

学生成绩管理系统实验报告

专业班级：计算机 212

学号：2109010215

姓名：杜思维

程序源码

main.c:

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

#include "log.h"
#include "menu.h"
#include "LinkList.h"

#define CSV_NAME "export.csv"

LinkList *pList = NULL;

/**
 * @brief 读入数据 (回调)
 *
 * @return DataType 读入的数据
 */
DataType input_data(jmp_buf *jb)
{
    DataType x;

    printf("请输入学号: ");
    scanf("%d", &x.sno);

    printf("请输入姓名: ");
    scanf("%s", &x.name);

    printf("请输入成绩: ");
    scanf("%d", &x.score);

    getchar();

    return x;
}

/**
 * @brief 输出 JSON 回调
 *
 * @param x 遍历项
 */
void print_json(LinkList *pNode)
{
    puts("    {");
    printf("        \"sno\": %d,\n", pNode->data.sno);
```

```
printf("      \\"name\\": \\"%s\\",\\n", pNode->data.name);
printf("      \\"score\\": %d\\n", pNode->data.score);
printf("    }");
}

/**
 * @brief 菜单回调 创建
 *
 */
void menu_create()
{
    pList = createTailList(input_data, true);
}

/**
 * @brief 链表判空 (初始化)
 *
 * @retval true 不为空 (已初始化)
 * @retval false 为空 (未初始化)
 */
bool check_not_null()
{
    if (pList && pList->next)
        return true;
    log_err("数据为空, 请先创建!");
    return false;
}

/**
 * @brief 菜单回调 输出 JSON
 *
 */
void menu_json()
{
    if (check_not_null())
        printList(pList, print_json, true);
}

/**
 * @brief 遍历回调 输出表格
 *
 * @param pNode 节点指针
 */
void print_table(LinkList *pNode)
{
    printf("%-10d | %-6s | %d\\n",
        pNode->data.sno,
        pNode->data.name,
        pNode->data.score);
}

/**
```

```
* @brief 菜单回调 输出
*
*/
void menu_print()
{
    if (check_not_null())
    {
        printf("%-10s | %-6s | %s\n", "学号", "姓名", "成绩");
        print_split(26);
        forEach(pList, print_table, false);
    }
}

/**
 * @brief 读入索引并查找节点
 *
 * @return LinkList* 找到的节点
 */
LinkList *input_get_node()
{
    printf("请输入节点索引: ");
    int i = 0;
    scanf("%d", &i);
    getchar();
    return getNode(pList, i - 1);
}

/**
 * @brief 菜单回调 删除
 *
 */
void menu_delete()
{
    if (check_not_null())
        deleteAfterNode(input_get_node());
}

/**
 * @brief 菜单回调 插入
 *
 */
void menu_insert()
{
    if (check_not_null())
        insertAfterNode(input_get_node(), input_data(NULL));
}

/**
 * @brief 要查找的成绩
 *
 */
int score = 0;
```

```
/**
 * @brief 按成绩查找节点回调
 *
 * @param n 遍历节点
 * @retval true 目标匹配
 * @retval false 目标不匹配
 */
bool find_by_score(LinkList *pNode)
{
    if (pNode->next)
        return pNode->next->data.score == score;
    return false;
}

/**
 * @brief 菜单回调 删除指定成绩
 *
 */
void menu_delete_score()
{
    if (check_not_null())
    {
        printf("请输入要查找的成绩: ");
        scanf("%d", &score);
        deleteAfterNode(findNode(pList, find_by_score, true));
    }
}

/**
 * @brief 文件名
 *
 */
#define FILE_NAME "data.dat"

/**
 * @brief 文件指针
 *
 */
FILE *pFile;

/**
 * @brief 遍历回调 保存文件
 *
 * @param pNode 节点指针
 */
void save_file(LinkList *pNode)
{
    fwrite(&pNode->data, sizeof(DataType), 1, pFile);
}
```

```
/**
 * @brief 遍历回调 加载文件
 *
 * @param jb 跨函数跳转的缓冲区
 * @return DataType 读取的节点数据
 */
DataType load_file(jmp_buf *jb)
{
    DataType x;
    if (!fread(&x, sizeof(DataType), 1, pFile))
    {
        longjmp(*jb, true);
    }
    else
        return x;
}

/**
 * @brief 菜单回调 保存文件
 *
 */
void menu_save()
{
    if (check_not_null())
    {
        if (!(pFile = fopen(FILE_NAME, "wb")))
            log_err("无法打开文件 \"%s\"! 文件被占用?", FILE_NAME);

        foreach(pList, save_file, false);

        fclose(pFile);

        log_info("操作完成!");
    }
}

/**
 * @brief 菜单回调 加载文件
 *
 */
void menu_load()
{
    if (!(pFile = fopen(FILE_NAME, "rb")))
        log_err("无法打开文件 \"%s\"! 文件不存在?", FILE_NAME);

    pList = createTailList(load_file, false);

    fclose(pFile);

    log_info("操作完成!");
}
```

```
/**
 * @brief 遍历回调 保存 CSV
 *
 * @param pNode 节点指针
 */
void save_csv(LinkList *pNode)
{
    fprintf(pFile, "%d,%s,%d\n",
            pNode->data.sno,
            pNode->data.name,
            pNode->data.score);
}

/**
 * @brief 菜单回调 保存 CSV
 *
 */
void menu_csv()
{
    if (check_not_null())
    {
        if (!(pFile = fopen(CSV_NAME, "w")))
            log_err("无法打开文件 \"%s\"! 文件被占用?", CSV_NAME);

        fputs("学号,姓名,成绩\n", pFile);
        forEach(pList, save_csv, false);

        fclose(pFile);

        log_info("操作完成!");
    }
}

/**
 * @brief 主函数
 *
 * @return int
 */
int main()
{
    menus menu, *pMenu = &menu;
    init_menu(pMenu, "学生成绩管理系统", 9);

    add_menu(pMenu, "创建数据", menu_create);
    add_menu(pMenu, "输出 JSON", menu_json);
    add_menu(pMenu, "插入节点", menu_insert);
    add_menu(pMenu, "删除节点", menu_delete);
    add_menu(pMenu, "删除指定成绩的节点", menu_delete_score);
    add_menu(pMenu, "保存到文件", menu_save);
}
```

```
    add_menu(pMenu, "从文件加载", menu_load);
    add_menu(pMenu, "保存到 CSV", menu_csv);
    add_menu(pMenu, "遍历输出", menu_print);

    menu_main(pMenu);

    return 0;
}
```

menu.c:

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

#include "log.h"
#include "menu.h"

/**
 * @brief 获取菜单项描述最大长度
 *
 * @param m 菜单结构体指针
 * @return int 长度
 */
int max_item_len(menus *m)
{
    int len = 0, tmp = 0;

    for (char i = 1; i <= m->size; i++)
        if ((tmp = strlen(m->items[i - 1].desc)) > len)
            len = tmp;

    return len + 3;
}

/**
 * @brief 输出分割符
 *
 * @param len 长度
 */
void print_split(int len)
{
    for (int i = 0; i < len; i++)
        putchar('-');
    putchar('\n');
}

/**
 * @brief 输出菜单项
 *
 * @param m 菜单项指针
 */
```

```
*/
void print_menu(menus *m)
{
    int len = max_item_len(m);

    puts(m->title);

    print_split(len);

    for (char i = 1; i <= m->size; i++)
        printf("%d. %s\n", i, m->items[i - 1].desc);
    puts("0. 退出");

    print_split(len);

    printf("请输入选项: ");
}

/**
 * @brief 初始化菜单结构体
 *
 * @param m 菜单结构体指针
 * @param title 菜单标题
 * @param size 菜单大小
 */
void init_menu(menus *m, char *title, int size)
{
    m->title = title;
    m->size = size;
    m->items = (menu_item *)malloc(sizeof(menu_item) * size);
}

/**
 * @brief 菜单主函数
 *
 * @param m
 */
void menu_main(menus *m)
{
    short c = 0;
    do
    {
        print_menu(m);
        scanf("%hd", &c);
        getchar();
        putchar('\n');

        if (c > 0 && c <= m->size)
        {
            m->items[c - 1].fun();
            putchar('\n');
        }
    }
}
```



```
        system("pause");
        system("cls");
    } while (c);
}

/**
 * @brief 添加菜单项
 *
 * @param menu 菜单指针
 * @param desc 菜单描述
 * @param fun 菜单函数
 */
void add_menu(menu *menu, char *desc, menu_fun fun)
{
    static char i = 0;
    if (i >= menu->size)
    {
        log_err("菜单列表溢出!");
        return;
    }

    menu->items[i++] = (menu_item){
        .desc = desc,
        .fun = fun,
    };
}
```

menu.h:

```
#ifndef _MENU_
#define _MENU_

/**
 * @brief 菜单回调
 *
 */
typedef void (*menu_fun)();

/**
 * @brief 菜单项
 *
 */
typedef struct menu_item
{
    // 描述
    char *desc;
    // 函数
    menu_fun fun;
} menu_item;

/**
```

```
* @brief 菜单结构体
*
*/
typedef struct menus
{
    // 标题
    char *title;
    // 大小
    char size;
    // 菜单项
    menu_item *items;
} menus;

/**
 * @brief 菜单主函数
 *
 * @param m
 */
void menu_main(menus *m);

/**
 * @brief 初始化菜单结构体
 *
 * @param m 菜单结构体指针
 * @param title 菜单标题
 * @param size 菜单大小
 */
void init_menu(menus *m, char *title, int size);

/**
 * @brief 新建菜单项
 *
 * @param m 菜单指针
 * @param desc 描述
 * @param fun 函数指针
 * @return menu_item 菜单项
 */
void add_menu(menus *menu, char *desc, menu_fun fun);

/**
 * @brief 输出分割符
 *
 * @param len 长度
 */
void print_split(int len);
#endif // _MENU_
```

log.c:

```
#include <stdio.h>

#include "log.h"

/**
 * @brief 日志函数
 *
 * @param tag 日志标签 (级别)
 * @param fn_name 函数名
 * @param fmt 待格式化的字符串
 * @param ... 可变参数
 */
void log_log(char *tag, const char *fn_name, char *fmt, ...)
{
    va_list ap;

    fprintf(stderr, "[%s] %s(): ", tag, fn_name);

    va_start(ap, fmt);
    vfprintf(stderr, fmt, ap);
    va_end(ap);

    fputc('\n', stderr);
}
```

log.h:

```
#ifndef _LOG_
#define _LOG_

#include <stdarg.h>

/**
 * @brief 日志函数
 *
 * @param tag 日志标签 (级别)
 * @param fn_name 函数名
 * @param fmt 待格式化的字符串
 * @param ... 可变参数
 */
void log_log(char *tag, const char *fn_name, char *fmt, ...);

/**
 * @brief 指定 tag 日志
 *
 * @param tag 日志标签 (级别)
 * @param ... 可变参数
 */
#define log_tag(tag, ...) log_log(tag, __FUNCTION__, __VA_ARGS__)
```

```
/**
 * @brief 错误日志
 *
 * @param ... 可变参数
 */
#define log_err(...) log_tag("ERROR", __VA_ARGS__)

/**
 * @brief 警告日志
 *
 * @param ... 可变参数
 */
#define log_warn(...) log_tag("WARN", __VA_ARGS__)

/**
 * @brief 信息日志
 *
 * @param ... 可变参数
 */
#define log_info(...) log_tag("INFO", __VA_ARGS__)

/**
 * @brief 调试日志
 *
 * @param ... 可变参数
 */
#define log_dbg(...) log_tag("DEBUG", __VA_ARGS__)

#endif // _LOG_
```

LinkedList.c:

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

#include "log.h"
#include "LinkedList.h"

/**
 * @brief 初始化节点
 *
 * @param x 数据
 * @param next 下一个节点的指针
 * @return LinkedList* 生成的节点
 */
LinkedList *initNode(DataType x, LinkedList *next)
{
    LinkedList *L = (LinkedList *)malloc(sizeof(LinkedList));
    L->data = x;
    L->next = next;
```

```
    return L;
}

/**
 * @brief 尾插法建立带头节点链表
 *
 * @param callback 创建回调
 *
 * @return LinkList* 创建后的头节点
 */
LinkList *createTailList(create_callback callback, bool has_hint)
{
    DataType x;
    bool is_continue = true;
    LinkList *headNode = initNode(x, NULL), *L = headNode;
    jmp_buf jb;

    while (is_continue)
    {
        if (setjmp(jb))
            break;
        else
            L = L->next = initNode(callback(&jb), NULL);

        if (has_hint)
        {
            printf("\n 按回车继续, 输入 0 退出: ");
            if (is_continue = getchar() - '0')
                putchar('\n');
        }
    }

    return headNode;
}

/**
 * @brief 输出链表
 *
 * @param L 链表指针
 * @param callback 元素回调
 * @param new_line 是否换行
 */
void printList(LinkList *L, forEach_callback callback, bool new_line)
{
    putchar('[');

    while (L = L->next)
    {
        if (new_line)
            putchar('\n');

        callback(L);
    }
}
```

```
        putchar(',');
    }

    printf("\b ");
    if (new_line)
        putchar('\n');
    puts("]");
}

/**
 * @brief 在节点后插入
 *
 * @param n 节点
 * @param x 数据
 */
void insertAfterNode(LinkList *n, DataType x)
{
    if (!n)
    {
        log_err("节点为 NULL, 插入失败!");
        return;
    }
    n->next = initNode(x, n->next);
}

/**
 * @brief 删除节点
 *
 * @param n 前一个节点
 */
void deleteAfterNode(LinkList *n)
{
    if (!n)
    {
        log_err("节点为 NULL, 删除失败!");
        return;
    }
    LinkList *tmp = n->next;

    if (tmp)
    {
        n->next = tmp->next;
        free(tmp);
    }
}

/**
 * @brief 按索引获取节点
 *
 * @param L 链表指针
 * @param i 索引
 */
```

```
* @return LinkedList* 获得的节点
*/
LinkedList *getNode(LinkedList *L, int i)
{
    if (i >= 0)
        for (int tmp = 0; L; tmp++)
        {
            if (tmp >= i)
                return L;
            L = L->next;
        }

    log_err("索引不合法!");
    return NULL;
}

/**
 * @brief 按条件查找节点
 *
 * @param L 链表指针
 * @param callback 条件回调
 * @param has_head 包括头节点
 * @return LinkedList* 找到的节点
 */
LinkedList *findNode(LinkedList *L, find_callback callback, bool has_head)
{
    if (!has_head)
        L = L->next;

    while (L)
    {
        if (callback(L))
            return L;

        L = L->next;
    }

    log_warn("未找到节点!");
    return NULL;
}

/**
 * @brief 遍历节点
 *
 * @param L 链表指针
 * @param callback 遍历回调
 * @param has_head 包括头节点
 */
void forEach(LinkedList *L, forEach_callback callback, bool has_head)
{
    if (!has_head)
```

```
        L = L->next;

    while (L)
    {
        callback(L);
        L = L->next;
    }
}
```

LinkedList.h:

```
#ifndef _LINK_LIST_
#define _LINK_LIST_

#include <stdbool.h>
#include <setjmp.h>

/**
 * @brief 数据类型
 *
 */
typedef struct Student
{
    // 学号
    int sno;
    // 姓名
    char name[7];
    // 成绩
    int score;
} DataType;

/**
 * @brief 节点结构
 *
 */
typedef struct Node
{
    // 数据域
    DataType data;
    // 指针域
    struct Node *next;
} LinkedList;

/**
 * @brief 初始化节点
 *
 * @param x 数据
 * @param next 下一个节点的指针
 * @return LinkedList* 生成的节点
 */
LinkedList *initNode(DataType x, LinkedList *next);
```



```
/**
 * @brief 在节点后插入
 *
 * @param n 节点
 * @param x 数据
 */
void insertAfterNode(LinkList *n, DataType x);

/**
 * @brief 删除节点
 *
 * @param n 前一个节点
 */
void deleteAfterNode(LinkList *n);

/**
 * @brief 遍历回调
 *
 * @param 遍历元素
 */
typedef void (*forEach_callback)(LinkList *L);

/**
 * @brief 创建回调
 *
 * @return 创建数据
 */
typedef DataType (*create_callback)(jmp_buf *jb);

/**
 * @brief 尾插法建立带头节点链表
 *
 * @param callback 创建回调
 *
 * @return LinkList* 创建后的头节点
 */
LinkList *createTailList(create_callback callback, bool has_hint);

/**
 * @brief 输出 JSON
 *
 * @param L 链表指针
 * @param callback 元素回调
 * @param new_line 是否换行
 */
void printList(LinkList *L, forEach_callback callback, bool new_line);

/**
```

```
* @brief 查找回调
*
* @param n 遍历节点
*
* @retval true 条件成立
* @retval false 条件不成立
*/
typedef bool (*find_callback)(LinkedList *n);

/**
 * @brief 按索引获取节点
 *
 * @param L 链表指针
 * @param i 索引
 * @return LinkedList* 获得的节点
 */
LinkedList *getNode(LinkedList *L, int i);

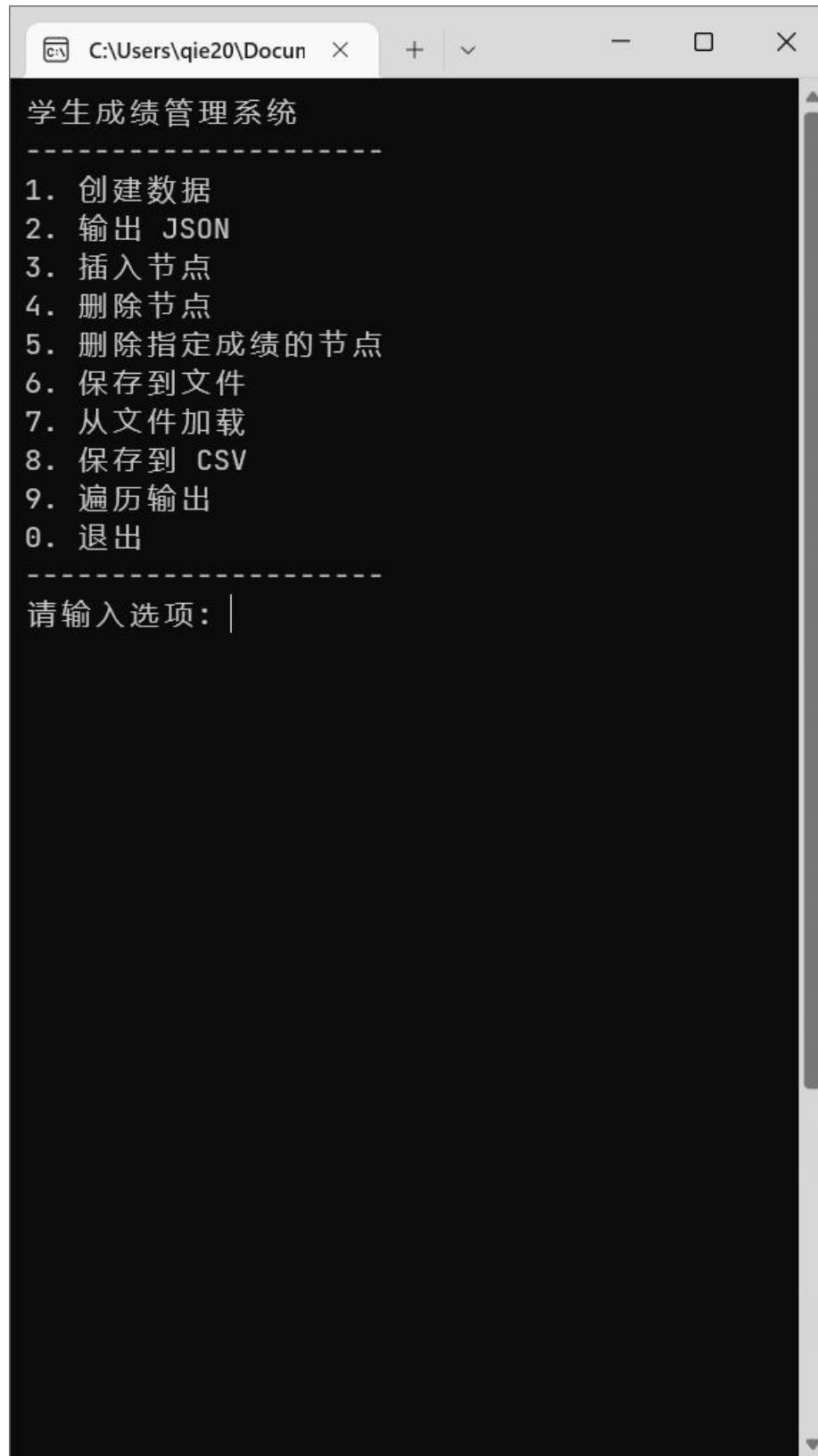
/**
 * @brief 按条件查找节点
 *
 * @param L 链表指针
 * @param callback 条件回调
 * @param has_head 包括头节点
 * @return LinkedList* 找到的节点
 */
LinkedList *findNode(LinkedList *L, find_callback callback, bool has_head);

/**
 * @brief 遍历节点
 *
 * @param L 链表指针
 * @param callback 遍历回调
 * @param has_head 包括头节点
 */
void forEach(LinkedList *L, forEach_callback callback, bool has_head);

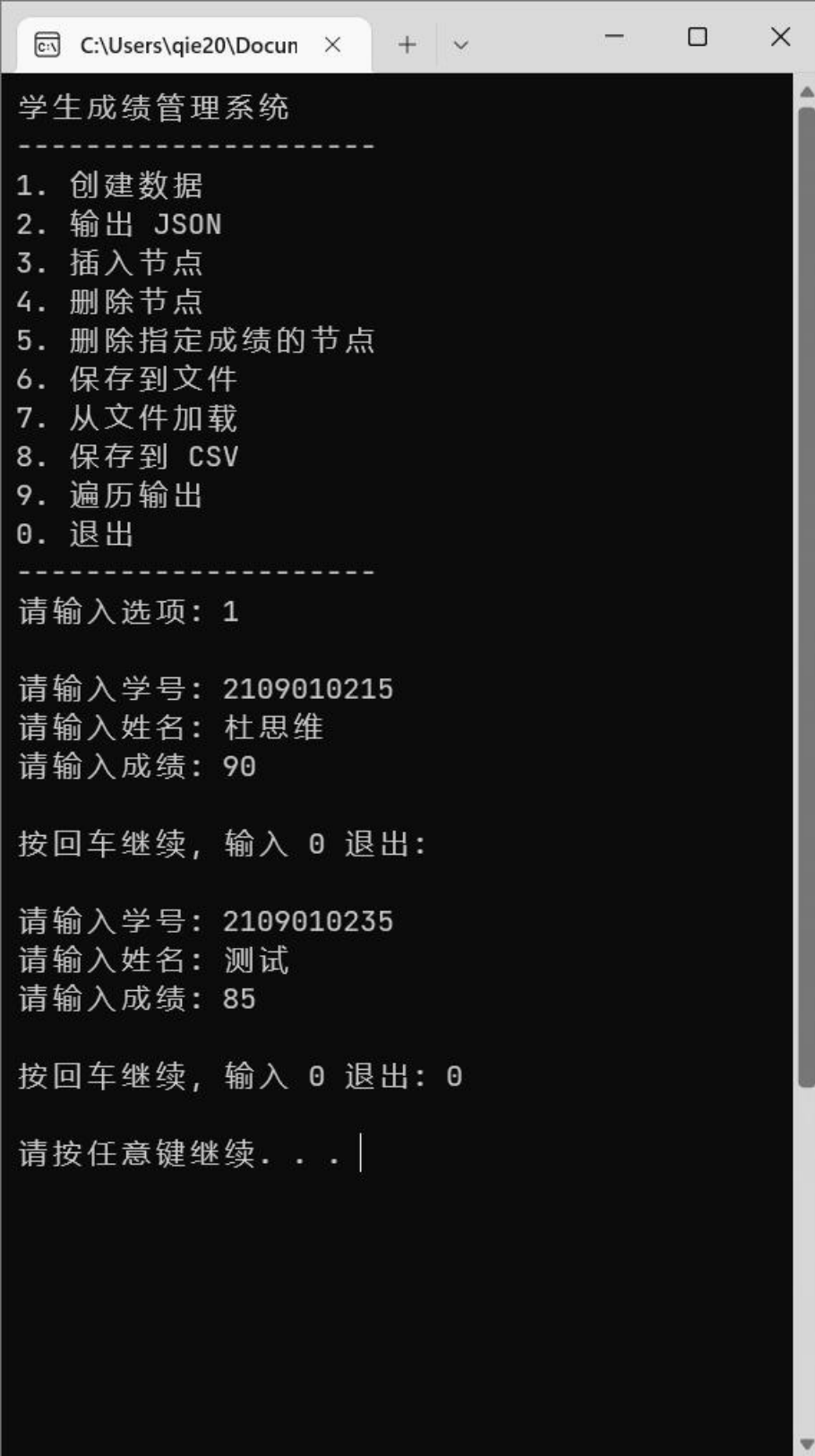
#endif // _LINK_LIST_
```

2. 运行结果截图

1. 主界面



2. 创建数据



```
C:\Users\qie20\Docun > 学生成绩管理系统
-----
1. 创建数据
2. 输出 JSON
3. 插入节点
4. 删除节点
5. 删除指定成绩的节点
6. 保存到文件
7. 从文件加载
8. 保存到 CSV
9. 遍历输出
0. 退出
-----
请输入选项: 1

请输入学号: 2109010215
请输入姓名: 杜思维
请输入成绩: 90

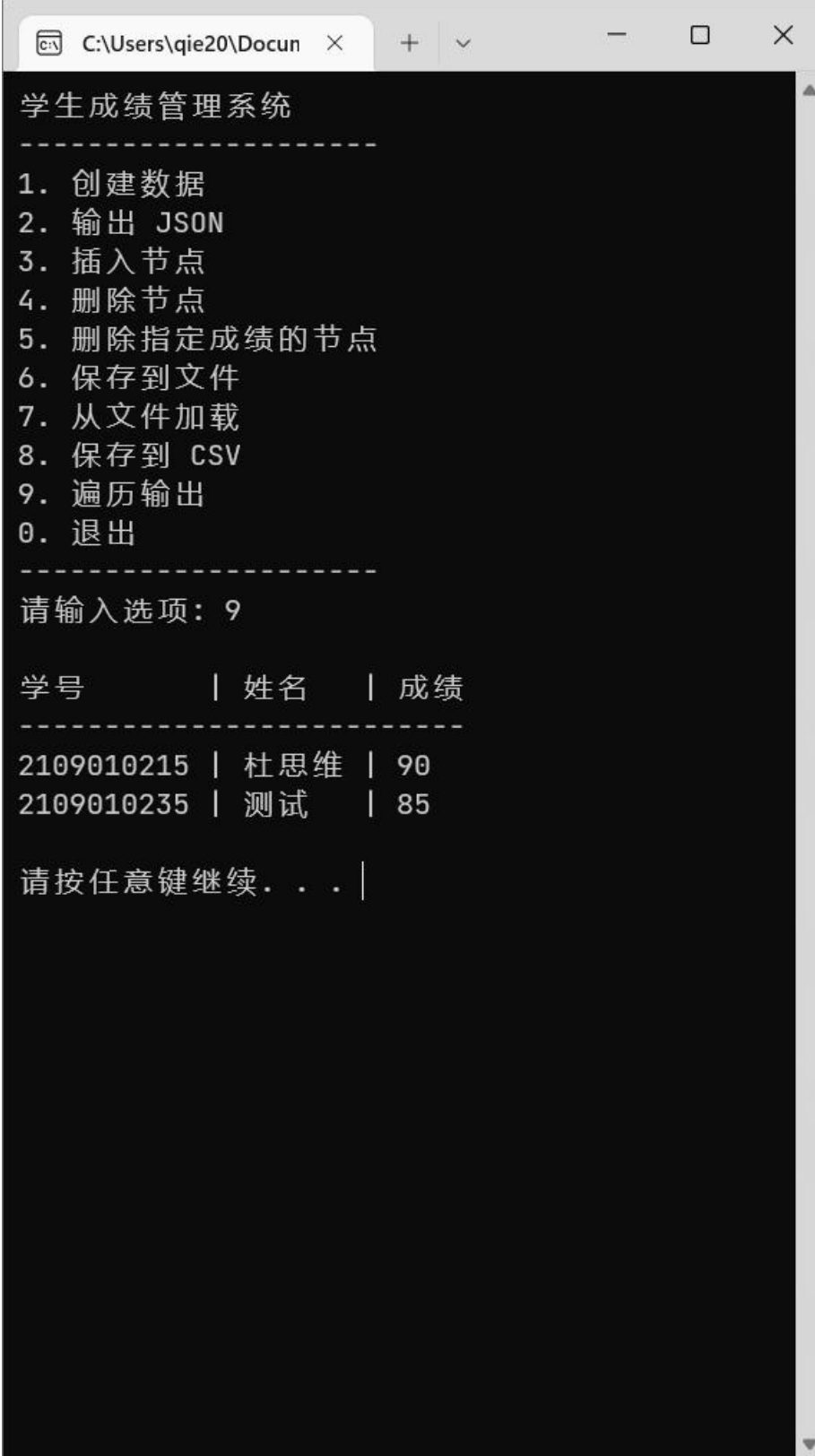
按回车继续, 输入 0 退出:

请输入学号: 2109010235
请输入姓名: 测试
请输入成绩: 85

按回车继续, 输入 0 退出: 0

请按任意键继续. . . |
```

3.遍历输出




```
C:\Users\qie20\Docun > 学生成绩管理系统
-----
1. 创建数据
2. 输出 JSON
3. 插入节点
4. 删除节点
5. 删除指定成绩的节点
6. 保存到文件
7. 从文件加载
8. 保存到 CSV
9. 遍历输出
0. 退出
-----
请输入选项：9

学号      | 姓名    | 成绩
-----
2109010215 | 杜思维  | 90
2109010235 | 测试    | 85

请按任意键继续. . .
```

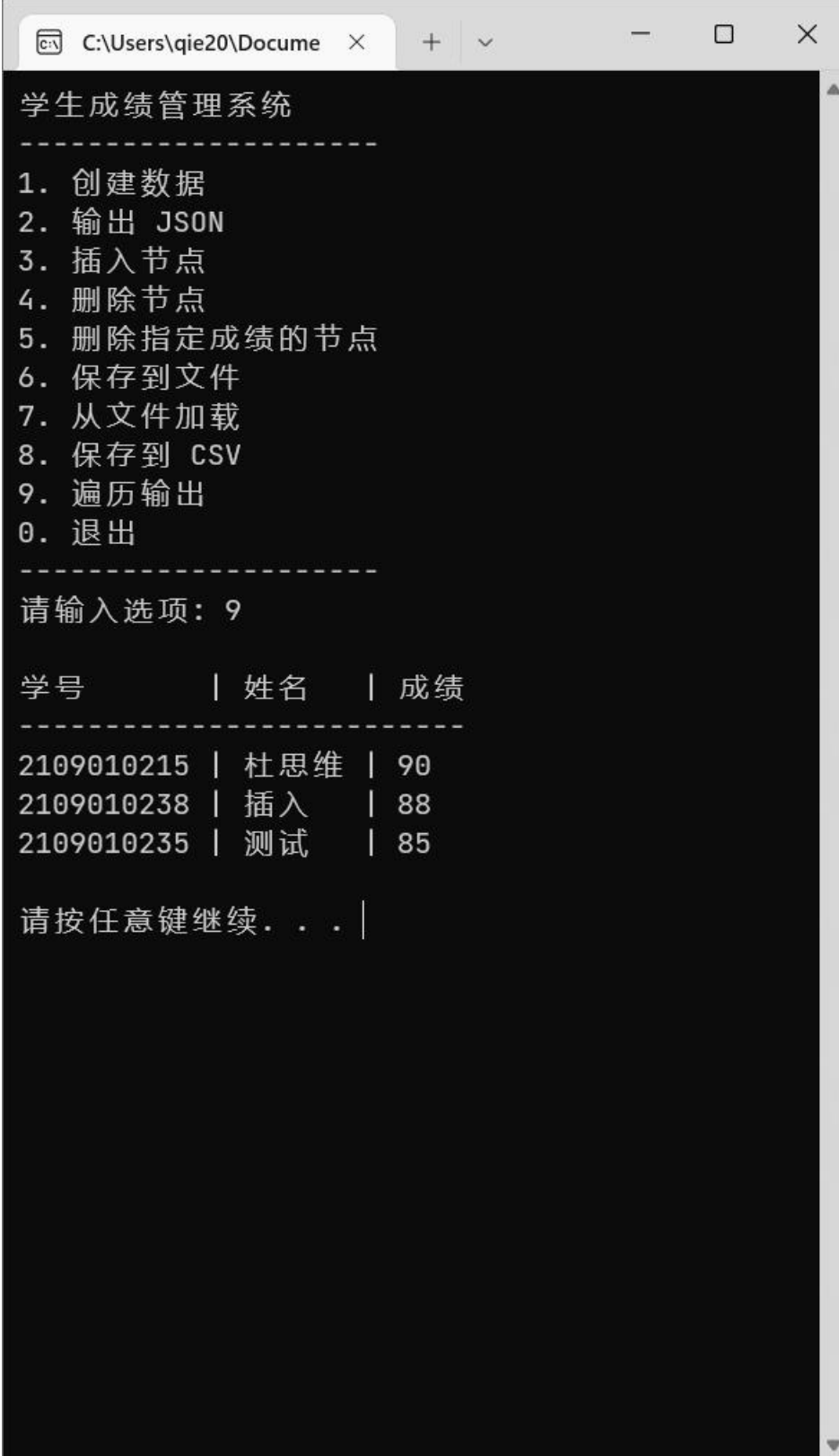
4.插入节点



```
C:\Users\qie20\Docume X + - □ X
学生成绩管理系统
-----
1. 创建数据
2. 输出 JSON
3. 插入节点
4. 删除节点
5. 删除指定成绩的节点
6. 保存到文件
7. 从文件加载
8. 保存到 CSV
9. 遍历输出
0. 退出
-----
请输入选项：3

请输入学号：2109010238
请输入姓名：插入
请输入成绩：88
请输入节点索引：2

请按任意键继续... |
```



```
C:\Users\qie20\Docume X + - □ X
学生成绩管理系统
-----
1. 创建数据
2. 输出 JSON
3. 插入节点
4. 删除节点
5. 删除指定成绩的节点
6. 保存到文件
7. 从文件加载
8. 保存到 CSV
9. 遍历输出
0. 退出
-----
请输入选项：9

学号      | 姓名    | 成绩
-----
2109010215 | 杜思维  | 90
2109010238 | 插入    | 88
2109010235 | 测试    | 85

请按任意键继续... |
```

5. 删除节点

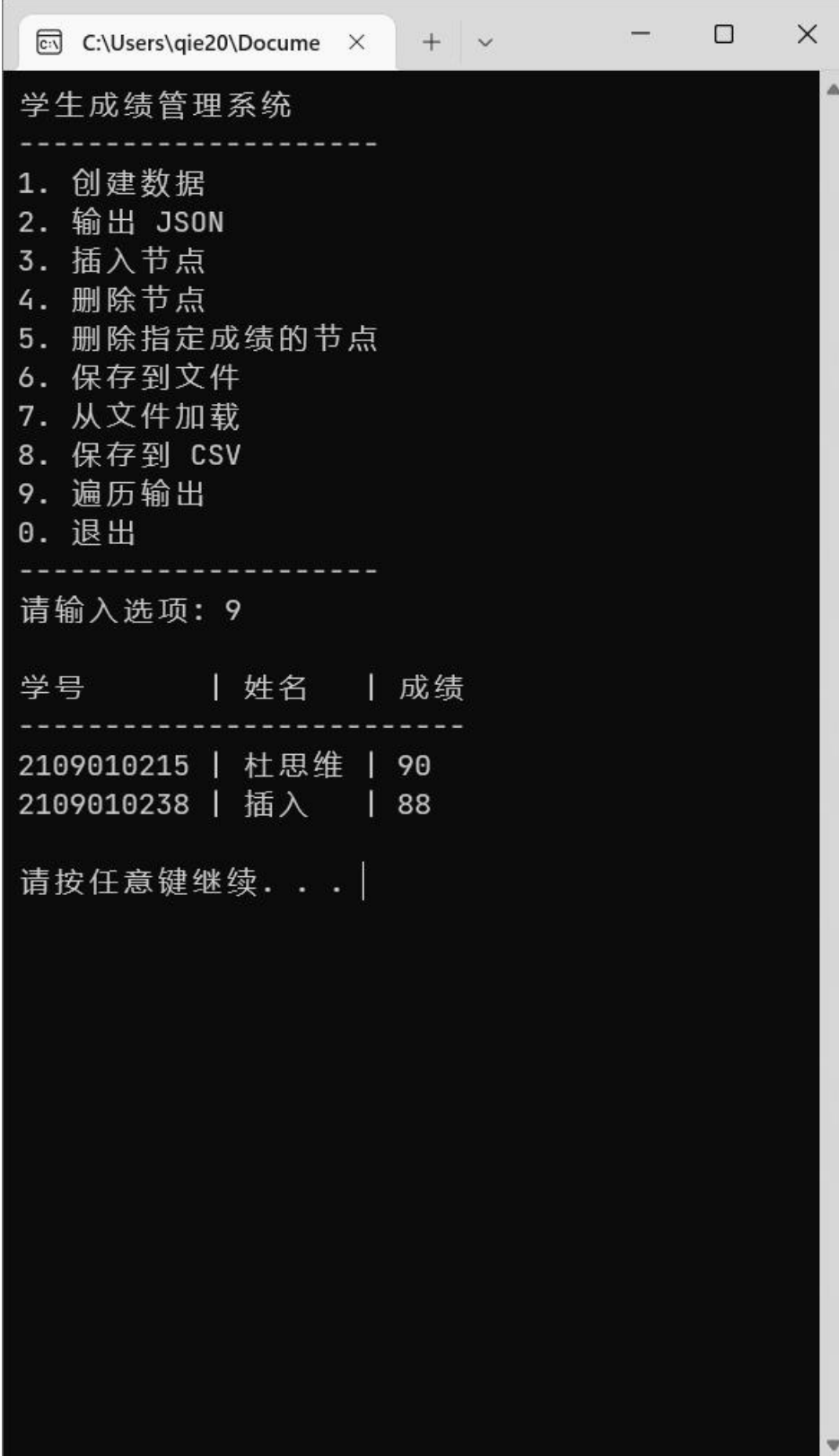


```
C:\Users\qie20\Docume  X  +  -  □  X

学生成绩管理系统
-----
1. 创建数据
2. 输出 JSON
3. 插入节点
4. 删除节点
5. 删除指定成绩的节点
6. 保存到文件
7. 从文件加载
8. 保存到 CSV
9. 遍历输出
0. 退出
-----
请输入选项：4

请输入节点索引：3

请按任意键继续. . . |
```

```
C:\Users\qie20\Docume X + - □ X
学生成绩管理系统
-----
1. 创建数据
2. 输出 JSON
3. 插入节点
4. 删除节点
5. 删除指定成绩的节点
6. 保存到文件
7. 从文件加载
8. 保存到 CSV
9. 遍历输出
0. 退出
-----
请输入选项：9

学号      | 姓名    | 成绩
-----
2109010215 | 杜思维  | 90
2109010238 | 插入    | 88

请按任意键继续... |
```

6.删除指定成绩的节点





```
C:\Users\qie20\Docume X + - □ X
学生成绩管理系统
-----
1. 创建数据
2. 输出 JSON
3. 插入节点
4. 删除节点
5. 删除指定成绩的节点
6. 保存到文件
7. 从文件加载
8. 保存到 CSV
9. 遍历输出
0. 退出
-----
请输入选项：9

学号      | 姓名    | 成绩
-----
2109010215 | 杜思维  | 90

请按任意键继续... |
```

7.文件的保存与加载



```
C:\Users\qie20\Docume X + - □ X
学生成绩管理系统
-----
1. 创建数据
2. 输出 JSON
3. 插入节点
4. 删除节点
5. 删除指定成绩的节点
6. 保存到文件
7. 从文件加载
8. 保存到 CSV
9. 遍历输出
0. 退出
-----
请输入选项：6

[INFO] menu_save(): 操作完成!

请按任意键继续. . . |
```

重新启动程序



```
C:\Users\qie20\Docume  X + - □ X
学生成绩管理系统
-----
1. 创建数据
2. 输出 JSON
3. 插入节点
4. 删除节点
5. 删除指定成绩的节点
6. 保存到文件
7. 从文件加载
8. 保存到 CSV
9. 遍历输出
0. 退出
-----
请输入选项：9

[ERROR] check_not_null(): 数据为空, 请先创建!

请按任意键继续. . . |
```

从文件加载



```
C:\Users\qie20\Docume >
学生成绩管理系统
-----
1. 创建数据
2. 输出 JSON
3. 插入节点
4. 删除节点
5. 删除指定成绩的节点
6. 保存到文件
7. 从文件加载
8. 保存到 CSV
9. 遍历输出
0. 退出
-----
请输入选项: 7

[INFO] menu_load(): 操作完成!

请按任意键继续. . . |
```



```
C:\Users\qie20\Docume X + - □ X
学生成绩管理系统
-----
1. 创建数据
2. 输出 JSON
3. 插入节点
4. 删除节点
5. 删除指定成绩的节点
6. 保存到文件
7. 从文件加载
8. 保存到 CSV
9. 遍历输出
0. 退出
-----
请输入选项：9

学号      | 姓名    | 成绩
-----
2109010215 | 杜思维  | 90

请按任意键继续... |
```

8.输出 JSON



```
C:\Users\qie20\Docume X + - □ X
学生成绩管理系统
-----
1. 创建数据
2. 输出 JSON
3. 插入节点
4. 删除节点
5. 删除指定成绩的节点
6. 保存到文件
7. 从文件加载
8. 保存到 CSV
9. 遍历输出
0. 退出
-----
请输入选项：2

[
  {
    "sno": 2109010215,
    "name": "杜思维",
    "score": 90
  },
  {
    "sno": 2109010235,
    "name": "测试",
    "score": 85
  },
  {
    "sno": 2109010238,
    "name": "测试2",
    "score": 88
  }
]

请按任意键继续. . . |
```


9. 保存到 CSV

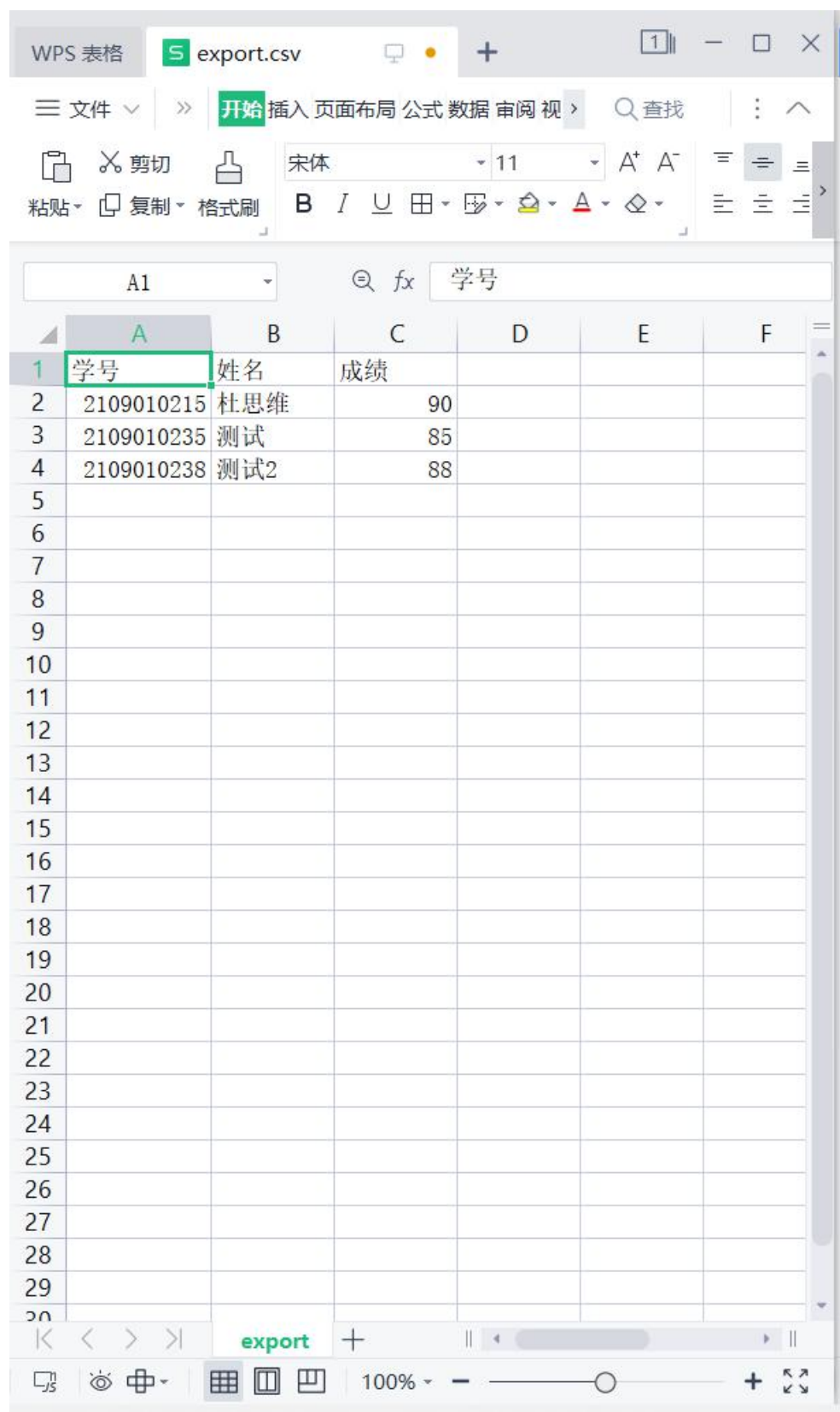


```
C:\Users\qie20\Docume X + - □ X
学生成绩管理系统
-----
1. 创建数据
2. 输出 JSON
3. 插入节点
4. 删除节点
5. 删除指定成绩的节点
6. 保存到文件
7. 从文件加载
8. 保存到 CSV
9. 遍历输出
0. 退出
-----
请输入选项：8

[INFO] menu_csv(): 操作完成!

请按任意键继续. . . |
```

用 Excel 打开



3. 总结

- (1) 实验完成功能
- (2) 实验创新点（例如添加菜单功能等）
- (3) 列举程序编写中遇到的问题，及解决方法

- (1)
 - 1. 定义包含学生成绩信息的结构体类型 **Student**, 每个节点包含学生的学号、姓名和成绩;
 - 2. 定义包含学生信息的单链表结构体 **LinkList**;
 - 3. 用尾插法创建带头结点的单链表;
 - 4. 遍历单链表, 输出所有的学生信息;
 - 5. 在单链表中的指定位置插入新节点;
 - 6. 在单链表中删除指定位置的节点;
 - 7. 删除单链表中成绩为 **x** 的学生信息。
 - 8. 给程序添加足够的提示信息, 例如 **scanf** 之前, 添加一条提示语句“请输入学号: ”。
 - 9. 给程序添加菜单, 用户可以根据需要选择相应菜单, 执行不同操作。
 - 10. 尝试使用文件来读取学生的基本信息, 创建学生的单链表。
- (2)
 - 1. 将程序拆分为多个模块实现, 提高程序的复用性和可维护性
 - 2. 添加 **JSON** 和 **CSV** 输出功能, 方便将输出的内容导入到其它程序中处理 (如 **CSV** 可导入到 **Excel**)
 - 3. 添加了简单的日志系统, 方便调试输出
- (3)
 - 1. 有一处逻辑上需要类似于 **Java** 中 **try...catch...** 的跨函数跳转, 使用 **longjmp** 解决
 - 2. 源文件为 **UTF-8** 编码, 而 **Windows** 使用 **GBK** 编码, 存在各种乱码错误, 常见的解决方案有如下几种:
 - 1) 通过在程序最开始加入 **system("chcp 65001");** 来将控制台切换为 **UTF-8** 编码, 但由于 **Windows** 本身存在的设计缺陷, 汉字将无法输入进程序
 - 2) 将所有的 **char** 类型的多字节字符替换为 **wchar_t** 类型的宽字节字符, 双引号的字符串也要加上前导 **L** 转变类型, **Windows** 会自动处理编码相关问题, 但此操作过于繁琐, 而且类型的转变也会带

来程序空间复杂度的膨胀

3) 将源文件的编码类型更改为 **GBK**，但如今 **UTF-8** 为国际通用的首选编码

最终经过多种尝试,发现通过在编译指令中加入 **-fexec-charset=GBK** 在编译期间将编码动态转换为 **GBK**，不仅保留了 **UTF-8** 的源文件，能保证程序在 **GBK** 编码下运行。