下标
    int j = 0; //文件读入下标
    while (fscanf(fp1,"%d",&numbers[i++])!=EOF);
    i--;
    //输出原始数据
    puts("num1 中的数字：");
    for (j = 0; j < i; j++) {
        printf("%d ",numbers[j]);
    }
    printf("\n");

    while (fscanf(fp2,"%d",&numbers[i++])!=EOF);
    i--;
    puts("num2 中的数字：");
    for (; j < i; j++) {
        printf("%d ",numbers[j]);
    }
    printf("\n");
```

```
    //关闭文件指针
    fclose(fp1);
    fclose(fp2);

    qsort(numbers,i, sizeof(numbers[0]),compInc); //进行排序

    puts("排序后的数字：");
    for (int l = 0; l < i; l++) {
        printf("%d ",numbers[l]);
    }//展示排序结果
    //写入文件
    FILE * fp3;
    fp3=fopen("num3.txt","w+");
    for (int m = 0; m < i; ++m) {
        fprintf(fp3,"%d ",numbers[m]);
    }
    puts("\n\n 文件写入成功！");
    fclose(fp3);
    return 0;
}
```

测试数据（输入、输出）：




选择E:\workspace\cfile\combineNumbersIN2TXT\main2.exe
准备好了吗?
请输入任意字符继续. . .
num1中的数字:
2 5 12 8 4 65 1 3 7 -1 -1 4 8 6 2 11 9 23 555 7676891 -231
num2中的数字:
9 7 1998 345627 2193 367 345 98210 0 -5 3 57 41 6
排序后的数字:
-231 -5 -1 -1 0 1 2 2 3 3 4 4 5 6 6 7 7 8 8 9 9 11 12 23 41 57 65 345
367 555 1998 2193 98210 345627 7676891
文件写入成功!
Process exited after 2.943 seconds with return value 0
请按任意键继续. . .

第 1_3 题

现有两个文本文件 db1.txt 和 db2.txt。db1.txt 中第一列为姓名，第二列为英语成绩；db2.txt 中第一列为姓名，第二列为数学成绩。通过姓名字段将 db1.txt 文件关联到 db2.txt 文件生成 db3.txt 文件。db3.txt 文件第一列为姓名，第二列为英语成绩，第三列为数学成绩，第四列为平均成绩。

算法描述：

学生的姓名和分数信息选用结构体数组保存,

1. 首先从 db1.txt 中获取姓名和分数数据，保存到结构体数组中，
2. 再从 db2.txt 中获取数学成绩，遍历结构体数组，将数学成绩值赋给 name 相同的结构体。
3. 再遍历结构体，计算平均分数
4. 对结构体按照平均数进行排序
5. 遍历结构体输出

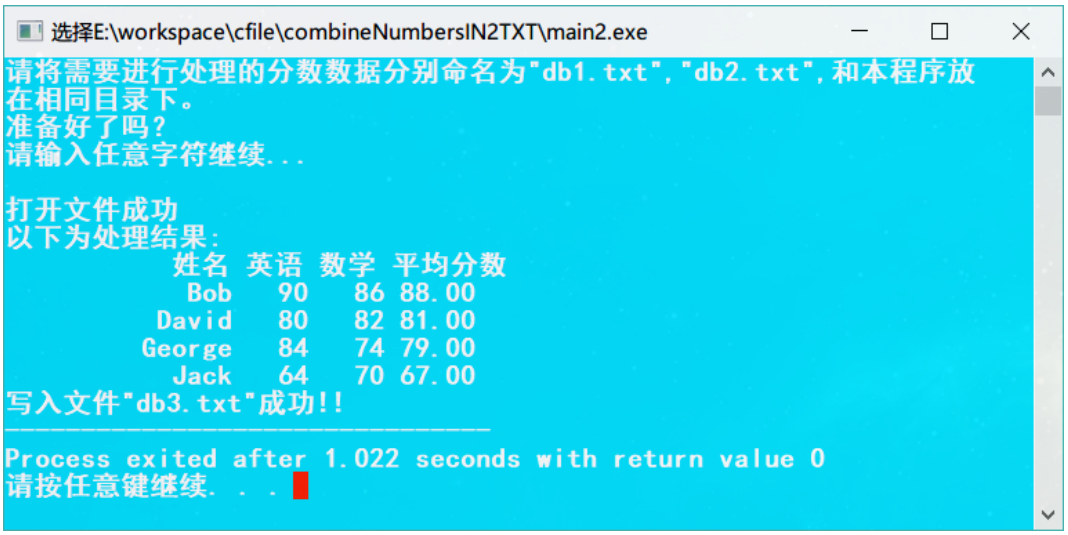
源程序：

no1_3.c

```
typedef struct {
    char name[20]; //姓名
    int englishGrade; //英语成绩
    int mathGrade; //数学成绩
    double average; //平均成绩
} StuGrades;

StuGrades StructstuGrades[MAX_STUDENTS_AMOUNT];
//结构体数组,存放学生信息

int comInt(const void *p, const void *q) {
    return -(int) ((*(StuGrades *) p).average - (*(StuGrades *) q).average);
}

int main() {
    /**
     * @brief 成绩处理工具
     * @author 张俊华 16030199025
     */
    //初始化文件指针
    puts("请将需要进行处理的分数数据分别命名为\"db1.txt\",\"db2.txt\",和本程序放在相同目录下。");
    puts("准备好了吗？");
    puts("请输入任意字符继续...");
    getchar();

    FILE *fp1 = fopen("db1.txt", "r");
    FILE *fp2 = fopen("db2.txt", "r");
    FILE *fp3 = fopen("db3.txt", "w+");
    if (fp1 == NULL || fp2 == NULL || fp3 == NULL) {
        puts("打开文件失败！");
    }
    puts("打开文件成功");

    int i = 0;
    char name[20];
    int grade;
    while (fscanf(fp1, "%s%d", StructstuGrades[i].name, &StructstuGrades[i].englishGrade) !=
EOF)
        i++;

    while (fscanf(fp2, "%s%d", name, &grade) != EOF) {
```

```
        //寻找学生在上一个名单的位置
        for (int j = 0; j < i; ++j) {
            if (strcmp(name, StructstuGrades[j].name) == 0) {
                StructstuGrades[j].mathGrade = grade;
            }
        }
    }

    for (int j = 0; j < i; ++j) {
        StructstuGrades[j].average = (StructstuGrades[j].englishGrade +
StructstuGrades[j].mathGrade) / 2.0;
    }//计算平均分数
    //排序
    qsort(StructstuGrades, i, sizeof(StuGrades), comInt);

    //输出
    puts("以下为处理结果:");
    printf("%15s %4s %4s %4s\n", "姓名", "英语", "数学", "平均分数");
    for (int k = 0; k < i; ++k) {
        printf("%15s %4d %4d %4.2lf\n", StructstuGrades[k].name,
StructstuGrades[k].englishGrade,
               StructstuGrades[k].mathGrade, StructstuGrades[k].average);
    }
    //写入文件
    for (int k = 0; k < i; ++k) {
        fprintf(fp3, "%15s %4d %4d %4.2lf\n", StructstuGrades[k].name,
StructstuGrades[k].englishGrade,
                StructstuGrades[k].mathGrade, StructstuGrades[k].average);
    }
    printf("写入文件\"db3.txt\"成功!!");
}
```

测试数据（输入、输出）：



第 2_2 题

统计一个英文文本文件中 26 个英文字母出现次数并按英文字母序输出统计结果，查找并替换此英文文本文件中某字符串。

算法描述：

统计字母出现的次数：

先定义数组 countlist，用该数组保存字母次数。下标 0-26 分别对应字母 a-z，字母每出现一次，对应下标的数组值加 1。最后遍历数组并输出，即可得到每个字母出现的次数。

查找替换字符串：

1. 打开一个临时文件，
2. 遍历文件，若这一行不包含需要替换的字符串，则将这一行直接复制到临时文件中
3. 若这一行包含需要替换的字符串，则对这行字符串执行替换操作
4. 替换时，先将前半段相同的字符复制到新的字符串中，再把替换内容复制到新字符串中，最后加上原字符串的后半部分和终止标志\0
5. 清空原文件，将替换好的临时文件内容写到原文件中

源程序：

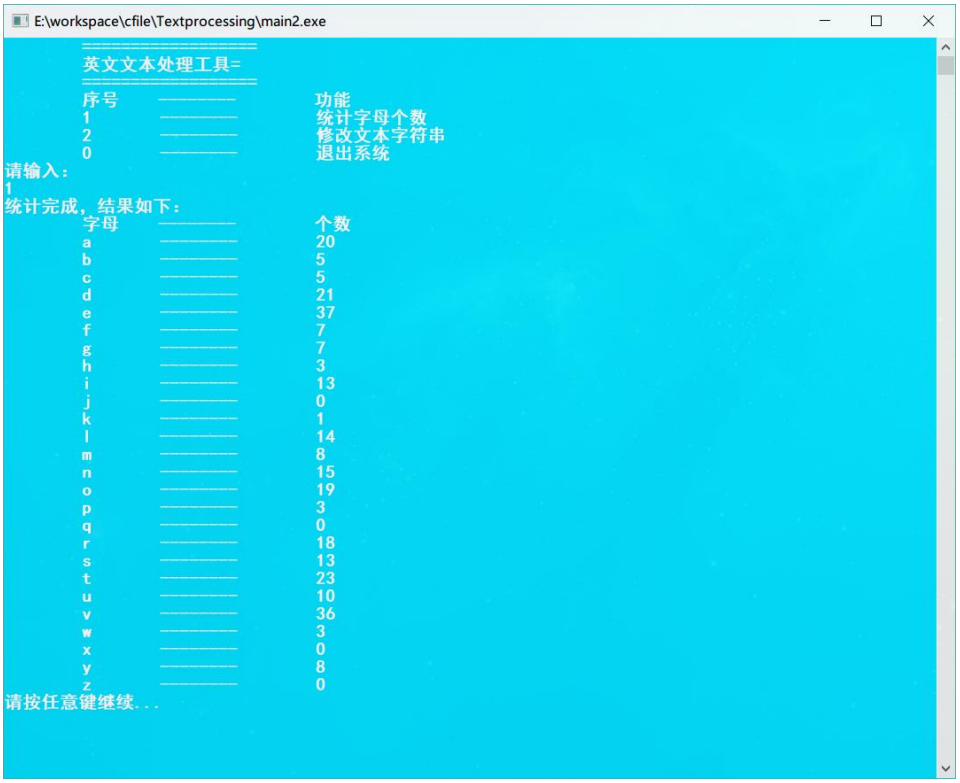
no2_2.c

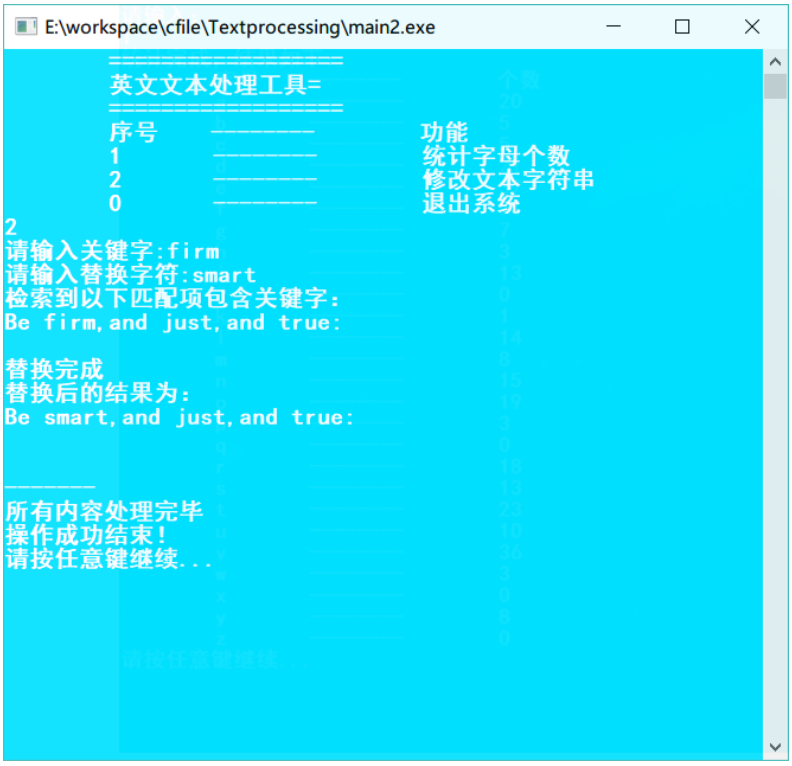
```c
void countAlpha() {
    /**
     * @brief 计算字母出现的次数
     * @author 张俊华 16030199025
     */
    FILE *fp;
    char c;
    int countlist[26] = {0};
    /*
     * countlist 存放了不同字母的字符数
     */
    fp = fopen("text.txt", "r");
    //只读打开文件
    if (fp == NULL) return;

    while (fscanf(fp, "%c", &c) != EOF) {
        if (isalpha(c)) {
            if (c >= 'a') {
                countlist[c - 'a']++;
            } else {
                countlist[c - 'A']++;
            }
        }
    }
    printf("统计完成，结果如下：\n");
    printf("\t 字母\t--------\t 个数\n");
    for (int i = 0; i < 26; ++i) {
        printf("\t%c\t--------\t%d\n", 'a' + i, countlist[i]);
    }

    fclose(fp);
}

void StrReplace(char *strSrc, char *strReplace, int position, int len) {
    /**
     * @brief 替换字符串
     * @param strSrc 原始字符串
     * @param strReplace 替换字符串
```

```
     * @param position 替换位置
     * @return <description>
     * @exception <name> <description>
     * @see <name>
     * @warning <text>
     * @author 张俊华 16030199025
     */
    char tempString[1000] = {0};
    //暂存字符串
    int oldlen = strlen(strSrc);
    memcpy(tempString, strSrc + position + len, strlen(strSrc) - position - len);

    memcpy(strSrc + position, strReplace, strlen(strReplace));
    //把替换字符串覆盖到源字符串的对应位置上
    memcpy(strSrc + position + strlen(strReplace), tempString, strlen(tempString));
    //加上原来保存的后半部分
    *(strSrc + oldlen + strlen(strReplace) - len) = '\0';
}

void replaceString() {
    /**
     * @brief 替换文件中的字符串
     * @author 张俊华 16030199025
     */

    FILE *fp;
    FILE *fptemp;
    char string[1000]; //用于读取原始字符串
    char keyword[200] = {0}; //检索字符串
    char strReplace[200]; //替换字符串

    fp = fopen("text.txt", "r");
    fptemp = fopen("temp.txt", "wt");
    if (fp == NULL) return;
    if (fptemp == NULL) return;
    //初始化文件指针

    printf("请输入关键字:");
    scanf("%s", keyword);
    printf("请输入替换字符:");
    scanf("%s", strReplace);

    //读取用户输入
    while (fgets(string, 1000, fp)) {
        //逐行处理
        for (int j = 0; j < strlen(string); j++) {
            //检索是否包含 keyword
            int findflag = 1;
            //检索状态
            for (int i = 0; i < strlen(keyword); i++) {
                if (*(string + j + i) != keyword[i]) {
                    findflag = 0;//检索到
                    break;
                }
            }
            if (findflag) {
                printf("检索到以下匹配项包含关键字：\n");
                printf("%s\n", string);

                StrReplace(string, strReplace, j, strlen(keyword));
                //替换字符串

                printf("替换完成\n");
                printf("替换后的结果为：\n");
```

```
                printf("%s\n\n-------\n", string);

            }//执行替换
        }
        fputs(string, fptemp);
    }
    printf("所有内容处理完毕\n");
    fclose(fp);
    fclose(fptemp);
    fp = fopen("text.txt", "wt");
    fptemp = fopen("temp.txt", "r");
    while (fgets(string, 1000, fptemp)) {
        fputs(string, fp);
    }
    fclose(fptemp);
    fclose(fp);
    printf("操作成功结束！\n");
}


int main(int argc, char const *argv[]) {
    int choose = 0;

    while (1) {

        printf("\t==================\n");
        printf("\t 英文文本处理工具=\n");
        printf("\t==================\n");
        printf("\t 序号\t--------\t 功能\n");
        printf("\t1\t--------\t 统计字母个数\n");
        printf("\t2\t--------\t 修改文本字符串\n");
        printf("\t0\t--------\t 退出系统\n");
        scanf("%d", &choose);
        switch (choose) {
            case 1:
                countAlpha();
                break;
            case 2:
                replaceString();
                break;
            case 0:
                return 0;
            default:
                puts("请检查输入...");
        }
        puts("请按任意键继续...");
        fflush(stdin);
        getchar();
        system("cls");

    }
}
```

测试数据（输入、输出）：


E:\workspace\cfile\Textprocessing\main2.exe
英文文本处理工具=
序号 ———— 功能
1 ———— 统计字母个数
2 ———— 修改文本字符串
0 ———— 退出系统
请输入：
1
统计完成，结果如下：
字母 ———— 个数
a ———— 20
b ———— 5
c ———— 5
d ———— 21
e ———— 37
f ———— 7
g ———— 7
h ———— 3
i ———— 13
j ———— 0
k ———— 1
l ———— 14
m ———— 8
n ———— 15
o ———— 19
p ———— 3
q ———— 0
r ———— 18
s ———— 13
t ———— 23
u ———— 10
v ———— 36
w ———— 3
x ———— 0
y ———— 8
z ———— 0
请按任意键继续. . .


E:\workspace\cfile\Textprocessing\main2.exe
英文文本处理工具=
序号 ———— 功能
1 ———— 统计字母个数
2 ———— 修改文本字符串
0 ———— 退出系统
2
请输入关键字:firm
请输入替换字符:smart
检索到以下匹配项包含关键字：
Be firm,and just,and true:
替换完成
替换后的结果为：
Be smart,and just,and true:
————
所有内容处理完毕
操作成功结束！
请按任意键继续. . .

第 3_2 题

设计一个复数类型，输入实部和虚部生成一个复数，可进行两个复数求和、两个复数求差、两个复数求积运算。

算法描述：

复数类型定义为结构体 complexNum，包含两个 double 类型变量分别为复数的实部和虚部。
定义 4 个函数，传入两个 complexNum 类型的参数，返回值为 complexNum，分别实现复数的加减乘除运算。
程序执行的过程中，调用这几个函数即可实现对复数的运算。

源程序：

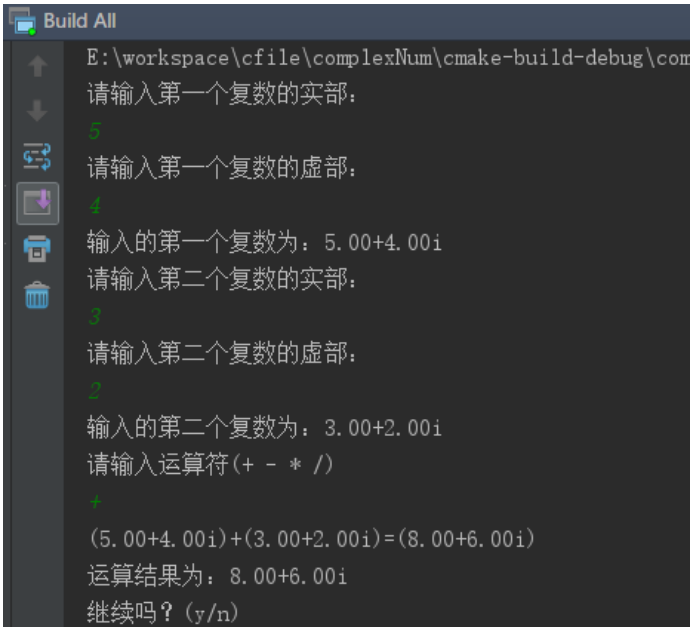
no3_2.c

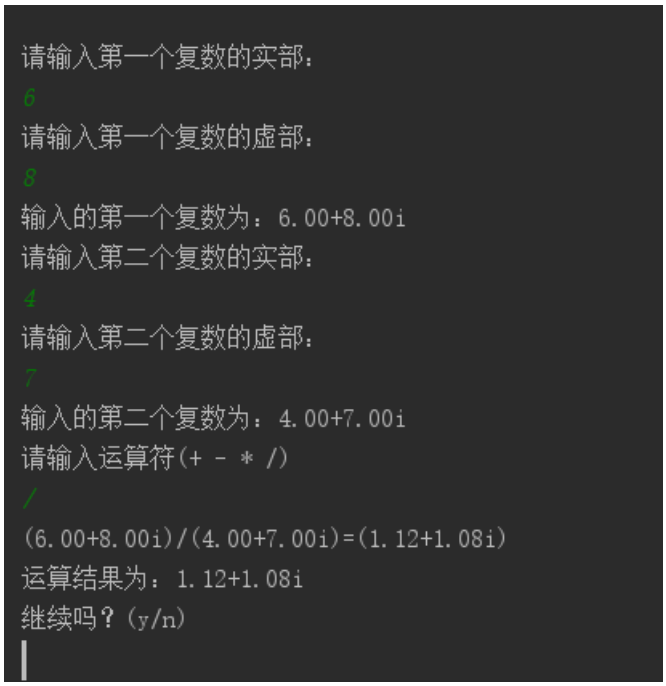
```c
typedef struct {
    double n;
    double i;
} complexNum;

complexNum add(complexNum complexNum1, complexNum complexNum2) {
    /**
     * @brief 两个复数相加
     * @param complexNum1 第一个复数
     * @param complexNum2 第二个复数
     * @return 两个复数相加的结果
     * @author 张俊华 16030199025
     */
    complexNum complexNum3;
    complexNum3.n = complexNum1.n + complexNum2.n;
    complexNum3.i = complexNum1.i + complexNum2.i;
    return complexNum3;
}

complexNum del(complexNum complexNum1, complexNum complexNum2) {
    /**
     * @brief 两个复数相减
     * @param complexNum1 第一个复数
     * @param complexNum2 第二个复数
     * @return 两个复数相减的结果
     * @author 张俊华 16030199025
     */
    complexNum complexNum3;
    complexNum3.n = complexNum1.n - complexNum2.n;
    complexNum3.i = complexNum1.i - complexNum2.i;
    return complexNum3;
}

complexNum multi(complexNum complexNum1, complexNum complexNum2) {
    /**
     * @brief 两个复数相乘
     * @param complexNum1 第一个复数
     * @param complexNum2 第二个复数
     * @return 两个复数相乘的结果
     * @author 张俊华 16030199025
     */
    complexNum complexNum3;
    complexNum3.n = (complexNum1.n * complexNum2.n) + (complexNum1.i * complexNum2.i);
    complexNum3.i = (complexNum1.n * complexNum2.i) + (complexNum2.n * complexNum1.i);
    return complexNum3;
}
```

```
complexNum div(complexNum complexNum1, complexNum complexNum2) {
    /**
     * @brief 两个复数相除
     * @param complexNum1 第一个复数
     * @param complexNum2 第二个复数
     * @return 两个复数相除的结果
     * @author 张俊华 16030199025
     */
    complexNum complexNum3;
    complexNum3.n = complexNum1.n / (complexNum2.n * complexNum2.n + complexNum2.i *
complexNum2.i)
                    * complexNum2.n
                    + complexNum2.i / (complexNum2.n * complexNum2.n + complexNum2.i *
complexNum2.i)
                      * complexNum2.i;
    complexNum3.i = complexNum1.n / (complexNum2.n * complexNum2.n + complexNum2.i *
complexNum2.i)
                    * complexNum2.i
                    + complexNum2.i / (complexNum2.n * complexNum2.n + complexNum2.i *
complexNum2.i)
                      * complexNum2.n;
    return complexNum3;
}

char *show(complexNum aComplexNum) {
    /**
     * @brief 复数转字符串
     * @param aComplexNum 复数
     * @return 复数的字符串表示
     * @author 张俊华 16030199025
     */
    static char complexNumString[50];
    sprintf(complexNumString, "%.2lf+%.2lfi", aComplexNum.n, aComplexNum.i);
    return complexNumString;
}

int main() {
    complexNum complexNum1;
    complexNum complexNum2;
    while (1) {
        puts("请输入第一个复数的实部：");
        scanf("%lf", &complexNum1.n);
        puts("请输入第一个复数的虚部：");
        scanf("%lf", &complexNum1.i);

        printf("输入的第一个复数为：%.2lf+%.2lfi\n", complexNum1.n, complexNum1.i);

        puts("请输入第二个复数的实部：");
        scanf("%lf", &complexNum2.n);
        puts("请输入第二个复数的虚部：");
        scanf("%lf", &complexNum2.i);

        printf("输入的第二个复数为：%.2lf+%.2lfi\n", complexNum2.n, complexNum2.i);

        puts("请输入运算符(+ - * /)");
        char operator;
        while (((operator = (char) getchar()) != '\n') && operator != EOF);

        operator = (char) getchar();

        switch (operator) {
            case '+':
                printf("(%s)+", show(complexNum1));
                printf("(%s)", show(complexNum2));
```

```
                printf("=(%s)\n", show(add(complexNum1, complexNum2)));
                printf("运算结果为：%s", show(add(complexNum1, complexNum2)));
                break;

            case '-':
                printf("(%s)-", show(complexNum1));
                printf("(%s)", show(complexNum2));
                printf("=(%s)\n", show(del(complexNum1, complexNum2)));
                printf("运算结果为：%s", show(del(complexNum1, complexNum2)));
                break;

            case '*':
                printf("(%s)*", show(complexNum1));
                printf("(%s)", show(complexNum2));
                printf("=(%s)\n", show(multi(complexNum1, complexNum2)));
                printf("运算结果为：%s", show(multi(complexNum1, complexNum2)));
                break;

            case '/':
                printf("(%s)/", show(complexNum1));
                printf("(%s)", show(complexNum2));
                printf("=(%s)\n", show(div(complexNum1, complexNum2)));
                printf("运算结果为：%s", show(div(complexNum1, complexNum2)));
                break;

            default:
                puts("请检查输入！");
                break;
        }
        puts("\n 继续吗？(y/n)");
        while (((operator = (char) getchar()) != '\n') && operator != EOF);
        operator = (char) getchar();
        if (operator == 'n')
            break;
    }

    return 0;
}
```

测试数据（输入、输出）：





第 4_1 题

模拟 KTV 点歌系统。用户可按歌名查找某首歌曲或按歌手名查找其所有歌曲，点歌后显示所点歌曲歌词。管理员可添加和删除歌曲，每个歌曲的歌词用一个单独的文件存储。

算法描述：

歌曲信息保存在 songlib.csv 中，管理员密码保存在 admin.db 中，歌词文件为对应歌曲名.lrc 的纯文本。

源程序：

no2_2.c

```c
/*模拟 KTV 点歌系统。用户可按歌名查找某首歌曲或按歌手名查找其所有歌曲，
点歌后显示所点歌曲歌词。管理员可添加和删除歌曲，每个歌曲的歌词用一个单独的文
        件存储。
        */

#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <ctype.h>

void Printlrc(char *song) {
    /**
     * @brief 打印歌词
     * @param song 歌曲名
     * @author 张俊华 16030199025
     */
    if(isspace(song[strlen(song) - 1]))
        song[strlen(song) - 1]='\0';
    //去掉字符串结尾的换行符
    strcat(song, ".lrc");
    //strcat 连接字符串，给字符串加上.lrc 后缀
    FILE *fp;
    fp = fopen(song, "r");
    if (fp == NULL) return;
    //初始化文件指针，只读模式
    char string[1000];
    while (fgets(string, 1000, fp)) {
        puts(string);//每读到一行就在屏幕上显示一行
        sleep(1);

    }
    puts("\n\n 按任意键继续...");
    fflush(stdin);
    getchar();
}

void ChooseSong() {
    /**
     * @brief 检索歌曲
     * @author 张俊华 16030199025
     */
    FILE *fp;
    fp = fopen("songlib.csv", "r");
    if (fp == NULL) return;
    //初始化文件指针，只读
    char keyword[100];
    printf("请输入歌手名或歌曲名: \n");
    scanf("%s", keyword);
    //检索关键字
```

```
    int index = 0;
    //结果集下标序号
    char result[1000][100];
    //结果集
    char string[1000];
    //用来保存从文件中读到的字符串
    while (fgets(string, 1000, fp)) {
        //逐行处理，检索关键字
        for (int j = 0; j < strlen(string); j++) {
            //检索是否包含 keyword
            int findflag = 1;
            //检索状态
            for (int i = 0; i < strlen(keyword); i++) {
                if (*(string + j + i) != keyword[i]) {
                    findflag = 0;//发现不匹配
                    break;
                }
            }
            if (findflag) {
                //如果匹配
                int i = 0;
                strcpy(result[++index], string);
                //把匹配的歌曲信息保存到数组中
                printf("检索到以下歌曲条目包含关键字：\n");
                printf("\t%d\t--\t", index);//输出序号
                for (i = 0; i < strlen(string); ++i) {
                    if (*(string + i) == ',') break;//输出歌手名，遇到逗号截止
                    printf("%c", *(string + i));
                }
                printf("\t---%s\n", string + i + 1);//输出歌曲名
            }
        }

    }
    if (index == 0) {
        printf("未检索到\n\n");
        return;
    }
    printf("请输入选择的序号：");
    int choose;
    scanf("%d", &choose);
    //让用户选择一个下标
    if (choose <= index) {
        printf("点歌成功！\n\n");
        Printlrc(result[index]);
        //调用 printlic，打印对应歌曲的歌词
        printf("\n\n");
    } else printf("请检查输入的内容!!\n");

}

void AddSong() {
    /*
     * 管理员函数，追加一首歌
     */
    FILE *fp;
    fp = fopen("songlib.csv", "a"); //追加打开文件
    if (fp == NULL) return;

    printf("请输入歌手名：");
    char artist[100];
    scanf("%s", artist);
    printf("请输入歌曲名：");
    char songName[100];
```

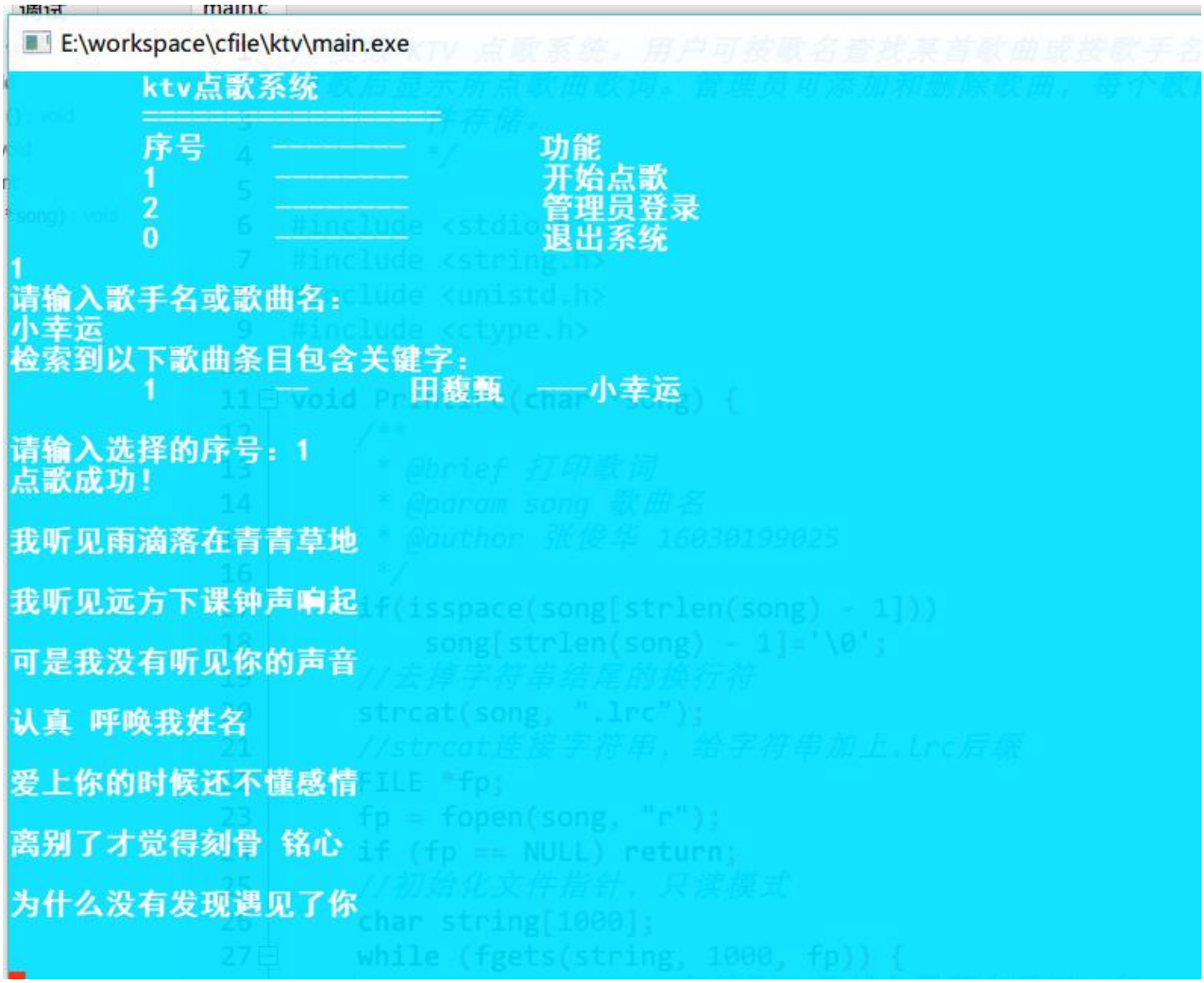
```
    scanf("%s", songName);

    fprintf(fp, "%s,%s\n", artist, songName);
    //格式化追加
    printf("增加歌曲成功\n");
    printf("请手动导入歌词文件，文件名：“歌手名,歌曲名.lrc”（注意半角逗号）\n\n");
    fclose(fp);
}

void DelSong() {
    /*
     * 管理员函数，删除歌曲
     */
    FILE *fp;
    FILE *fptemp; //中转保存，临时文件
    fptemp = fopen("temp.txt", "wt");
    fp = fopen("songlib.csv", "r");
    if (fptemp == NULL) return;
    if (fp == NULL) return;
    //初始化文件指针

    char string[100];
    int index = 0;
    printf("请输入歌手名: ");
    char artist[100];
    scanf("%s", artist);
    printf("请输入歌曲名: ");
    char songName[100];
    scanf("%s", songName);

    while (fgets(string, 1000, fp)) {
        int findArtist = 0;
        int findSongName = 0;
        for (int j = 0; j < strlen(string); j++) {
            //检索是否包含 artist
            int findArtistflag = 1;
            //检索状态
            for (int i = 0; i < strlen(artist); i++) {
                if (*(string + j + i) != artist[i]) {
                    findArtistflag = 0;
                    continue;
                }
            }
            if (findArtistflag == 1) {
                findArtist = 1;
                continue;
            }
            int findSongNameFlag = 1;
            //检索是否包含 Songname
            for (int i = 0; i < strlen(songName); i++) {
                if (*(string + j + i) != songName[i]) {
                    findSongNameFlag = 0;//检索到
                    continue;
                }
            }
            if (findSongNameFlag == 1) {
                findSongName = 1;
                continue;
            }
        }
        if (findArtist && findSongName) {
            int i = 0;
            printf("检索到以下歌曲条目包含关键字: \n");
            printf("\t%d\t--\t", ++index);
```

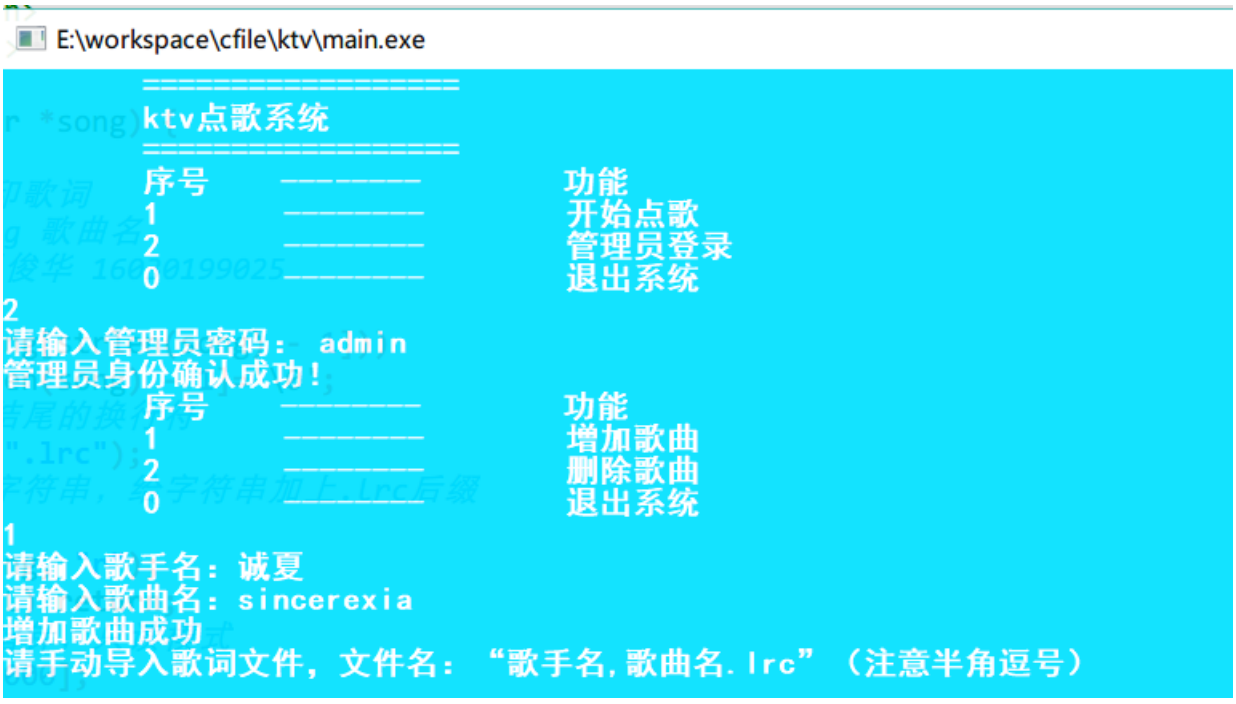
```
            for (i = 0; i < strlen(string); ++i) {
                if (*(string + i) == ',') break;
                printf("%c", *(string + i));
            }
            printf("\t---%s\n", string + i + 1);
            printf("删除吗(y/n)");
            fflush(stdin);
            char choice;
            scanf("%c", &choice);
            if (choice == 'y') continue;
        }

        fputs(string, fptemp);
    }
    fclose(fp);
    fclose(fptemp);
    fptemp = fopen("temp.txt", "r");
    fp = fopen("songlib.csv", "wt"); //
    if (fptemp == NULL) return;
    if (fp == NULL) return;
    while (fgets(string, 1000, fptemp)) {
        fputs(string, fp);
    }

    fclose(fptemp);
    fclose(fp);
    printf("\n");

}

void Admin() {
    /*
     * 管理员后台
     */
    char psw[100];
    char correctPsw[100];
    printf("请输入管理员密码: ");
    scanf("%s", psw);

    FILE *fp;
    fp = fopen("admin.db", "r");
    if (fp == NULL) return;

    fgets(correctPsw, 100, fp);
    //从文件中读取密码
    for (int i = 0; i < strlen(correctPsw); ++i) {
        //比较是否相同，发现任何一位不同直接退出
        if (psw[i] != correctPsw[i]) {
            printf("管理员密码错误! \n");
            return;
        }
    }
    printf("管理员身份确认成功! \n");
    printf("\t序号\t--------\t功能\n");
    printf("\t1\t--------\t增加歌曲\n");
    printf("\t2\t--------\t删除歌曲\n");
    printf("\t0\t--------\t退出系统\n");
    int choose;
    scanf("%d", &choose);
    switch (choose) {
        case 1:
            AddSong();
            break;
        case 2:
```

```
            DelSong();
            break;
        case 0:
            return;
        default:
            puts("请检查输入...");
            getchar();
    }
}
void ShowLib() {
    /**
     * @brief 展示所有曲库
     * @author 张俊华 16030199025
     */
    FILE *fp;
    fp = fopen("songlib.csv", "r");
    if (fp == NULL) return;
    //初始化文件指针，只读
    //检索关键字
    int index = 0;
    //结果集下标序号
    char result[1000][100];
    //结果集
    char string[1000];
    //用来保存从文件中读到的字符串
    printf("曲库包含以下歌曲条目：\n");
    while (fgets(string, 1000, fp)) {
        int i = 0;
        strcpy(result[++index], string);
        printf("\t%d\t--\t", index);//输出序号
        for (i = 0; i < strlen(string); ++i) {
            if (*(string + i) == ',') break;//输出歌手名，遇到逗号截止
            printf("%c", *(string + i));
        }
        printf("\t---%s\n", string + i + 1);//输出歌曲名
    }
}
int main(void) {
    int choose = 0;

    while (1) {

        printf("\t==================\n");
        printf("\tktv 点歌系统\n");
        printf("\t==================\n");
        printf("\t 序号\t--------\t 功能\n");
        printf("\t1\t--------\t 展示曲库\n");
        printf("\t2\t--------\t 开始点歌\n");
        printf("\t3\t--------\t 管理员登录\n");
        printf("\t0\t--------\t 退出系统\n");
        scanf("%d", &choose);
        switch (choose) {
            case 1:
                ShowLib();
                break;
            case 2:
                ChooseSong();
                break;
            case 3:
                Admin();
                break;
            case 0:
                return 0;
            default:
```

```
                puts("请检查输入...");
                getchar();
            }
        }
}
```

测试数据（输入、输出）：





第 5_1 题

程序自动生成一个位于 99 内的随机数，要求用户猜这个数。用户输入一个数后，程序有三种应答：too big, too small，you win

算法描述：

用时间做种子值，调用 rand 函数生成随机数，接收用户输入，与生成的随机数比较大小，给出提示，当与生成的随机数吻合时，给出提示，结束游戏。

源程序：

no2_2.c

```c
int main() {
    int number;
    int playerNumber;
    srand((unsigned)time(NULL));
    puts("随机数游戏开始！你准备好了吗？");
    getchar();
    while (1){
        number = rand()%100;
        while(1){
            puts("input a number:");
            scanf("%d",&playerNumber);

            if(playerNumber==number){
                puts("you Win!!");
                break;
            } else if(playerNumber>number){
                puts("too Big!!");
            } else{
                puts("too Small!!");
            }
        }
        puts("again?(y/n)");
        number = getchar();
        if(number=='n'){
            break;
        }
    }

    return 0;
}
```

测试数据（输入、输出）：

```
随机数游戏开始! 你准备好了吗?

input a number:
55
too Big! !
input a number:
34
too Big! !
input a number:
21
too Big! !
input a number:
2
too Small!!
input a number:
13
too Small!!
input a number:
19
too Big! !
input a number:
17
too Small!!
input a number:
18
you Win!!
again?(y/n)
```

第 5_2 题

产生一组随机数，要求每个数字不能重复。例如：1，20，3，17，80，4，35，88 符合要求， 3，20，1，17，80，3，35，88 不符合要求

算法描述：

定义 randomList 数列储存生成的随机数，以时间为种子值，调用 rand 函数生成随机数，每生成一个随机数则遍历一遍当前数组，检测到重复则重新生成随机数，若不重复，则加入数组。生成指定数量的随机数之后则遍历输出。

源程序：

no5_2.c

```c
int main() {

    int randomList[MAX_NUMBER]; //数列存储生成的随机数
    int n; //需要生成的随机数个数
    int min; //随机数最小值
    int max; //随机数最大值

    do {
        puts("请输入需要生成的随机数数量：");
        scanf("%d", &n);
        if(n>=0&&n<=MAX_NUMBER) {
            break;
        } else{
            puts("请检查输入的数字，请重新输入");
        }
    }while (1);

    do {
        puts("请输入需要生成的随机数最小值：");
        scanf("%d", &min);
        puts("请输入需要生成的随机数最大值：");
        scanf("%d", &max);
        if(min<max) {
            break;
        } else{
            puts("请检查输入的数字，请重新输入");
        }
    }while (1);
    while (1) {
        for (int i = 0; i < n; ++i) {

            int newRandomNumber = 0;
            srand((unsigned) time(NULL));
            newRandomNumber = (rand() % (max - min)) + min;
            int findSameFlag = 0;
            for (int j = 0; j < i; ++j) {
                if (randomList[j] == newRandomNumber) {
                    i--;
                    findSameFlag = 1;
                    break;
                    //检查到重复即重新生成
                }
            }
            if(!findSameFlag)
                randomList[i] = newRandomNumber;
            //无重复将新随机数加入数列
        }
```

```
        puts("生成了以下随机数:");
        for (int k = 0; k < n; ++k) {
            printf("%d , ", randomList[k]);
        }
        puts("继续生成吗？(y/n)");
        char choise;
        choise = getchar();
        if(choise == n){
            break;
        }

    }

}
```

测试数据（输入、输出）：