

《程序设计基础课程设计》第 1 周实验报告

班级：1803012

姓名：杨煜

学号：18030100204

所做题目：一、数值处理

第 1 题

1.原始题目及要求；

(1) 高精度计算

要求：用整型数组表示 10 进制大整数（超过 2^{32} 的整数），数组的每个元素存储大整数的一位数字，实现大整数的加减法。

2.题目的分析

(1) 涉及知识点

数组、流程控制、函数等

(2) 题目功能理解

由于数组的每个元素仅存储一位数字，所以所需要的数组类型只需使用为 `char` 型；为了表示便利，可以在存储过程中强制每一个数都以有符号数的形式存储；整数的加法可以分为符号位相同的加法和符号位相反的加法，前者只需要按位相加，而后者可以用绝对值较大者的绝对值减去较小者的绝对值，并最终取前者符号位；整数的减法可以用整数大加法来实现，这位编程实现提供了一种简单的方式。

3.题目的总体设计：设计思路、算法描述

(1) 程序模块：

(A) 初始化模块

初始化一个大整数，包括分配地址空间与初始赋值。

(B) 销毁模块

销毁一个大整数，释放空间。

(C) 赋值模块

为一个大整数赋值。

(D) 大于算符模块

无符号大整数大于运算。

(E) 加法模块

实现有符号大整数加法

(F) 减法模块

通过加法模块实现减法。

(2) 模块调用关系

减法模块调用加法模块；加法模块调用大于运算符模块和赋值模块。

(3) 输入输出数据说明

输入数据：计算功能选择，计算功能所需两个操作数。

输出数据：计算结果。

(4) 总体流程

- 1) 显示功能选择界面，提示用户选择功能，转 2。
- 2) 用户输入功能序号，用户选择功能。输入若为 '1'，则转 3；若为 '2' 则转 4 若为 '3'，则转 5；若为其他，则提示输入错误，转 2。
- 3) 用户输入两操作数，相加后显示运算结果，转 1。
- 4) 用户输入两操作数，相减后显示运算结果，转 1。
- 5) 退出程序。

4. 各功能模块/函数的设计说明：

(1) `big_int big_int_initialize(const char* val)`

函数名称: `big_int_initialize`

函数功能: 初始化一个大整数，包括分配地址空间与初始赋值

参数: `big_int obj`(保存分配地址的指针), `const char* val`(表示初始值的字符串)

返回值: `big_int`(表示结果)

执行流程:

- 1) 计算初始值字符串长，转 2。
- 2) 若初始值有符号分配长度为串长加 1 的地址空间，否则分配加 2 的地址空间
若成功转 4，失败转 3。
- 3) 返回 NULL
- 4) 若初始值无符号位则添加符号位，拷贝初始值，转 5。
- 5) 返回分配到的地址空间的首地址。

可能结果:

- 1) 分配失败，返回 NULL
- 2) 分配成功，返回大整数首地址

(2) `void big_int_destory(big_int obj)`

函数名称: `big_int_destory`

函数功能: 销毁一个大整数

参数: `big_int obj`(要销毁的大整数的首地址)

返回值: `void`

执行流程:

- 1) 释放地址空间，转 2。
- 2) 结束运行。

可能结果:

- 1) 完成销毁

(3) `big_int big_int_assign(big_int dst, const big_int src)`

函数名称: `big_int_assign`

函数功能: 为一个大整数赋值

参数: `const big_int src`(赋值来源)

返回值: `big_int`(表示结果)

执行流程:

- 1) 计算源整数的长度并加 1 (多出的一位用于存储终结符), 转 2。
- 2) 申请之前计算出的长度的地址空间, 若申请成功转 3, 否则返回 NULL。
- 3) 将源整数的内容拷贝到目的数组, 转 4。
- 4) 返回新的地址空间的首地址表示的大整数。

可能结果:

- 1) 赋值失败, 返回 NULL
- 2) 赋值成功, 返回大整数首地址

(4) `bool big_int_unsigned_greater(const big_int a, const big_int b)`

函数名称: `big_int_unsigned_greater`

函数功能: 无符号大整数大于运算

参数: `const big_int a`(大于运算左侧操作数), `const big_int b`(大于运算右侧操作数)

返回值: `bool`(表示是否大于)

执行流程:

- 1) 计算 `a` 的长度, 保存为 `sz_a`, 计算 `b` 的长度, 保存为 `sz_b`, 转 2。
- 2) 如果 `sz_a` 等于 `sz_b` 则返回 `sz_a>sz_b` 的运算结果, 否则转 3。
- 3) 由数字的高位到低位比较数字的大小, 一旦出现数字不一样的位置, 返回该位置数字比较的结果, 否则转 4。
- 4) 执行到此处证明两数相等, 放回 `false`, 结束运行。

可能结果:

- 1) `true`, 表示 `a>b`;
- 2) `false`, 表示 `a<=b`;

(5) `big_int big_int_add(const big_int src1, const big_int src2)`

函数名称: `big_int_add`

函数功能: 大整数加法

参数: `const big_int src1`(操作数 1), `const big_int src2`(操作数 2)

返回值: `big_int`(表示结果)

执行流程:

- 1) 计算 `src1` 的长度, 保存为 `sz1`, 计算 `src2` 的长度, 保存为 `sz2`, 保存两者中较大者的长度加 2 (多余的两位中一位是终结符, 另一位表示两数相加的结果的位数不会超过两者中位数大者的长度加 1) 为 `len`, 转 2。
- 2) 分配长度为 `len` 的地址空间, 若失败则返回 NULL, 否则在最后一位添加终结符后转 3。
- 3) 若 `src1` 的符号位等于 `src2` 的符号位, 转 4, 否则转 6。
- 4) 从低位到高位依次完成两操作数相加, 结果存入新的地址空间, 转 5。
- 5) 去除多余的地址空间, 返回结果, 结束运行。
- 6) 判断两数绝对值大小, 转 7。
- 7) 从低位到高位用绝对值大的数的绝对值减去绝对值较小的数的绝对值, 转 8。
- 8) 结果符号位设置为绝对值较大者的符号位, 去除多余的地址空间, 转 9
- 9) 返回结果, 结束运行。

可能结果:

- 1) 计算失败, 返回 NULL。
- 2) 计算成功, 返回大整数首地址。

(5) `big_int big_int_sub(const big_int src1, const big_int src2)`

函数名称: `big_int_sub`

函数功能: 大整数减法

参数: `const big_int src1`(被减数), `const big_int src2`(减数)

返回值: `big_int`(表示结果)

执行流程:

- 1) 计算 `src2` 的长度加 1, 保存为 `len`, 转 2。
- 2) 分配长度为 `len` 的地址空间, 若失败则返回 NULL, 否则转 3。
- 3) 将 `src2` 的内容拷贝到新的地址空间, 转 4。
- 4) 翻转新整数的符号位, 转 5。
- 5) 调用 `big_int_add` 计算原被减数和新整数加法结果, 转 6。
- 6) 返回结果, 结束。

可能结果:

- 1) 计算失败, 返回 NULL
- 2) 计算成功, 返回大整数首地址

5. 源程序:

(1) `big_int_test.c`

```
#include<stdio.h>
#include<stdbool.h>
#include"big_int.h"
int main(){
    int choice;
    big_int a, b, res;
    char t_a[512], t_b[512];
    bool is_end = false;
    while( !is_end ){
        printf("请选择功能\n");
        printf("1.加法\n2.减法\n3.退出程序\n");
        scanf("%d", &choice );
        switch( choice ){
            case 1:{
                printf("请输入两个操作数\n");
                scanf("%s %s", t_a, t_b );
                a = big_int_initialize(t_a);
                b = big_int_initialize(t_b);
                res = big_int_assign( big_int_add(a,b) );
                printf("结果为: %s\n", res );
                free( a );
                free( b );
                free( res );
                break;
            }
```

```

};
case 2:{
    printf("请输入两个操作数\n");
    scanf("%s %s", t_a, t_b );
    a = big_int_initialize(t_a);
    b = big_int_initialize(t_b);
    res = big_int_assign( big_int_sub(a,b) );
    printf("结果为: %s\n", res );
    free( a );
    free( b );
    free( res );
    break;
};
case 3:{
    printf("退出程序\n");
    is_end = true;
    break;
};
default:{
    printf("不存在此选项,请重新输入\n");
    break;
};
}
}

return 0;
}

(2) big_int.h
#ifndef __BIG_INT_H
#define __BIG_INT_H
#include<string.h>
#include<malloc.h>
#include<stdbool.h>
typedef char* big_int;// 采用字符数组存储大整数, 大整数首地址即字符指针
/*
* 函数名称: big_int_initialize
* 函数功能: 初始化一个大整数, 包括分配地址空间与初始赋值
* 参数: big_int obj( 保存分配地址的指针 ), const char* val( 表示初始值的字符串 )
* 返回值: big_int( 表示结果 )
*/
big_int big_int_initialize( const char* val ){
    size_t len = strlen( val ) + 1; // 字符数组需要多一个字符存储终结符
    bool flag = val[0] != '-' && val[0] != '+'; // 标识初始值是否有符号
    if( !flag ){

```

```

        ++len;
    }// 如果输入的无符号, 则添加上'+', 从而数组长度需要加 1
    big_int temp = (big_int)malloc( sizeof(char)*(len) );
    if( !temp ){
        return NULL;
    }// 分配空间, 若成功则继续执行, 失败则返回错误
    if( flag ){
        strcpy( temp+1, val );
        temp[0] = '+';
    }
    else{
        strcpy( temp, val );
    }// 初始输入确定拷贝起始位置
    return temp;
}

/*
 * 函数名称: big_int_destory
 * 函数功能: 销毁一个大整数
 * 参数: big_int obj( 要销毁的大整数的首地址 )
 * 返回值: void
 */
void big_int_destory( big_int obj ){
    free( obj );
    return;
}

/*
 * 函数名称: big_int_assign
 * 函数功能: 为一个大整数赋值
 * 参数: const big_int src( 赋值来源 )
 * 返回值: big_int( 表示结果 )
 */
big_int big_int_assign( const big_int src ){
    size_t len = strlen( src )+1;// 多一位存储终结符
    big_int temp = (big_int)malloc( sizeof(char)*(len) );
    if( !temp ){
        return NULL;
    }
    strcpy( temp, src );
    return temp;
}

/*

```

```

* 函数名称: big_int_unsigned_greater
* 函数功能: 无符号大整数大于运算
* 参数: const big_int a( 大于运算左侧操作数 ), const big_int b( 大于运算右侧操作数 )
* 返回值: bool( 表示是否大于 )
*/
bool big_int_unsigned_greater( const big_int a, const big_int b ){
    size_t sz_a = strlen( a ), sz_b = strlen( b );
    if( sz_a != sz_b ){
        return sz_a > sz_b;
    } // 长度不同时, 长度长的数字大
    for( size_t i=1; i<sz_a; --i ){
        if( a[i] != b[i] ){
            return a[i] > b[i];
        }
    } // 长度相同时, 最高不等位的比较结果即最终结果
    return false; // 每一位均相等, 即两数相等, 即不大于
}

/*
* 函数名称: big_int_add
* 函数功能: 大整数加法
* 参数: const big_int src1( 操作数 1 ), const big_int src2( 操作数 2 )
* 返回值: big_int( 表示结果 )
*/
big_int big_int_add( const big_int src1, const big_int src2 ){
    size_t sz1 = strlen( src1 ), sz2 = strlen( src2 ),
    len = (sz1 > sz2 ? sz1 : sz2) + 2;
    // 加法结果的位数不会超过两个操作数长度最大者加一
    big_int temp = (big_int)malloc( sizeof(char)*(len) );
    if( !temp ){
        return NULL;
    }
    temp[len-1] = '\0'; // 添加终结符
    if( src1[0] == src2[0] ){
        // 两数符号相同时, 绝对值相加, 最后添上符号位即可
        int carry_in = 0; // 进位值
        size_t index1=sz1-1, index2=sz2-1, index = len-2;
        while( index1 >= 1 && index2 >= 1 ){
            // 从低位到高位按位相加
            int sum = ( src1[index1]-'0' ) + ( src2[index2]-'0' ) + carry_in;
            carry_in = sum/10;
            temp[index] = '0' + sum%10;
            --index1;
            --index2;
        }
    }
}

```

```

        --index;
    }

    // 将长度更长的数的剩余位和进位加到结果中
    while( index1>=1 ){
        int sum = ( src1[index1]-'0' ) + carry_in;
        carry_in = sum/10;
        temp[index] = '0' + sum%10;
        --index1;
        --index;
    }
    while( index2>=1 ){
        int sum = ( src2[index2]-'0' ) + carry_in;
        carry_in = sum/10;
        temp[index] = '0' + sum%10;
        --index2;
        --index;
    }
    if( carry_in ){
        temp[index] = '1';
        --index;
    }
    temp[index--] = src1[0];// 确定符号位
    size_t gap = index + 1;// 表示最终结果相对于首地址的偏移
    if( gap ){
        //若有空余空间, 填满空间的前端
        size_t f_len = len - gap;
        for( size_t i=0; i<f_len; ++i ){
            temp[i] = temp[i+gap];
        }
    }
    temp = ( big_int )realloc( temp, len - gap );// 释放需要的地址空间
    return temp;// 赋值并返回结果
}

else{
    // 两数符号不同时, 用绝对值大的数的绝对值减绝对值小的数的绝对值,
    // 结果符号位为绝对值大的数的符号位
    int carry_in = 0;// 借位值
    bool flag = big_int_unsigned_greater( src1+1, src2+1 );
    // 确定绝对值大的是哪一个操作数
    const big_int t_src1 = flag? src1: src2;// 绝对值较大的数
    const big_int t_src2 = flag? src2: src1;// 绝对值较小的数
    size_t t_sz1 = flag? sz1: sz2;
    size_t t_sz2 = flag? sz2: sz1;

```



```

size_t index1=t_sz1-1, index2=t_sz2-1, index = len-2;
while( index1>=1 && index2>=1 ){
    int sum = ( t_src1[index1]-'0' ) - ( t_src2[index2]-'0' ) - carry_in;
    carry_in = sum < 0; // 若结果小于 0 则表示不够减, 需要借位
    temp[index] = carry_in? sum+10+'0': sum+'0';
    --index1;
    --index2;
    --index;
}
while( index1>=1 ){
    int sum = ( t_src1[index1]-'0' ) - carry_in;
    carry_in = sum < 0; // 若结果小于 0 则表示不够减, 需要借位
    temp[index] = carry_in? sum+10+'0': sum+'0';
    --index1;
    --index;
}
while( index2>=1 ){
    int sum = ( t_src1[index2]-'0' ) - carry_in;
    carry_in = sum < 0; // 若结果小于 0 则表示不够减, 需要借位
    temp[index] = carry_in? sum+10+'0': sum+'0';
    --index2;
    --index;
} // 此运算规则下运算结果一定不会向比最高位更高的位借位
temp[index--] = t_src1[0];
size_t gap = index + 1;
if( gap ){
    size_t f_len = len - gap;
    for( size_t i=0; i<f_len; ++i ){
        temp[i] = temp[i+gap];
    }
}
temp = ( big_int )realloc( temp, len - gap );
return temp;
}
}

/*
 * 函数名称: big_int_sub
 * 函数功能: 大整数减法
 * 参数: const big_int src1( 被减数 ), const big_int src2( 减数 )
 * 返回值: bool( 表示是否成功进行了运算 )
 */
big_int big_int_sub( const big_int src1, const big_int src2 ){
    size_t len = strlen( src2 )+1; // 多一位用于存储终结符

```


《程序设计基础课程设计》第2周实验报告

班级：1803012

姓名：杨煜

学号：18030100204

所做题目：二、数据结构—— 二选一

第3题

1.原始题目及要求：

(1) 复杂数据结构-动态链表

要求：链表是一种重要的数据结构，需要动态的进行存储分配，要求通过函数分别实现动态链表的建立、结点的插入、结点的删除以及链表的输出。

2.题目的分析

(1) 涉及知识点

内存管理、结构体定义、指针运用、函数

(2) 题目功能理解

为了体现链表的结点，需要用结构体实现独立的数据结构。链表实现的重点就在于链表的插入与删除，这些可以通过指针，动态内存申请与释放等完成。同时考虑到实现的方便性，选择使用一个空节点（首结点）表示一个链表。

3.题目的总体设计：设计思路、算法描述

(1) 程序模块：

(A) 初始化模块

初始化链表,即分配头结点。

(B) 插入模块

在链表的某一结点后插入一个结点。

(C) 删除模块

删除链表的某一结点后的一个结点。

(D) 遍历链表模块

遍历链表并输出所有结点的值。

(E) 销毁模块

销毁链表。

(2) 模块调用关系

各模块相互独立，无相互调用关系。

(3) 输入输出数据说明

输入数据：功能选择，功能所需的位置和值。

输出数据：操作结果，链表打印结果。

(4) 总体流程

- 1) 显示功能选择界面, 提示用户选择功能, 转 2。
- 2) 用户输入功能序号, 用户选择功能。输入若为 '1', 则转 3; 若为 '2' 则转 4 若为 '3', 则转 5; 若为 '4', 则转 6; 若为其他, 则提示输入错误, 转 1。
- 3) 用户输入位置 and 值, 显示插入结果, 转 1。
- 4) 用户输入位置, 显示删除结果, 转 1。
- 5) 打印出现在链表的内容, 转 1。
- 6) 销毁链表, 结束运行。

4. 各功能模块/函数的设计说明:

(1) `linked_list linked_list_initialize()`

函数名称: `linked_list_initialize`

函数功能: 初始化链表, 即分配头结点

参数: 无

返回值: `linked_list` (表示分配结果)

执行流程:

- 1) 分配首结点地址空间, 若分配成功, 转 2, 否则返回 `NULL`。
- 2) 使得新结点的下一个结点为 `NULL`, 转 3)
- 3) 返回结点地址。

可能结果:

- 1) 分配失败, 返回 `NULL`
- 2) 分配成功, 返回链表首地址

(2) `bool linked_list_insert(linked_list obj, size_t loc, int val)`

函数名称: `linked_list_insert`

函数功能: 在链表的某一结点后插入一个结点

参数: `linked_list obj` (链表头结点), `size_t loc` (要插入的位置的前一个结点的位置), `int val` (新结点的值)

返回值: `bool` (表示操作是否成功)

执行流程:

- 1) 寻找指定的位置, 寻找失败, 返回 `false`, 否则转 2。
- 2) 申请新的地址空间, 若申请失败, 返回 `false`, 否则转 3。
- 3) 将新结点赋值后插入指定位置, 返回 `true`。

可能结果:

- 1) `true`, 表示插入成功;
- 2) `false`, 表示插入失败

(3) `bool linked_list_delete(linked_list obj, size_t loc)`

函数名称: `linked_list_delete`

函数功能: 删除链表的某一结点后的一个结点

参数: `linked_list obj` (链表头结点),
`size_t loc` (要删除的位置的前一个结点的位置)

返回值: `bool` (表示操作是否成功)

执行流程:

- 1) 寻找指定的位置, 寻找失败, 返回 `false`, 否则转 2。
- 2) 若指定位置之后没有元素, 返回 `false`, 否则转 3。

3) 改变指针指向并删除元素, 返回 `true`。

可能结果:

1) `true`, 表示删除成功。

2) `false`, 表示删除失败。

(4) `void linked_list_print(linked_list obj)`

函数名称: `linked_list_print`

函数功能: 遍历链表并输出所有结点的值

参数: `linked_list obj`(链表头结点)

返回值: `void`

执行流程:

1) 从除头结点外的第一个结点开始打印, 直至没有下一个结点。

可能结果:

1) 完成输出

(5) `void linked_list_destory(linked_list obj)`

函数名称: `linked_list_destory`

函数功能: 销毁链表

参数: `linked_list obj`(链表头结点)

返回值:`void`

执行流程:

1) 每次均删除头结点后的首个结点, 直至除头结点外没有结点, 转 2.

2) 释放头结点内存空间。

5. 源程序:

(1) `linked_list_test.c`

```
#include<stdio.h>
#include<stdbool.h>
#include "linked_list.h"
int main(){
    int choice;
    linked_list obj = linked_list_initialize();
    bool is_end = false;
    while( !is_end ){
        printf("请选择功能\n");
        printf("1.插入节点\n2.删除节点\n3.遍历链表\n4.结束程序\n");
        scanf("%d", &choice );
        switch( choice ){
            case 1:{
                int loc, val;
                printf("请输入要在哪个位置后插入节点\n");
                scanf("%d", &loc );
                printf("请输入节点节点值\n");
                scanf("%d", &val );
                if( linked_list_insert(obj, loc, val ) ){
                    printf("插入成功\n");
                }
            }
        }
    }
}
```

```

        }
        else{
            printf("插入失败\n");
        }
        break;
    };
    case 2:{
        int loc;
        printf("请输入要在哪个位置后删除节点\n");
        scanf("%d", &loc );
        if( linked_list_delete(obj, loc) ){
            printf("删除成功\n");
        }
        else{
            printf("删除失败\n");
        }
        break;
    };
    case 3:{
        printf("链表打印结果如下\n");
        linked_list_print( obj);
        break;
    }
    case 4:{
        printf("退出程序\n");
        is_end = true;
        break;
    };
    default:{
        printf("不存在此选项,请重新输入\n");
        break;
    };
}

}

linked_list_destory( obj );
return 0;
}

(2) big_int.h
#ifndef __LINKED_LIST_H
#define __LINKED_LIST_H
#include<stdbool.h>
#include<malloc.h>

/*

```

* 链表从下标 0 开始计算下标,但下标为 0 的结点是由于代表链表的头结点,
 * 其值并无实际意义, 故有效的链表结点的下标大于等于 1
 */

```
typedef struct linked_list_node{
    int val;//节点值
    struct linked_list_node* next;//下一节点
}linked_list_node, *linked_list;// 定义链表的结点
```

```
/*
* 函数名称: linked_list_initialize
* 函数功能: 初始化链表,即分配头结点
* 参数: 无
* 返回值: linked_list( 表示分配结果 )
*/
```

```
linked_list linked_list_initialize(){
    linked_list temp =(linked_list)malloc( sizeof( linked_list_node ) );//
    if( !temp ){
        return NULL;
    }
    temp->next = NULL;
    return temp;
}
```

```
/*
* 函数名称: linked_list_insert
* 函数功能: 在链表的某一结点后插入一个结点
* 参数: linked_list obj( 链表头结点 ),size_t loc( 要插入的位置的前一个结点的位置 ),
*       int val( 新结点的值 )
* 返回值: bool( 表示操作是否成功 )
*/
```

```
bool linked_list_insert( linked_list obj ,size_t loc, int val ){
    linked_list_node* cur = obj;
    size_t cnt=0;
    while( cur && cnt<loc ){
        cur = cur->next;
        ++cnt;
    }
    if( cnt<loc || !cur ){
        return false;
    }
    linked_list temp =(linked_list)malloc( sizeof( linked_list_node ) );
    if( !temp ){
        return false;
    }
}
```

```

    }
    temp->val = val;
    temp->next = cur->next;
    cur->next = temp;
    return true;
}

/*
 * 函数名称: linked_list_delete
 * 函数功能: 删除链表的某一结点后的一个结点
 * 参数: linked_list obj( 链表头结点 ),size_t loc( 要删除的位置的前一个结点的位置 ),
 * 返回值: bool( 表示操作是否成功 )
 */
bool linked_list_delete( linked_list obj ,size_t loc ){
    linked_list_node* cur = obj;
    size_t cnt=0;
    while( cur && cnt<loc ){
        cur = cur->next;
        ++cnt;
    }
    if( cnt<loc || ( cur && !cur->next ) || !cur ){
        return false;
    }
    linked_list_node* temp = cur->next->next;
    free( cur->next );
    cur->next = temp;
    return true;
}

/*
 * 函数名称: linked_list_print
 * 函数功能: 遍历链表并输出所有结点的值
 * 参数: linked_list obj( 链表头结点 )
 * 返回值: void
 */
void linked_list_print( linked_list obj ){
    linked_list_node* cur = obj->next;
    while( cur ){
        printf("%d ",cur->val);
        cur = cur->next;
    }
    printf("\n");
    return;
}

```



```

/*
 * 函数名称: linked_list_destory
 * 函数功能: 销毁链表
 * 参数: linked_list obj( 链表头结点 )
 * 返回值: void
 */
void linked_list_destory( linked_list obj ){
    linked_list_node* temp;
    while( temp = obj->next ){
        obj->next = obj->next->next;
        free( temp );
    }
    free( obj );
    return;
}
#endif

```

6. 测试数据（输入、输出）:

见最后一张图

7. 小结:

本次实验圆满完成，实验结果符合实验要求。

实验过程中，我复习到了 c 语言中内存管理、结构体定义、指针运用、函数等重要知识，收货颇丰。

```
请选择功能
1.插入节点
2.删除节点
3.遍历链表
4.结束程序
1
请输入要在哪个位置后插入节点
0
请输入节点节点值
0
插入成功
请选择功能
1.插入节点
2.删除节点
3.遍历链表
4.结束程序
1
请输入要在哪个位置后插入节点
0
请输入节点节点值
2
插入成功
请选择功能
1.插入节点
2.删除节点
3.遍历链表
4.结束程序
1
请输入要在哪个位置后插入节点
0
请输入节点节点值
3
插入成功
请选择功能
1.插入节点
2.删除节点
3.遍历链表
4.结束程序
3
链表打印结果如下
3 2 0
请选择功能
1.插入节点
2.删除节点
3.遍历链表
4.结束程序
2
请输入要在哪个位置后删除节点
1
删除成功
请选择功能
1.插入节点
2.删除节点
3.遍历链表
4.结束程序
3
链表打印结果如下
3 0
```

《程序设计基础课程设计》第3周实验报告

班级：1803012

姓名：杨煜

学号：18030100204

所做题目：三、文件处理—— 二选一

第5题

1.原始题目及要求：

(1) 位图图像文件缩放

要求：编写一个程序，可以在命令行输入参数，完成指定文件的缩放，并存储到新文件，命令行参数如下

`zoom file1.bmp 200 file2.bmp`

第一个参数为可执行程序名称，第二个参数为原始图像文件名，第三个参数为缩放比例（百分比），第四个参数为新文件名

2.题目的分析

(1) 涉及知识点

文件读写、结构体定义、内存管理、基本图像处理算法、命令行参数等

(2) 题目功能理解

由于文图文件特有的图片阻止形式，所以需要定义特定的结构体来存储文图文件的头部信息。而位图文件的放缩则可以采用简单的线性插值算法实现。

3.题目的总体设计：设计思路、算法描述

(1) 程序模块：

(A) 图片放缩模块

实现位图文件的放缩。

(2) 模块调用关系

模块间无相互调用关系。

(3) 输入输出数据说明

输入数据：形如 `zoom file1.bmp 200 file2.bmp` 的命令行命令。

输出数据：放缩后的位图。

(4) 总体流程

1) 命令行输入形如的命令 `zoom file1.bmp 200 file2.bm`，若成功则显示成功，并生成指定的文件，若失败则提示失败。

4.各功能模块/函数的设计说明：

(1) `bool bmp_zoom(const char *src, const char *dst , int percent)`

函数名称: `bmp_zoom`

函数功能: 图片放大缩小

参数: const char *src(要被放缩的图片名称), const char *dst(放缩结果的名称),
int percent(放缩的百分比)

返回值: bool(表示操作是否存在成功)

执行流程:

- 1) 打开源图片并创建目标图片, 成功则转 2, 若失败则返回 false。
- 2) 获取源图片文件头部信息转 3.
- 3) 获取原图片的位图数据转 4。
- 4) 修改原照片的宽高, 并将修改过的头部信息写入新图片中, 转 5.
- 5) 创建存储图片内容的缓冲区, 若失败返回 false, 否则转 6.
- 6) 采用线性插值法进行图片放缩, 结果写入文件, 释放堆空间, 关闭文件, 返回 true。

可能结果:

- 1) 放缩失败, 返回 false。
- 2) 放缩成功, 返回 true。

5. 源程序:

(1) bmp_zoom_test.c

```
#include "bmp_zoom.h"
#include "stdio.h"
int main( int argc, char* argv[] ){
    if( argc != 4 ){
        printf("参数错误\n");
        return 0;
    }
    char *src = argv[1];
    int percent = atoi( argv[2] );
    char *dst = argv[3];
    printf("%s", bmp_zoom( src, dst, percent )? "放缩成功\n":"放缩失败\n");
    return 0;
}
```

(2) bmp_zoom.h

```
#ifndef __BMP_ZOOM_H
#define __BMP_ZOOM_H
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
#define COLOR_WIDTH 3
#pragma pack(1) /* 必须在结构体定义之前使用,这是为了让结构体中各成员按 1 字节对齐*/
typedef struct tagBITMAPFILEHEADER {
    unsigned short bfType; //保存图片类型。 'BM'
    unsigned int bfSize; //位图文件的大小, 以字节为单位 (3-6 字节, 低位在前)
```

```

    unsigned short bfReserved1; //位图文件保留字, 必须为 0(7-8 字节)
    unsigned short bfReserved2; //位图文件保留字, 必须为 0(9-10 字节)
    unsigned int bfOffBits; // RGB 数据偏移地址,位图数据的起始位置, 以相对于位图 (11-14 字节, 低位在前)
} BITMAPFILEHEADER;

```

```

typedef struct tagBITMAPINFOHEADER {
    unsigned int biSize; //本结构所占用字节数 (15-18 字节)
    unsigned int biWidth; //位图的宽度, 以像素为单位 (19-22 字节)
    unsigned int biHeight; //位图的高度, 以像素为单位 (23-26 字节)
    unsigned short biPlanes; //目标设备的级别, 必须为 1(27-28 字节)
    unsigned short biBitCount; //每个像素所需的位数, 必须是 1 (双色) (29-30 字节), 4(16 色), 8(256 色) 16(高彩色)或 24 (真彩色) 之一
    unsigned int biCompression; //位图压缩类型, 必须是 0 (不压缩), (31-34 字节)
        // 1(BI_RLE8 压缩类型) 或 2(BI_RLE4 压缩类型) 之一
    unsigned int biSizeImage; //位图的大小(其中包含了为了补齐行数是 4 的倍数而添加的空字节), 以字节为单位 (35-38 字节)
    unsigned int biXPelsPerMeter; //位图水平分辨率, 每米像素数 (39-42 字节)
    unsigned int biYPelsPerMeter; //位图垂直分辨率, 每米像素数 (43-46 字节)
    unsigned int biClrUsed; //位图实际使用的颜色表中的颜色数 (47-50 字节)
    unsigned int biClrImportant; //位图显示过程中重要的颜色数 (51-54 字节)
} BITMAPINFOHEADER;

```

```

/*
* 函数名称: bmp_zoom
* 函数功能: 图片放大缩小
* 参数: const char *src( 要被放缩的图片名称 ), const char *dst( 放缩结果的名称 ),
*       int percent( 放缩的百分比 )
* 返回值: bool( 表示操作是否存在成功 )
*/

```

```

bool bmp_zoom( const char *src, const char *dst, int percent ){
    FILE *fpsrc = fopen(src, "rb");// 打开源图片
    FILE *fpdst = fopen(dst, "wb");// 创建目标图片
    if( !fpsrc || !fpdst ){// 打开失败
        return false;
    }
    BITMAPFILEHEADER head;
    BITMAPINFOHEADER info;
    memset(&head, 0, sizeof(BITMAPFILEHEADER));
    memset(&info, 0, sizeof(BITMAPINFOHEADER));
    fread(&head, sizeof(BITMAPFILEHEADER), 1, fpsrc);
    fread(&info, sizeof(BITMAPINFOHEADER), 1, fpsrc);
    // 获取源图片文件头部信息
    unsigned int src_width = info.biWidth; //获取原图片的宽

```

```

unsigned int src_height = info.biHeight; //获取原图片的高

//获取原图片的位图数据
unsigned char *src_data = (unsigned char *)malloc(src_width * src_height *
COLOR_WIDTH);
fseek(fpsrc, 54, SEEK_SET);
fread(src_data, src_width * src_height * COLOR_WIDTH, 1, fpsrc);

//修改原照片的宽高
double scale = (double)percent/100;
unsigned int dst_width = src_width * scale ;
unsigned int dst_height = src_height * scale;
double rate_y = (double)src_height/dst_height;
double rate_x = (double)src_width/dst_width;
head.bfSize = dst_width * dst_height * COLOR_WIDTH + 54;
info.biWidth = dst_width;
info.biHeight = dst_height;
//将修改过的头信息写进新照片
fwrite(&head, sizeof(BITMAPFILEHEADER), 1, fpdst);
fwrite(&info, sizeof(BITMAPINFOHEADER), 1, fpdst);

unsigned char *dst_data = (unsigned char *)malloc(dst_width * dst_height *
COLOR_WIDTH);
//创建存储图片内容的缓冲区
if( !dst_data ){
    return false;
}

//采用线性插值法进行图片放缩
for(unsigned int dst_y = 0; dst_y < dst_height; ++dst_y){
    unsigned int src_y = (unsigned int)(rate_y * dst_y);
    for(unsigned int dst_x = 0; dst_x < dst_width; ++dst_x){
        unsigned int src_x = (unsigned int)(rate_x * dst_x);
        memcpy(
            dst_data+(dst_y*dst_width+dst_x)*COLOR_WIDTH,
src_data+(src_y*src_width+src_x)*COLOR_WIDTH, COLOR_WIDTH );
    }
}

//结果写入文件
fseek(fpdst, 54, SEEK_SET);
fwrite(dst_data, dst_width * dst_height * 3, 1, fpdst);

//释放堆空间
free(dst_data);

```

```
    free(src_data);  
    //关闭文件  
    fclose(fp-src);  
    fclose(fpdst);  
  
    return true;  
}  
#endif
```

6. 测试数据（输入、输出）:

```
PS F:\MyProgram\c_lab> .\exe\bmp_zoom_test.exe .\src\bmp_zoom\24.bmp 50 .\src\bmp_zoom\res.bmp  
放缩成功  
PS F:\MyProgram\c_lab> .\exe\bmp_zoom_test.exe .\src\bmp_zoom\24.bmp 200 .\src\bmp_zoom\res.bmp  
放缩成功
```

下面两图分别是原图和缩小后的图片，放大图片由于过大无法在文档中显示。

杨煜

杨煜

7. 小结:

本次实验圆满完成，实验结果符合实验要求。

实验过程中，我复习到了c语言中文件读写、结构体定义、内存管理、基本图像处理算法、命令行参数等重要知识，收货颇丰。

但此次实验还并不完美，由于浮点数与整型之间的舍入问题，导致个小数倍的放缩在某些特定情况下会出现不完美的结果。

《程序设计基础课程设计》第 4 周实验报告

班级：1803012

姓名：杨煜

学号：18030100204

所做题目：四、算法设计—— 二选一

第 6 题

1.原始题目及要求；

(1) 快速排序算法

要求：编写一个程序，对用户输入的若干整数，采用快速排序算法，完成从小到大的排序。

2.题目的分析

(1) 涉及知识点

数组、快速排序算法等

(2) 题目功能理解

快速排序采用分治的思想有效的降低了排序的平均时间复杂度，具体策略为将原序列划分为大于某值和小于某值的两部分。之后再对这两部分再执行快速排序。当排序规模小于 10 时，快速排序的效果并不如简单的插入排序，故此时选用插入排序。

3.题目的总体设计：设计思路、算法描述

(1) 程序模块：

(A) 交换模块

交换两个个指针指向的对象的内容。

(B) 插入排序模块

插入排序。

(C) 划分模块

划分数组。

(D) 快速排序模块

快速排序。

(2) 模块调用关系

快速排序模块调用划分模块，插入排序模块，快速排序模块；划分模块调用交换模块。

(3) 输入输出数据说明

输入数据：无。

输出数据：排序结果。

(4) 总体流程

- 1) 获取一段随机数组成的整数序列，转 2。
- 2) 输出原序列，转 3。
- 4) 使用快速排序排序原序列，转 4。
- 5) 输出排序后的原序列，并输出结果。若已执行足够次数，则转 5，否则转 1。
- 6) 结束，返回。

4. 各功能模块/函数的设计说明：

(1) void swap(int *a, int *b)

函数名称: swap

函数功能: 交换两个个指针指向的对象的内容

参数: int *a(操作数 1), int *b(操作数 2)

返回值: void

执行流程:

- 1) 交换指针中对象，返回。

可能结果:

- 1) 交换完成

(2) void insertSort(int nums[], int start, int end)

函数名称: insertSort

函数功能: 插入排序

参数: int nums[(待排序数组首地址), int start(开始排序的位置),
int end(结束排序的位置加 1)

返回值: void

执行流程:

- 1) 插入排序，返回。

可能结果:

- 1) 排序完成

(3) int partiton(int nums[], int start, int end)

函数名称: partiton

函数功能: 划分数组

参数: int nums[(待划分数组首地址), int start(开始划分的位置),
int end(结束划分的位置加 1)

返回值: int(表示大小位于中间的数的位置)

执行流程:

- 1) 使得第一个元素是左中右中大小中等的，转 2。
- 2) 执行划分，转 3。
- 3) 把首元素填入对应位置，转 4。
- 4) 返回现在首元素的位置。

可能结果:

- 1) 完成划分

(4) void quick_sort(int nums[], int start, int end)

函数名称: quick_sort

函数功能: 快速排序

参数: int nums[(待排序数组首地址), int start(开始排序的位置),
int end(结束排序的位置加 1)

返回值: void

执行流程:

- 1) 若待排序列大小小于 1 则返回, 否则转 2。
- 2) 若待排序列大小小于 10 转 3, 否则转 4。
- 3) 执行插入排序, 返回。
- 4) 执行划分, 分别用快速排序排序前、后半段, 返回。

可能结果:

- 1) 完成排序

5. 源程序:

(1) quick_sort_test.c

```
#include<stdio.h>
#include<time.h>
#include<stdlib.h>
#include"quick_sort.h"

int main(){
    srand( time( NULL ) );
    int org_num[20];
    for( int i=0; i<5; ++i ){
        for( int j=0; j<20; ++j ){
            org_num[j] = rand();
        } // 获取 20 个随机数
        printf("未排序的序列为: \n");
        for( int j=0; j<20; ++j ){
            printf("%d ", org_num[j] );
        } // 显示未排序的序列
        printf("\n");
        quick_sort( org_num, 0, 20 );
        printf("排序后的序列为\n");
        for( int j=0; j<20; ++j ){
            printf("%d ", org_num[j] );
        } // 显示已排序的序列
        printf("\n\n");
    }
    return 0;
}
```

(2) quick_sort.h

```
#ifndef __QUICK_SORT_H
#define __QUICK_SORT_H

#include<malloc.h>
```

```

/*
 * 函数名称: swap
 * 函数功能: 交换两个个指针指向的对象的内容
 * 参数: int *a( 操作数 1 ), int *b( 操作数 2 )
 * 返回值: void
 */
void swap( int *a, int *b){
    int temp = *a;
    *a=*b;
    *b=temp;
    return;
}
/*
 * 函数名称: insertSort
 * 函数功能: 插入排序
 * 参数: int nums[( 待排序数组首地址 ), int start( 开始排序的位置 ),
 *       int end( 结束排序的位置加 1 )
 * 返回值: void
 */
void insertSort( int nums[], int start, int end){
    for( int i=start+1; i<end; i++ ){
        if( nums[i]<nums[i-1] ){
            int temp = nums[i];
            int j;
            for( j=i-1; j>=0; j-- ){
                if( nums[j]>temp ){
                    nums[j+1]=nums[j];
                }
                else{
                    break;
                }
            }
            nums[j+1] = temp;
        }
    }
}
/*
 * 函数名称: partiton
 * 函数功能: 划分数组
 * 参数: int nums[( 待划分数组首地址 ), int start( 开始划分的位置 ),
 *       int end( 结束划分的位置加 1 )
 * 返回值: int( 表示大小位于中间的数的位置 )
 */

```

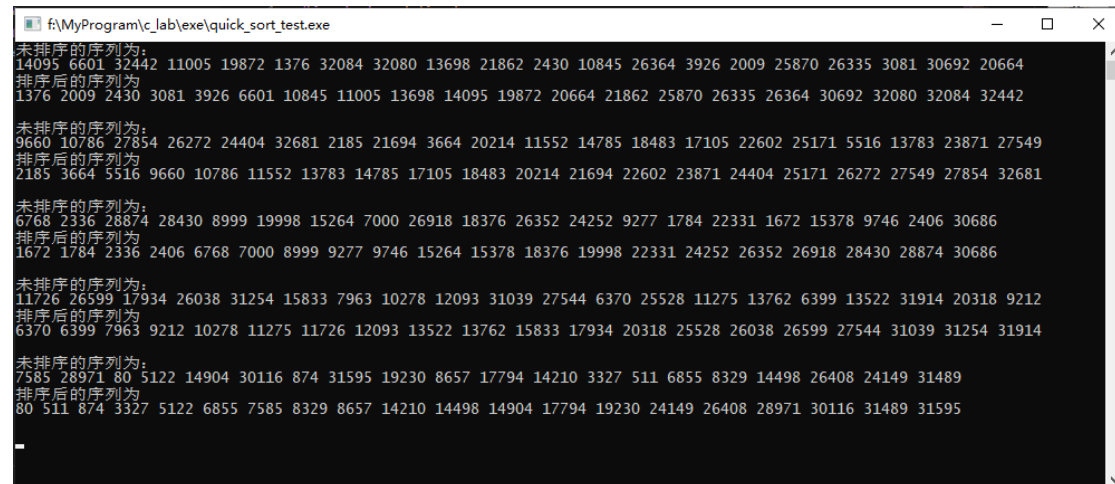
```

int partiton( int nums[], int start, int end ){
    int mid= (start+end-1)/2;
    if( nums[start]>nums[end-1] ){
        swap( &nums[start], &nums[end-1] );
    }
    if( nums[mid]>nums[end-1] ){
        swap( &nums[end-1], &nums[mid] );
    }
    if( nums[mid]>nums[start] ){
        swap( &nums[start], &nums[mid] );
    }// 使得第一个元素是左中右中大小中等的, 这样有助于提升性能
    int flag = nums[start];
    int left=start;
    int right=end-1;
    while( left<right ){
        while( left<right && nums[right]>=flag ){ right--; }
        nums[left]=nums[right];
        while( left<right && nums[left]<=flag ){ left++; }
        nums[right]=nums[left];
    }//执行划分
    nums[left]=flag;//把首元素填入对应位置
    return left;
}
/*
* 函数名称: quick_sort
* 函数功能: 快速排序
* 参数: int nums[( 待排序数组首地址 ), int start( 开始排序的位置 ),
*       int end( 结束排序的位置加 1 )
* 返回值: void
*/
void quick_sort( int nums[], int start, int end ){
    if( start<end-1 ){
        if( end-start>10 ){
            // 若排序范围大于 10, 则选择快速排序
            int mid = partiton( nums, start, end );//划分待排序列
            quick_sort( nums, start, mid );// 排序划分后的左半部分
            quick_sort( nums, mid+1, end );// 排序划分后的右半部分
        }
        else {
            // 否则选择插入排序
            insertSort( nums, start, end );
        }
    }
}
}

```

#endif

6. 测试数据（输入、输出）：



```
f:\MyProgram\c_lab\exe\quick_sort_test.exe
未排序的序列为:
14095 6601 32442 11005 19872 1376 32084 32080 13698 21862 2430 10845 26364 3926 2009 25870 26335 3081 30692 20664
排序后的序列为
1376 2009 2430 3081 3926 6601 10845 11005 13698 14095 19872 20664 21862 25870 26335 26364 30692 32080 32084 32442

未排序的序列为:
9660 10786 27854 26272 24404 32681 2185 21694 3664 20214 11552 14785 18483 17105 22602 25171 5516 13783 23871 27549
排序后的序列为
2185 3664 5516 9660 10786 11552 13783 14785 17105 18483 20214 21694 22602 23871 24404 25171 26272 27549 27854 32681

未排序的序列为:
6768 2336 28874 28430 8999 19998 15264 7000 26918 18376 26352 24252 9277 1784 22331 1672 15378 9746 2406 30686
排序后的序列为
1672 1784 2336 2406 6768 7000 8999 9277 9746 15264 15378 18376 19998 22331 24252 26352 26918 28430 28874 30686

未排序的序列为:
11726 26599 17934 26038 31254 15833 7963 10278 12093 31039 27544 6370 25528 11275 13762 6399 13522 31914 20318 9212
排序后的序列为
6370 6399 7963 9212 10278 11275 11726 12093 13522 13762 15833 17934 20318 25528 26038 26599 27544 31039 31254 31914

未排序的序列为:
7585 28971 80 5122 14904 30116 874 31595 19230 8657 17794 14210 3327 511 6855 8329 14498 26408 24149 31489
排序后的序列为
80 511 874 3327 5122 6855 7585 8329 8657 14210 14498 14904 17794 19230 24149 26408 28971 30116 31489 31595
```

7. 小结：

本次实验圆满完成，实验结果符合实验要求。

实验过程中，我复习到了c语言中数组、快速排序算法等重要知识，收货颇丰。

《程序设计基础课程设计》第 5 周实验报告

班级：1803012

姓名：杨煜

学号：18030100204

所做题目：五、综合系统——简单文件数据库

第 9 题

1.原始题目及要求；

(1) 模拟 KTV 点歌系统

要求：用户分为管理员和客户两类，分别显示不同文本格式菜单，通过菜单项对应数字进行选择。客户菜单包括查找及点歌，可按歌名查找同名所有歌曲并显示相关信息，或按歌手名查找其所有歌曲，点歌后显示所点歌曲歌词。管理员菜单包括可添加和删除歌曲，每个歌曲的歌词用一个单独的文件存储。

命令行参数如下：

`KTVsim -a(-u) xxxx`

第一个参数为可执行程序名称；第二个参数为用户身份，-a 表示管理员，-u 表示客户；按管理员和客户不同，显示对应的操作菜单；

2.题目的分析

(1) 涉及知识点

文件读写、内存管理、结构体定义、基本数据结构、高级格式化输入输出

(2) 题目功能理解

本题的主要难点在于文件的读写，在设计过程中应随时注意文件的打开模式，及文件的关闭时机。

3.题目的总体设计：设计思路、算法描述

(1) 程序模块：

(A) 添加歌曲模块

为 ktv 添加歌曲。

(B) 删除歌曲模块

为 ktv 删除歌曲。

(C) 管理员模式模块

打开管理员模式。

(D) 查找歌手模块

查找某一歌手的所演唱的歌曲。

(E) 查找同名歌曲模块

查找同名的所有歌曲。

(F) 用户查找歌曲选项的二级界面模块

打开用户查找歌曲选项的二级界面。

(G) 播放模块

播放歌曲。

(H) 用户模式模块

打开用户模式。

(I) 开始模块

开始使用 ktv

(2) 模块调用关系

开始模块调用用户模式模块，管理员模式模块；用户模式模块调用播放模块，用户查找歌曲选项的二级界面模块；用户查找歌曲选项的二级界面模块调用查找歌手模块，查找同名歌曲模块；管理员模式模块调用添加歌曲模块，删除歌曲模块。

(3) 输入输出数据说明

输入数据：形如 KTVsim -a(-u) xxxx 的命令行命令。

输出数据：操作结果。

(4) 总体流程

输入命令行参数后根据提示进行操作即可。

4. 各功能模块/函数的设计说明：

(1) void admin_add_song()

函数名称: admin_add_song

函数功能: 为 ktv 添加歌曲

参数: 无

返回值: void

执行流程:

- 1) 打开现有歌曲列表，若成功转 2，否则提示错误后返回。
- 2) 提示用户输入歌曲名，作者名，读取后转 3。
- 3) 与现存歌曲列表中的结果逐一比对，若找到对应的歌曲，则设置存在标志并退出比较，转 4。
- 4) 如果歌曲已经存在，则提示已存在，返回。否则转 5。
- 5) 以附加写入方式重新打开文件，准备添加歌曲。转 6。
- 6) 创建新歌曲的歌词文件，成功转 7，否则，提示错误并返回。
- 7) 打开新歌曲的歌词文件，并读入歌词。转 8。
- 8) 关闭打开的各文件，返回。

可能结果:

- 1) 添加歌曲成功
- 2) 添加歌曲失败

(2) void admin_delete_song()

函数名称: admin_delete_song()

函数功能: 为 ktv 删除歌曲

参数: 无

返回值: void

执行流程:

- 1) 打开现有歌曲列表，并创建新的歌曲列表，若成功转 2，否则提示错误后返回。

- 2) 提示用户输入歌曲名, 作者名, 读取后转 3。
- 3) 与现存歌曲列表中的结果逐一比对, 若找到对应的歌曲, 则不写入新的歌曲列表, 否则写入, 转 4。
- 4) 移除旧歌曲列表, 转 5。
- 5) 移除要删除的歌曲文件的歌词文件。转 6。
- 6) 重命名新的歌曲列表的文件名, 转 7。
- 7) 提示操作成功, 返回。

可能结果:

- 1) 删除歌曲成功
- 2) 删除歌曲失败

(3) void admin()

函数名称: admin

函数功能: 打开管理员模式

参数: 无

返回值: void

执行流程:

- 1) 显示操作界面, 转 2。
- 2) 读取用户操作选项, 若非退出, 根据选项做出相应操作, 否则, 转 3。
- 3) 返回。

可能结果:

- 1) 执行相关操作。

(4) void user_find_song_name()

函数名称: user_find_song_name

函数功能: 查找同名的所有歌曲

参数: 无

返回值: void

执行流程:

- 1) 打开现有歌曲列表, 若成功转 2, 否则提示错误后返回。
- 2) 提示用户输入歌曲名, 读取后转 3。
- 3) 与现存歌曲列表中的结果逐一比对, 若找到对应的歌曲, 则予以显示
否则, 不显示, 转 4。
- 4) 提示查询结束, 转 5。
- 5) 关闭打开的各文件, 返回。

可能结果:

- 1) 查找成功, 显示结果。
- 2) 查找失败, 提示失败。

(5) void user_find_song_singer()

函数名称: user_find_song_singer

函数功能: 查找某一歌手的所演唱的歌曲

参数: 无

返回值: void

执行流程:

- 1) 打开现有歌曲列表, 若成功转 2, 否则提示错误后返回。
- 2) 提示用户输入歌手名, 读取后转 3。

- 3) 与现存歌曲列表中的结果逐一比对, 若找到对应的歌手, 则予以显示
否则, 不显示, 转 4。
- 4) 提示查询结束, 转 5。
- 5) 关闭打开的各文件, 返回。

可能结果:

- 1) 查找成功, 显示结果。
- 2) 查找失败, 提示失败。

(5) void user_find_song()

函数名称: user_find_song

函数功能: 用户查找歌曲选项的二级界面

参数: 无

返回值: void

执行流程:

- 1) 显示操作界面, 转 2。
- 2) 读取用户操作选项, 若非退出, 根据选项做出相应操作, 否则, 转 3。
- 3) 返回。

可能结果:

- 1) 执行相关操作

(6) void user_play_song()

函数名称: user_play_song

函数功能: 播放歌曲

参数: 无

返回值: void

执行流程:

- 1) 打开现有歌曲列表, 若成功转 2, 否则提示错误后返回。
- 2) 提示用户输入歌曲名, 作者名, 读取后转 3。
- 3) 与现存歌曲列表中的结果逐一比对, 若找到对应的歌曲, 则设置存在标志并退出比较, 转 4。
- 4) 如果歌曲不存在, 则提示不存在, 返回。否则转 5。
- 5) 打开歌曲歌词文件, 若失败则提示并返回。否则转 6。
- 6) 逐行显示歌词内容, 转 7。
- 7) 提示点歌成功, 关闭打开的文件, 返回。。

可能结果:

- 1) 播放歌曲成功
- 2) 播放歌曲失败

(7) void ktv_start(char user_level, const char *name)

函数名称: ktv_start

函数功能: 开始使用 ktv

参数: char user_level(操作者的身份), const char *name(操作者的姓名)

返回值: void

执行流程:

- 1) 根据用户身份打开对应列表。若身份不存在, 则直接返回。否则转 2。
- 2) 查找对应用户是否存在, 若不存在直接返回, 否则转 3。

3) 显示操作界面, 转 4.

4) 读取用户操作选项, 若非退出, 根据选项做出相应操作, 否则, 转 5。

5) 结束

可能结果:

1) 执行相关操作

5. 源程序:

(1) ktv.c

```
#include"ktv.h"
#include<stdio.h>
#include<stdlib.h>
int main( int argc, char* argv[] ){
    if( argc != 3 ){
        printf("参数错误\n");
        return 0;
    }
    ktv_start( argv[1][1], argv[2] );
    return 0;
}
```

(2) ktv.h

```
#ifndef __KTV_H
#define __KTV_H
#include<stdio.h>
#include<string.h>
#include<stdbool.h>
/* 管理员操作的相关模块 */

/*
 * 函数名称: admin_add_song
 * 函数功能: 为 ktv 添加歌曲
 * 参数: 无
 * 返回值: void
 */
void admin_add_song(){
    FILE *songs_list = fopen("/home/yyang/MyProgram/c_lab/src/ktv/songs", "r");
    if( !songs_list ){
        printf( "打开歌单出错, 目前无法进行添加操作\n");
        return;
    } // 打开现有歌曲列表
    printf("\n=====添加歌曲=====\n");
    char name[512]; // 输入的歌曲名
    char singer[512]; // 输入的作者名
    char cur_name[512]; // 现在遍历到的歌曲名
```

```

char cur_singer[512];// 现在遍历到的作者名
bool is_existed = false;// 表示该作者的该歌曲已经在系统中
printf("请输入你要添加的歌曲的名称\n");
scanf( "%s", name );
printf("请输入该歌曲的歌手名称\n");
scanf( "%s", singer );
while( fscanf(songs_list, "%s", cur_name) != EOF && fscanf(songs_list, "%s", cur_singer )){
    // 与现存歌曲列表中的结果逐一比对
    // 若找到对应的歌曲, 则设置存在标志并退出比较
    if( strcmp( cur_singer, singer ) == 0 && strcmp( cur_name, name ) == 0 ){
        is_existed = true;
        break;
    }
}
if( is_existed ){// 如果已经存在
    printf("该歌手的该歌曲已经存在曲库之中, 无法重复添加\n");
    fclose( songs_list );
    return;
}
fclose( songs_list );
songs_list = fopen("/home/yyang/MyProgram/c_lab/src/ktv/songs", "a");
    // 以附加写入方式重新打开文件, 准备添加歌曲
if( !songs_list ){
    printf( "打开歌单出错, 目前无法进行添加操作\n");
    return;
}
char directory[1024]="/home/yyang/MyProgram/c_lab/src/ktv/";
size_t pre_len = strlen( directory );
strcpy( directory+pre_len, singer );
*(directory+pre_len+strlen(singer) ) = '_';
strcpy( directory+pre_len+strlen(singer)+1, name );
    // 新添加的歌曲的歌词文件所存放的路径名
//printf("%s\n", directory);
FILE *song = fopen(directory, "w+");
if( !song ){
    printf( "创建歌曲出错, 目前无法进行添加操作\n");
    fclose( songs_list );
    return;
}
// 打开新歌曲的歌词文件
printf( "请输入这首歌的歌词, 歌词以由仅含有一个'\n'的新一行为结束标志\n");
char txt[512];
while( scanf("%s",txt) && txt[0] != '#' ){
    fprintf(song, "%s\n", txt );
}

```

```

    fprintf(songs_list, "%s %s\n", name, singer );
    printf( "歌曲添加成功\n" );

    fclose(songs_list);
    fclose( song );
    return;
}

/*
 * 函数名称: admin_delete_song
 * 函数功能: 为 ktv 删除歌曲
 * 参数: 无
 * 返回值: void
 */
void admin_delete_song(){
    FILE *songs_list = fopen("/home/yyang/MyProgram/c_lab/src/ktv/songs", "r");
    // 未执行删除时的歌曲列表
    FILE *new_song_list = fopen("/home/yyang/MyProgram/c_lab/src/ktv/songs.temp", "w");
    // 执行删除后的歌曲列表
    if( !songs_list || !new_song_list ){
        printf( "打开歌单出错， 目前无法进行删除操作\n");
        return;
    }
    printf("\n=====删除歌曲=====\n");
    char name[512];// 输入的歌曲名
    char singer[512];// 输入的作者名
    char cur_name[512];// 现在遍历到的歌曲名
    char cur_singer[512];// 现在遍历到的作者名
    bool is_existed = false;// 表示该作者的该歌曲已经在系统中
    printf("请输入你要删除的歌曲的名称\n");
    scanf( "%s", name );
    printf("请输入该歌曲的歌手名称\n");
    scanf( "%s", singer );
    while( fscanf(songs_list, "%s", cur_name) != EOF && fscanf(songs_list, "%s", cur_singer) ){
        // 与现存歌曲列表中的结果逐一比对
        // 若找到对应的歌曲， 则不写入新的歌曲列表
        // 否则,写入新的歌曲列表
        if( strcmp( cur_singer, singer ) == 0 && strcmp( cur_name, name ) == 0 ){
            continue;
        }
        fprintf(new_song_list, "%s %s\n", cur_name, cur_singer );
    }
    char directory[1024]="/home/yyang/MyProgram/c_lab/src/ktv/";
    size_t pre_len = strlen( directory );

```

```

strcpy( directory+pre_len, singer );
*(directory+pre_len+strlen(singer) ) = '_';
strcpy( directory+pre_len+strlen(singer)+1, name );
    // 删除的歌曲的歌词文件所存放的路径名
fclose( songs_list );
fclose( new_song_list );
remove( "/home/yyang/MyProgram/c_lab/src/ktv/songs" );// 移除旧歌曲列表
remove( directory );// 移除要删除的歌曲文件的歌词文件
rename(
                                "/home/yyang/MyProgram/c_lab/src/ktv/songs.temp",
"/home/yyang/MyProgram/c_lab/src/ktv/songs" );
    // 重命名新的歌曲列表的文件名
printf("歌曲删除成功\n");
}

/*
* 函数名称: admin
* 函数功能: 打开管理员模式
* 参数: 无
* 返回值: void
*/
void admin( ){
    int choice;// 表示用户所选操作
    while(1){
        printf("\n=====管理员模式=====\n");
        printf("*****功能列表*****\n");
        printf("1.添加歌曲\n");
        printf("2.删除歌曲\n");
        printf("3.退出系统\n");
        printf("请输入对应序号选择功能，按回车键确认选择\n");
        printf("*****\n");
        printf("=====\n");
        scanf( "%d", &choice );
        switch( choice ){
            case 1: admin_add_song(); break;
            case 2: admin_delete_song(); break;
            case 3: printf("管理员操作结束\n"); return; break;
            default: printf("输入选项不存在，请核对后重新选择\n"); break;
        }
    }
}

/* 用户操作的相关模块 */

/*

```

```

* 函数名称: user_find_song_singer
* 函数功能: 查找某一歌手的所演唱的歌曲
* 参数: 无
* 返回值: void
*/
void user_find_song_singer(){
    FILE *songs_list = fopen("/home/yyang/MyProgram/c_lab/src/ktv/songs", "r");
    if( !songs_list ){
        printf( "打开歌单出错, 目前无法进行查询操作\n");
        return;
    }
    printf("\n=====查找歌曲=====\n");
    char singer[512];// 输入的作者名
    char cur_name[512];// 现在遍历到的歌曲名
    char cur_singer[512];// 现在遍历到的作者名
    printf("请输入你要查询的歌手的名称\n");
    scanf( "%s", singer );
    printf("*****查询结果*****\n");
    while( fscanf(songs_list, "%s", cur_name) != EOF && fscanf(songs_list, "%s", cur_singer ) ){
        // 与现存歌曲列表中的结果逐一比对
        // 若找到对应歌手的歌曲, 则予以显示
        // 否则, 不显示
        if( strcmp( cur_singer, singer ) == 0 ){
            printf("%s - %s\n", cur_name, cur_singer );
        }
    }
    printf("查询结束\n");
    fclose( songs_list );
}

/*
* 函数名称: user_find_song_name
* 函数功能: 查找同名的所有歌曲
* 参数: 无
* 返回值: void
*/
void user_find_song_name(){
    FILE *songs_list = fopen("/home/yyang/MyProgram/c_lab/src/ktv/songs", "r");
    if( !songs_list ){
        printf( "打开歌单出错, 目前无法进行查询操作\n");
        return;
    }
    printf("\n=====查找歌曲=====\n");
    char name[512];// 输入的歌曲

```

```

char cur_name[512];// 现在遍历到的歌曲名
char cur_singer[512];// 现在遍历到的作者名
printf("请输入你要查询的歌曲的名称\n");
scanf( "%s", name );
printf("*****查询结果*****\n");
while( fscanf(songs_list, "%s", cur_name) != EOF && fscanf(songs_list, "%s", cur_singer )){
    // 与现存歌曲列表中的结果逐一比对
    // 若找到对应歌名的歌曲, 则予以显示
    // 否则, 不显示
    if( strcmp( cur_name, name ) == 0 ){
        printf("%s - %s\n", cur_name, cur_singer );
    }
}
printf("查询结束\n");
fclose( songs_list );
}

/*
* 函数名称: user_find_song
* 函数功能: 用户查找歌曲选项的二级界面
* 参数: 无
* 返回值: void
*/
void user_find_song(){
    int choice;
    while(1){
        printf("\n=====查找歌曲=====\n");
        printf("*****功能列表*****\n");
        printf("1.按歌名查找\n");
        printf("2.按歌手查找\n");
        printf("3.返回上级\n");
        printf("请输入对应序号选择功能, 按回车键确认选择\n");
        printf("*****\n");
        printf("=====");
        scanf( "%d", &choice );
        switch( choice ){
            case 1: user_find_song_name(); break;
            case 2: user_find_song_singer(); break;
            case 3: printf("正在返回上一级\n"); return; break;
            default: printf("输入选项不存在, 请核对后重新选择\n"); break;
        }
    }
}

```



```

/*
* 函数名称: user_play_song
* 函数功能: 播放歌曲
* 参数: 无
* 返回值: void
*/
void user_play_song(){
    FILE *songs_list = fopen("/home/yyang/MyProgram/c_lab/src/ktv/songs", "r");
    if( !songs_list ){
        printf( "打开歌单出错， 目前无法进行点播操作\n");
        return;
    }
    printf("\n=====点播歌曲=====\n");
    char name[512];// 输入的歌曲名
    char singer[512];// 输入的作者名
    char cur_name[512];// 现在遍历到的歌曲名
    char cur_singer[512];// 现在遍历到的作者名
    printf("请输入你要点播的歌曲的名称\n");
    scanf( "%s", name );
    printf("请输入该歌曲的歌手名称\n");
    scanf( "%s", singer );
    bool is_existed = false;// 是否存在对应的歌曲
    while( fscanf(songs_list, "%s", cur_name) != EOF && fscanf(songs_list, "%s", cur_singer) ){
        // 与现存歌曲列表中的结果逐一比对
        // 若找到对应歌手的歌曲，则设置存在标志
        if( strcmp( cur_singer, singer ) == 0 && strcmp( cur_name, name ) == 0 ){
            is_existed = true;
            break;
        }
    }
    if( !is_existed ){
        printf("所点歌曲不在曲库中， 点歌失败\n");
        fclose(songs_list);
        return;
    }
    char directory[1024]="/home/yyang/MyProgram/c_lab/src/ktv/";
    size_t pre_len = strlen( directory );
    strcpy( directory+pre_len, singer );
    *(directory+pre_len+strlen(singer) ) = '_';
    strcpy( directory+pre_len+strlen(singer)+1, name );
    // 要播放的歌曲的歌词文件所存放的路径名
    FILE *song = fopen(directory, "r");
    if( !song ){
        printf( "无法打开该歌曲歌词， 点歌失败\n");
    }
}

```

```

        fclose( songs_list );
        return;
    }
    char buf[512];
    while( fgets(buf, 512,song ) != NULL ){
        printf( "%s", buf );
    }// 逐行显示歌词
    fclose( songs_list );
    fclose( song );
    printf("点歌成功\n");
}

/*
* 函数名称: user
* 函数功能: 打开用户模式
* 参数: 无
* 返回值: void
*/
void user(){
    int choice;
    while(1){
        printf("\n=====用户模式=====\\n");
        printf("*****功能列表*****\\n");
        printf("1.查找歌曲\\n");
        printf("2.点播歌曲\\n");
        printf("3.退出系统\\n");
        printf("请输入对应序号选择功能，按回车键确认选择\\n");
        printf("*****\\n");
        printf("=====\\n");
        scanf( "%d", &choice );
        switch( choice ){
            case 1: user_find_song(); break;
            case 2: user_play_song(); break;
            case 3: printf("用户操作结束\\n"); return; break;
            default: printf("输入选项不存在，请核对后重新选择\\n"); break;
        }
    }
}

/* 开始模块 */

/*
* 函数名称: ktv_start
* 函数功能: 开始使用 ktv

```

```

* 参数: char user_level( 操作者的身份 ), const char *name( 操作者的姓名 )
* 返回值: void
*/
void ktv_start( char user_level, const char *name ){
    if( user_level == 'a' ){// 管理员模式
        FILE *admin_list = fopen( "/home/yyang/MyProgram/c_lab/src/ktv/admin", "r" );
        if( !admin_list ){
            printf("打开管理员列表失败, 无法进行验证, 程序结束运行。\\n");
            return;
        }// 打开管理员名称列表, 准备进行验证
        char buf[512];// 临时缓冲区
        while( fscanf(admin_list,"%s", buf) != EOF ){
            // 与管理列表中的名称逐一比对
            // 若发现该名管理员, 则开启管理员界面, 操作结束后退出系统
            if( strcmp( buf, name ) == 0 ){
                printf("验证通过, 正在进入系统\\n");
                admin();// 打开管理员操作界面
                printf("已退出系统, 欢迎下次使用。\\n");
                fclose(admin_list);
                return;
            }
        }
        printf("该管理员未登记入系统中, 请核对名称后重试。\\n");
        fclose(admin_list);
    }
    else if( user_level == 'u' ){// 用户模式
        FILE *user_list = fopen( "/home/yyang/MyProgram/c_lab/src/ktv/user", "r" );
        if( !user_list ){
            printf("打开用户列表失败, 无法进行验证, 程序结束运行。\\n");
            return;
        }// 打开用户名称列表, 准备进行验证
        char buf[512];
        while( fscanf(user_list,"%s", buf) != EOF ){
            // 与用户列表中的名称逐一比对
            // 若发现该名用户, 则开启用户界面, 操作结束后退出系统
            bool flag = strcmp( buf, name ) == 0;
            if( flag ){
                printf("验证通过, 正在进入系统\\n");
                user();// 打开用户操作界面
                printf("已退出系统, 欢迎下次使用。\\n");
                fclose(user_list);
                return;
            }
        }
    }
}

```

```

        fclose(user_list);
        printf("该用户未登记入系统中，请核对名称后重试。\\n");
    }
    else{// 若非以上两者，说明存在错误
        printf("用户类型错误，请核对输入后重新启动程序。\\n");
    }
    return;
}

#endif

```

6. 测试数据（输入、输出）:

```

PS F:\MyProgram\c_lab\exe> .\ktv_test.exe -u yyang
验证通过，正在进入系统

=====用户模式=====
*****功能列表*****
1.查找歌曲
2.点播歌曲
3.退出系统
请输入对应序号选择功能，按回车键确认选择
*****

PS F:\MyProgram\c_lab\exe> .\ktv_test.exe -a yyang
验证通过，正在进入系统

=====管理员模式=====
*****功能列表*****
1.添加歌曲
2.删除歌曲
3.退出系统
请输入对应序号选择功能，按回车键确认选择
*****

=====
1

=====添加歌曲=====
newsong
请输入该歌曲的歌手名称
me
请输入这首歌的歌词，歌词以由仅含有一个'#'的新一行为结束标志
hjkfdashflk
hasjklfhas#
#
歌曲添加成功

=====管理员模式=====
*****功能列表*****
1.添加歌曲
2.删除歌曲
3.退出系统
请输入对应序号选择功能，按回车键确认选择
*****

=====
3
管理员操作结束
已退出系统，欢迎下次使用。

```

```

已退出系统，欢迎下次使用。
PS F:\MyProgram\c_lab\exe> .\ktv_test.exe -u yyang
验证通过，正在进入系统

=====用户模式=====
*****功能列表*****
1.查找歌曲
2.点播歌曲
3.退出系统
请输入对应序号选择功能，按回车键确认选择
*****

=====
2

=====点播歌曲=====
请输入你要点播的歌曲的名称
newsong
请输入该歌曲的歌手名称
me
hjkfdashflk
hasjklfhas#
点歌成功

=====用户模式=====
*****功能列表*****
1.查找歌曲
2.点播歌曲
3.退出系统
请输入对应序号选择功能，按回车键确认选择
*****

=====
3
用户操作结束
已退出系统，欢迎下次使用。
PS F:\MyProgram\c_lab\exe> █

```

7. 小结:

本次实验圆满完成，实验结果符合实验要求。

实验过程中，我复习到了c语言中文件读写、内存管理、结构体定义、基本数据结构、高级格式化输入输出。