

程序分组设计训练 实验 4

实验报告

学期：2022-2023 第二学期

学院：计算机与信息技术学院

姓名：张鲮泮

学号：22281052

班级：计算机 2202 班

编制日期：2023 年 5 月 22 日

目 录

实验 文件——外部程序调用函数

1 文件——外部程序调用函数.....	1
1.1 结果展示	1
1.1.1 大体结构	1
1.1.2 自动模式	2
1.1.3 交互模式	5
1.2 相关问题回答	6
1.2.1 问题 1	6
1.2.2 问题 2	7
1.2.3 问题 3	11
1.2.4 问题 4	12

实验总结

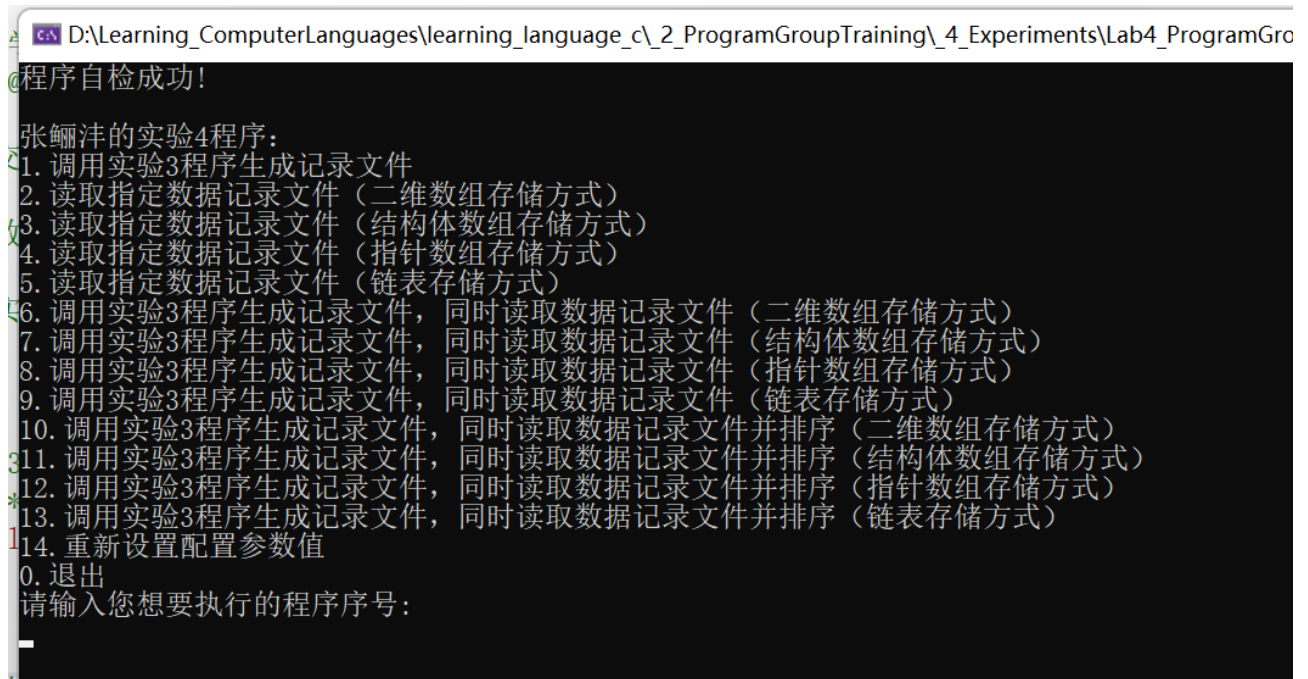
2 遇到的问题及解决办法.....	14
2.1 程序与调试相关问题	14
2.1.1 头文件混乱——LNK2005 报错	14
2.1.2 静态链接与动态链接——LINK1120、LINK2001 报错	16
2.2 程序调试中发现的相关知识点	18
2.2.1 什么是动态链接与静态链接	18
3 实验的收获与心得	21
参考文献	21
相关附件:	22

实验 文件——外部程序调用函数

1 文件——外部程序调用函数

1.1 结果展示

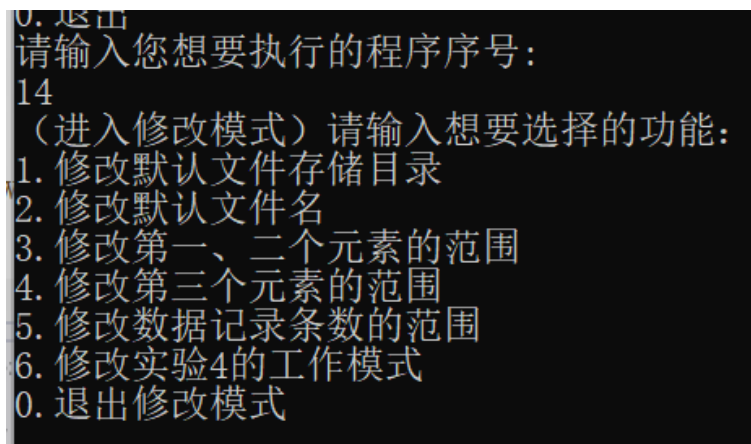
1.1.1 大体结构



```
D:\Learning_ComputerLanguages\learning_language_c\2_ProgramGroupTraining\4_Experiments\Lab4_ProgramGro
@程序自检成功!

张颢洋的实验4程序:
1. 调用实验3程序生成记录文件
2. 读取指定数据记录文件 (二维数组存储方式)
3. 读取指定数据记录文件 (结构体数组存储方式)
4. 读取指定数据记录文件 (指针数组存储方式)
5. 读取指定数据记录文件 (链表存储方式)
6. 调用实验3程序生成记录文件, 同时读取数据记录文件 (二维数组存储方式)
7. 调用实验3程序生成记录文件, 同时读取数据记录文件 (结构体数组存储方式)
8. 调用实验3程序生成记录文件, 同时读取数据记录文件 (指针数组存储方式)
9. 调用实验3程序生成记录文件, 同时读取数据记录文件 (链表存储方式)
10. 调用实验3程序生成记录文件, 同时读取数据记录文件并排序 (二维数组存储方式)
11. 调用实验3程序生成记录文件, 同时读取数据记录文件并排序 (结构体数组存储方式)
12. 调用实验3程序生成记录文件, 同时读取数据记录文件并排序 (指针数组存储方式)
13. 调用实验3程序生成记录文件, 同时读取数据记录文件并排序 (链表存储方式)
14. 重新设置配置参数值
0. 退出
请输入您想要执行的程序序号:
_
```

图 1-1-1-1 程序自检报告与主菜单目录



```
0. 退出
请输入您想要执行的程序序号:
14
(进入修改模式) 请输入想要选择的功能:
1. 修改默认文件存储目录
2. 修改默认文件名
3. 修改第一、二个元素的范围
4. 修改第三个元素的范围
5. 修改数据记录条数的范围
6. 修改实验4的工作模式
0. 退出修改模式
```

图 1-1-1-2 选项 14 与内部菜单目录

1.1.2 自动模式

当所需读取文件不存在时，运行 2/3/4/5 选项：

```
0. 退出
请输入您想要执行的程序序号:
5
指定文件不存在
-842150451
张鲔泮的实验4程序
```

图 1-1-1-3 文件不存在运行 5 选项

```
0. 退出
请输入您想要执行的程序序号:
4
指定文件不存在
-842150451
```

图 1-1-1-4 文件不存在运行 4 选项

```
0. 退出
请输入您想要执行的程序序号:
3
指定文件不存在
-842150451
```

图 1-1-1-5 文件不存在运行 3 选项

```
0. 退出
请输入您想要执行的程序序号:
2
指定文件不存在
-842150451
```

图 1-1-1-6 文件不存在运行 2 选项

运行 1 选项：

```
0. 退出
请输入您想要执行的程序序号:
1
请输入你需要生成的数据条数（若使用配置文件/命令行参数请输入no）:
no
Use Configuration Files. 使用配置文件/命令行参数。
Whether to enter the file path yourself? Please input yes / no.
是否自行输入文件路径? 请输入yes/no.
no
Use Configuration Files. 使用配置文件。
张鲔泮的实验3程序:
1. 生成数据记录文件（使用二维数组存储）
2. 生成数据记录文件（使用结构体数组存储）
请输入您要执行的功能的编号
2
10 9 68
15 6 3
6 2 38
15 11 78
6 15 4
```

图 1-1-1-7 运行 1 选项

注：Lab3_ProgramGroupTraining.exe 中已经有用户交互模式，因此会出现类似实验 4 中要求的用户交互页面，但是实际上不属于实验 4 程序部分。

之后运行 2/3/4/5 选项（此处只展示 1 项）：

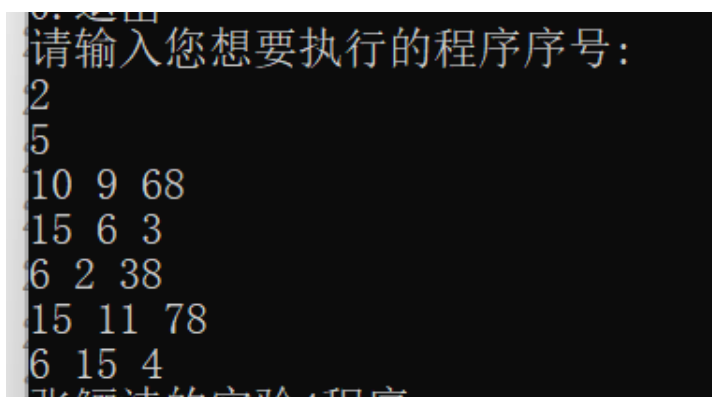


图 1-1-1-8 运行 2 选项

载入的文件内容与生成内容一样（可对比图 1-1-1-7 生成数据）。

运行 6/7/8/9 选项（此处只展示 2 项）：

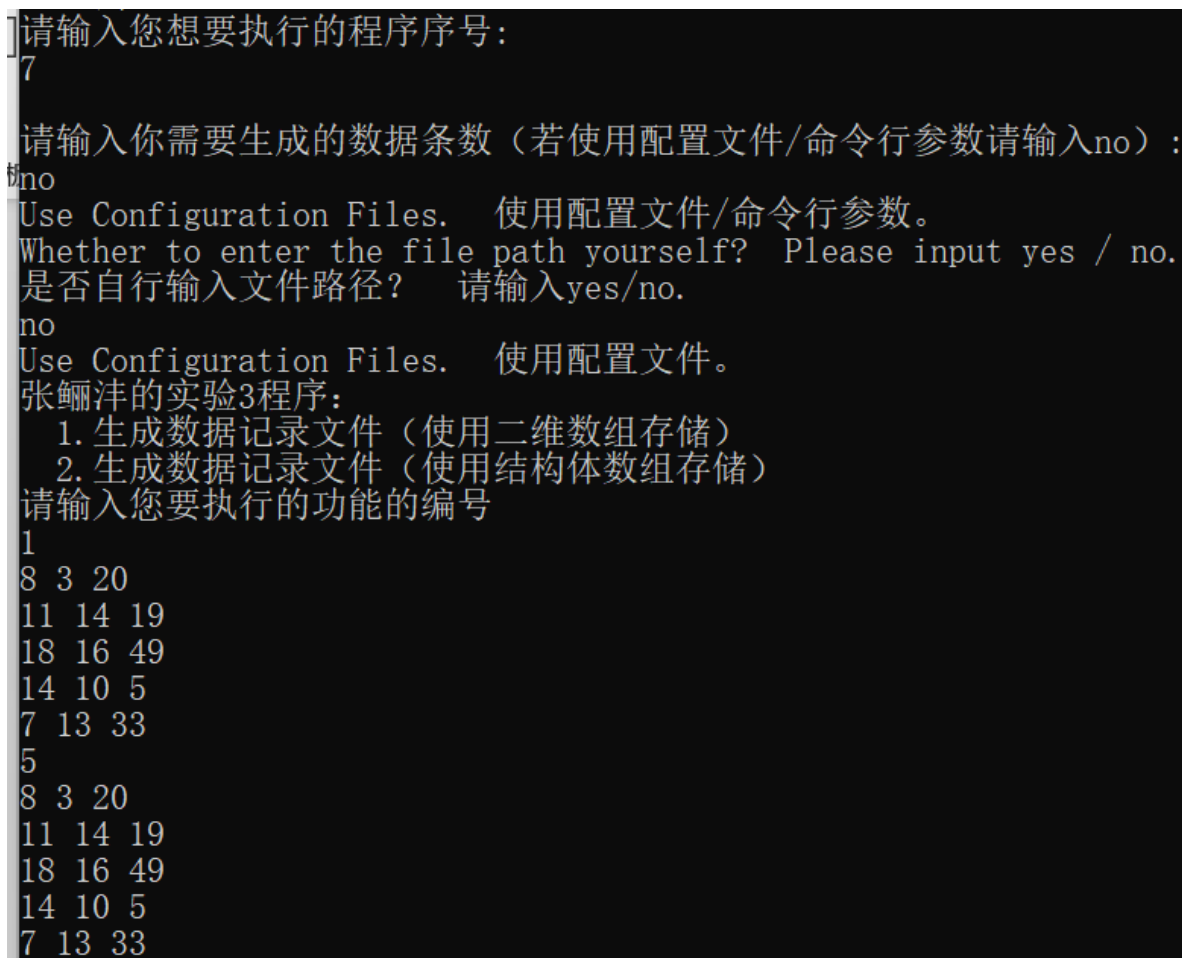


图 1-1-1-9 运行 7 选项

```

请输入您想要执行的程序序号:
9
请输入您需要生成的数据条数 (若使用配置文件/命令行参数请输入no):
no
Use Configuration Files. 使用配置文件/命令行参数。
Whether to enter the file path yourself? Please input yes / no.
是否自行输入文件路径? 请输入yes/no.
no
Use Configuration Files. 使用配置文件。
张颢沅的实验3程序:
1. 生成数据记录文件 (使用二维数组存储)
2. 生成数据记录文件 (使用结构体数组存储)
请输入您要执行的功能的编号
2
17 12 4
5 19 77
9 6 86
1 19 52
4 17 69
5
17 12 4
5 19 77
9 6 86
1 19 52
4 17 69

```

图 1-1-1-10 运行 9 选项

运行 10/11/12/13 选项 (此处只展示 1 项):

```

请输入您想要执行的程序序号:
10
请输入您需要生成的数据条数 (若使用配置文件/命令行参数请输入no):
no
Use Configuration Files. 使用配置文件/命令行参数。
Whether to enter the file path yourself? Please input yes / no.
是否自行输入文件路径? 请输入yes/no.
no
Use Configuration Files. 使用配置文件。
张颢沅的实验3程序:
1. 生成数据记录文件 (使用二维数组存储)
2. 生成数据记录文件 (使用结构体数组存储)
请输入您要执行的功能的编号
2
4 16 75
19 6 11
11 19 78
13 16 78
19 3 63

二维数组排序用时: 0.000000 s

请按任意键继续. . .
5
19 6 11
19 3 63
4 16 75
13 16 78
11 19 78

```

图 1-1-1-11 运行 10 选项

1.1.3 交互模式

以运行 6 选项为例：

```
14. 重新设置配置参数值
0. 退出
请输入您想要执行的程序序号：
6

请输入你需要生成的记录文件名称（可带有绝对路径或相对路径，输入no表示使用配置默认文件名）：
111.txt

请输入你需要生成的数据条数（若使用配置文件/命令行参数请输入no）：
6
Use Configuration Files. 使用配置文件/命令行参数。

请输入你需要生成的数据条数（若使用配置文件/命令行参数请输入no）：
no
Use Configuration Files. 使用配置文件/命令行参数。
张鲔沅的实验3程序：
  1. 生成数据记录文件（使用二维数组存储）
  2. 生成数据记录文件（使用结构体数组存储）
请输入您要执行的功能的编号
2
9 7 49
6 9 7
2 12 45
5 12 34
9 2 34
2 1 91

0
9 7 49
6 9 7
2 12 45
5 12 34
9 2 34
2 1 91
张鲔沅的实验4程序：
请输入您要执行的程序序号：
```

图 1-1-1-12 交互模式下运行 6 选项

红色框是 Lab4 部分程序，蓝色框为 Lab3 部分程序。可以看出在交互模式下，程序成功运行

1.2 相关问题回答

1.2.1 问题 1

设计一个用于实验 4 的配置文件，用于存储实验 4 的工作模式。请在实验 4 程序设计说明书中给出你设计的配置文件的文件名称，相对存储位置，以及标识实验 4 工作模式的方法；

答：

设计的工作模式配置文件的文件名称为：zlf_method.ini。

相对存储位置：在 Lab4.cpp 目录下的 set 文件夹内。（为使文件归纳方便，特地设置 set 文件夹去存放相关文件）

标识实验 4 工作模式的方法：用“1”“2”的方法来标识。“1”为自动模式，“2”为交互模式。

```
/*
*函数名称: void modifysysmethod
*函数功能: 修改实验4工作模式
*输入参数: CONF data 数据参数;
*          confvod* conf 配置文件参数
*返回值: void
*版本信息: create by Lifeng Zhang, 2023-05-16
*/
void modifysysmethod(CONF* data, confvod* conf)
{
    int n;
    printf("请输入实验4的工作模式（1代表自动模式，2代表交互模式）：\n");
    while (true)
    {
        if (scanf("%d", &n) == 1 && n >= 1 && n <= 2)
        {
            FILE* fp;
            errno_t err;
            err = fopen_s(&fp, "set\\zlf_method.ini", "w");
            if (err == 0)
            {
                fprintf(fp, "%d\n", n);
                printf("已修改zlf_method.ini模式文件配置！\n");
            }
            else
                printf("打开lab4配置文件失败\n");
        }
    }
}
```

图 1-2-1-1 设置工作模式函数 modifysysmethod

1.2.2 问题 2

请参考流程图 1-2-2-1 设计实现实验 4 程序，并思考以下问题，在实验报告中加以说明：

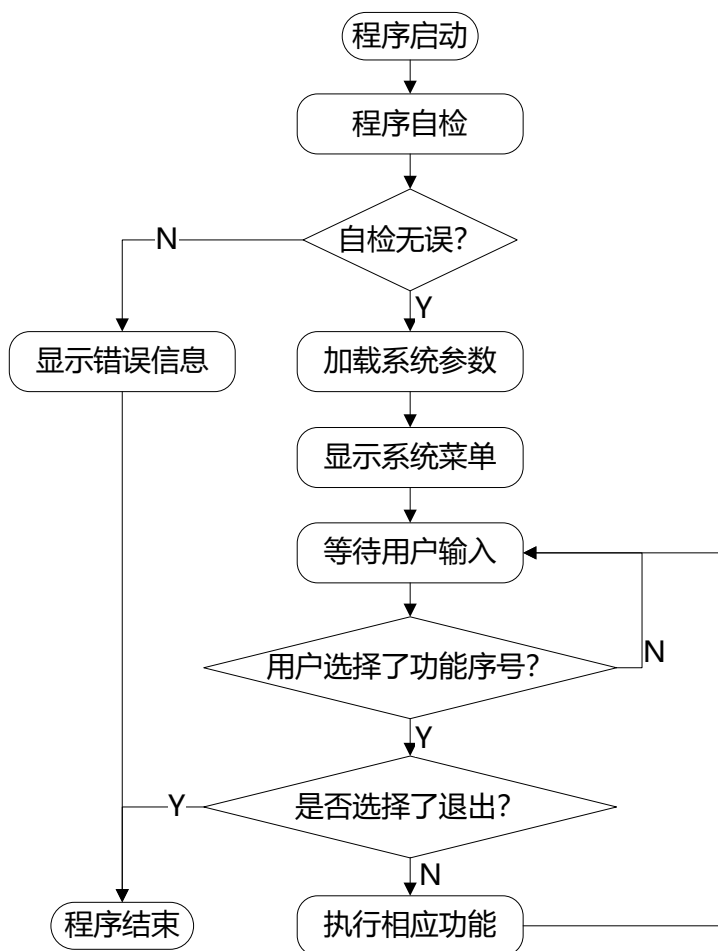


图 1-2-2-1 实验 4 主程序流程图

a) 通常情况下程序启动时的自检过程，目的是检查保障程序运行的各种外部条件是否存在或正常，结合实验程序，思考一下实验 4 的程序自检过程中需要检查什么？请在实验报告中加以阐述；

b) 无论是自动模式还是交互模式下，实验 4 为什么能够准确找到实验 3 生成的数据记录文件？请在实验报告中加以解释；

答：

a) 自检过程中需要检查以下三方面文件是否存在：

①Lab3 相关配置文件；

②Lab4 相关模式文件

③Lab3_ProgramGroupTraining.exe 文件，即需要调用的 Lab4 文件。

当三个文件都存在时，自检完毕。

```
int syscheck()
{
    int count = 0;
    if (access("conf.ini", 0)) // 配置文件
        count++;
    else
        printf("所需文件不存在1——配置文件\n");

    if (access("zlf_method.ini", 0)) // 模式文件
        count++;
    else
        printf("所需文件不存在2——模式文件\n");

    if (access("Lab3_ProgramGroupTraining.exe", 0)) // Lab3.exe文件
        count++;
    else
        printf("所需文件不存在3——Lab3.exe文件\n");

    if (count == 3)
    {
        printf("程序自检成功!\n\n");
        return 1;
    }
}
```

图 1-2-2-2 实验 4 程序自检函数截图

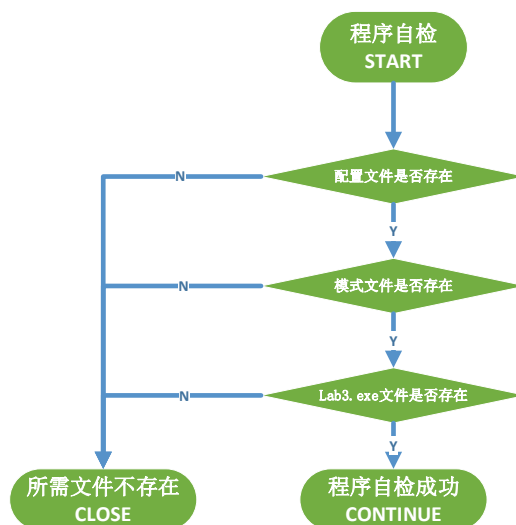


图 1-2-2-3 实验 4 程序自检函数流程图

b) 对于程序自检功能查找文件，本人运用了 **access** 函数。

access 用法如下：

access()函数：用于文件读取，判断文件是否存在，并判断文件是否可写。

access(const char *pathname, int mode); //位于<unistd.h>中。

int _access(const char *pathname, int mode); //位于<io.h>中。

pathname 为文件路径或目录路径；mode 为访问权限。

如果文件具有指定的访问权限，则函数返回 0；如果文件不存在或者不能访问指定的权限，则返回-1。

mode 的值和含义如下所示：

00——只检查文件是否存在

02——写权限

04——读权限

06——读写权限^[1]

使用过程中，发现 access 函数在检查文件是否存在时，无需写明相对路径。

access 在查找文件时，是自动从所含 access 函数的 cpp 文件向所在目录与子目录查找，之后返回相关数字。因此不必写明相对路径。

在自动模式中，通过规定配置文件的参数，设定文件查找位置与数据记录文件生成位置，可以准确找到实验 3 生成的数据记录文件。

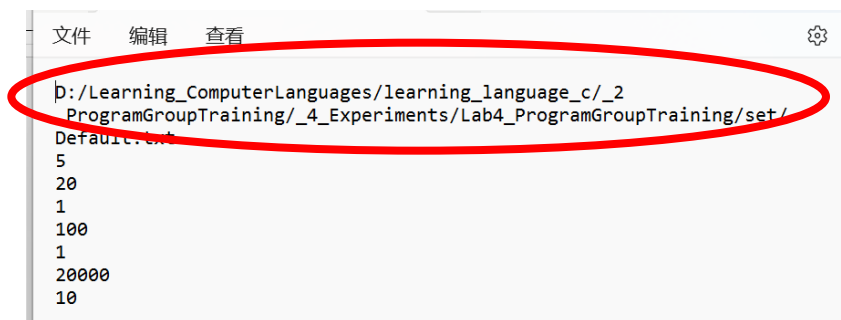


图 1-2-2-4 实验 4 conf.ini 配置文件设定生成文件路径

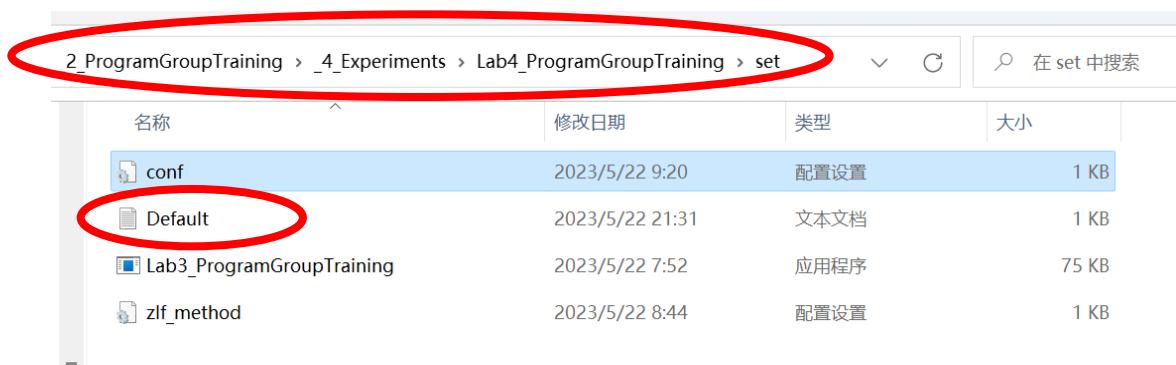


图 1-2-2-4 实验 4 自动模式中生成文件及文件地址

相对存储位置：在 Lab4.cpp 目录下的 set 文件夹内。（为使文件归纳方便，特地设置 set 文件夹去存放相关文件）

1.2.3 问题 3

请使用文件计时函数对二维数组，结构体数据，指向结构体的指针数组三种存储方式的排序过程进行计时，针对 10000 条数据记录规模以上的数据集进行排序计时，采用不同规模数据集进行多次比较，在实验报告中对实验情况进行分析，对三种不同数据容器使用 `qsort` 进行排序的效率进行比较；

答：

以 30,000 条、400,000 数据，分别运行，结果如下：

```
5 17 33
14 17 54
10 8 16

二维数组排序用时: 0.003000 s
请按任意键继续. . .
```

图 1-2-3-1 二维数组排序第 1 次用时

```
9 2 42
12 13 42
19 13 26

二维数组排序用时: 0.045000 s
请按任意键继续. . .
```

图 1-2-3-2 二维数组排序第 2 次用时

```
5 16 61
1 16 77
4 5 15
18 1 8
3 17 57

结构体数组排序用时: 0.033000 s
请按任意键继续. . .
```

图 1-2-3-3 结构体数组排序第 1 次用时

```
14 4 28
15 8 85
14 16 20
1 17 20

结构体数组排序用时: 5.132000 s
请按任意键继续. . .
```

图 1-2-3-4 结构体数组排序第 2 次用时

```
15 3 24
13 2 17
8 1 47
10 6 26

结构体指针数组排序用时: 0.030000 s
请按任意键继续. . .
```

图 1-2-3-5 指针数组排序第 1 次用时

```
11 8 12
19 6 79
3 11 30

结构体指针数组排序用时: 5.228000 s
请按任意键继续. . .
```

图 1-2-3-6 指针数组第 2 次用时

综上所述，二维数组排序最快，结构体数组与指针数组两者排序较慢。

1.2.4 问题 4

请使用三种数据集对链表存储方式的排序过程进行计时，在实验报告中分析比较冒泡排序和快速排序在不同数据集上效率的区别，测试时可以使用的三种数据集包括：

正序数据集：输入链表的数据集是已从小到大排序好的数据集；

逆序数据集：输入链表的数据集是从大到小逆序排序的数据集；

随机数据集：随机排列的数据集；

答：

以 30,000 条，对三种方式分别运行 2 次，结果如下：

（1）冒泡排序

①正序数据集

```
2 4 15
1 3 11
15 9 54

链表排序——冒泡排序用时: 5.296000 s
请按任意键继续. . .
```

图 1-2-4-1 链表冒泡排序正序数据集第 1 次用时

```
12 1 98
14 6 4
7 9 4

链表排序——冒泡排序用时: 5.219000 s
请按任意键继续. . .
```

图 1-2-4-2 链表冒泡排序正序数据集第 2 次用时

②逆序数据集

```
8 18 10
8 4 34
17 7 75
16 8 57

链表排序——冒泡排序用时: 5.303000 s
请按任意键继续. . .
```

图 1-2-4-3 链表冒泡排序逆序数据集第 1 次用时

```
8 18 10
18 17 57
4 5 3
16 10 54

链表排序——冒泡排序用时: 4.942000 s
请按任意键继续. . .
```

图 1-2-4-4 链表冒泡排序逆序数据集第 2 次用时

③随机数据集

```
11 10 6
18 6 55
8 9 47

链表排序——冒泡排序用时: 4.887000 s
请按任意键继续. . .
```

图 1-2-4-5 链表冒泡排序随机数据集第 1 次用时

```
13 7 8
19 6 95
3 10 17

链表排序——冒泡排序用时: 5.270000 s
请按任意键继续. . .
```

图 1-2-4-6 链表冒泡排序随机数据集第 2 次用时

(2) 快速排序

①正序数据集

```
18 2 9
1 3 47
4 8 87

链表排序——快排比较程序用时: 0.011000 s
请按任意键继续. . .
```

图 1-2-4-7 链表快速排序正序数据集第 1 次用时

```
1 4 73
19 1 73
5 3 77
17 16 56

链表排序——快排比较程序用时: 0.010000 s
请按任意键继续. . .
```

图 1-2-4-8 链表快速排序正序数据集第 2 次用时

②逆序数据集

```
15 11 1
16 19 96
13 9 36
4 18 75

链表排序——快排比较程序用时: 0.008000 s
请按任意键继续. . .
```

图 1-2-4-9 链表快速排序逆序数据集第 1 次用时

```
1 8 17
11 13 11
18 9 47
6 16 99
15 9 76
18 19 44

链表排序——快排比较程序用时: 0.008000 s
请按任意键继续. . .
```

图 1-2-4-10 链表快速排序逆序数据集第 2 次用时

③随机数据集

```
6 2 12
4 3 83
17 11 37
9 3 81

链表排序——快排比较程序用时: 0.009000 s
请按任意键继续. . .
```

图 1-2-4-11 链表快速排序随机数据集第 1 次用时

```
7 12 60
8 3 23
2 6 3
18 9 41

链表排序——快排比较程序用时: 0.008000 s
请按任意键继续. . .
```

图 1-2-4-12 链表快速排序随机数据集第 2 次用时

综上所述，无论 3 种数据集，快速排序速度大于冒泡排序速度。

实验总结

2 遇到的问题及解决办法

2.1 程序与调试相关问题

2.1.1 头文件混乱——LNK2005 报错

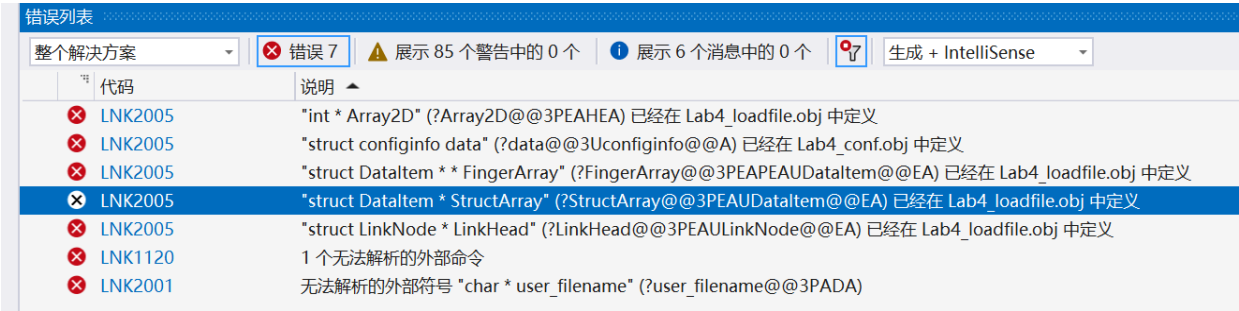


图 2-1-1 LNK2005 报错

编写程序后进行调试时，发现部分时候程序出现错误，并下方报 LNK2005。

查询相关网页解释如下：

通常，此错误意味着违反了唯一定义规则，该规则允许对给定对象文件中使用的任何模板、函数、类型或对象进行唯一定义，且允许在整个可执行文件中对外部可见对象或函数进行唯一定义。

以下是导致此错误的一些常见原因。

1.头文件定义变量时，可能会发生此错误。例如，如果在项目的多个源文件中包含此头文件，会导致错误：

```
// LNK2005_global.h
int global_int; // LNK2005
```

2.头文件定义的函数不是 inline 时，可能会发生此错误。如果将此头文件包含在多个源文件中，则会获得可执行文件中的函数的多个定义。

3.如果在头文件中的类声明之外定义成员函数，也会发生此错误。^[2]

针对原因 1，网页提供了以下解决方案：

1.在头文件: extern int global_int;中声明变量 extern，然后定义它并选择性地在唯一的源文件:int global_int=17;中进行初始化。此变量现在是全局变量，可通过将其声明为 extern（例如，

通过包含头文件)在任何源文件中使用。对于必须是全局变量的变量,建议使用此解决方案,但良好的软件工程做法可以最大程度地减少全局变量。

2.将变量声明为 static: static int static_int=17;。这将定义的范围限制为当前对象文件,并允许多个对象文件拥有自己的变量副本。不建议在头文件中定义静态变量,因为可能会与全局变量混淆。建议将静态变量定义移动到使用它的源文件。

3.将变量声明为 selectany:__declspec(selectany) int global_int=17;。这会使链接器选择唯一定义以供所有外部引用使用,并丢弃其余定义。组合导入库时,此解决方案有时很有用。在其他情况下,不建议使用它来避免链接器错误。^[2]

于是,根据解决方案,判断出以下问题:

在初期编写头文件时,每一个 cpp 文件对应头文件中,都会有其他函数头文件,而其他函数头文件中有次 cpp 头文件,导致循环定义,报错 LINK2005.

因此,整理头文件后, LINK2005 错误减少。

之后将全局变量设声明为 extern, 如下:

```
// 全局变量的声明
extern CONF data;           //
extern int sys_method;       //
extern char filepath[MAX_ARRAY_LEN]; //
extern int record_num;       //

extern int* Array2D;         // 二维
extern DATAITEM* StructArray; // 结构
extern DATAITEM** FingerArray; // 指针
extern LINKNODE* LinkHead;   // 链表
```

图 2-1-2 添加 extern 声明

修改完成后,不报错 LINK2005。但是,报错 LINK1120、LINK2001。

2.1.2 静态链接与动态链接——LINK1120、LINK2001 报错

之后将全局变量设声明为 `extern` 后，出现大量报错 LINK1120、LINK2001，如下图所示：

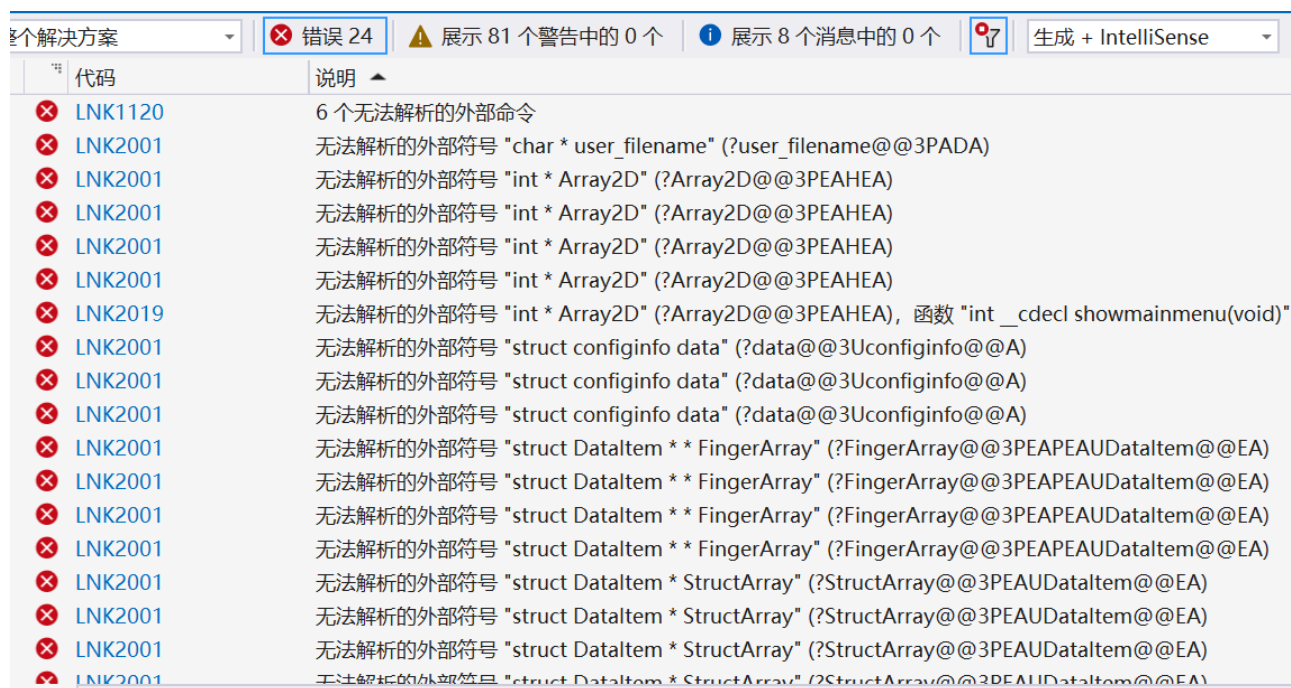


图 2-1-3 LINK1120、LNK2001 报错

查询相关网页解释如下：

编译后的代码引用或调用符号。该符号未在链接器搜索的任何库或对象文件中定义。

此错误消息后为错误 LNK1120。若要修复错误 LNK1120，请首先修复所有 LNK2001 和 LNK2019 错误。

可通过多种方法获取 LNK2001 错误。所有这些方法都涉及对链接器无法解析或查找定义的函数或变量引用。编译器可以识别代码何时未声明符号，但当它不定义符号时，编译器无法识别。这是因为定义可能位于不同的源文件或库中。如果代码引用了某个符号，但从未定义该符号，则链接器会生成错误。^[3]

于是，没看懂。删除 `extern` 后报错 C3861、C2065

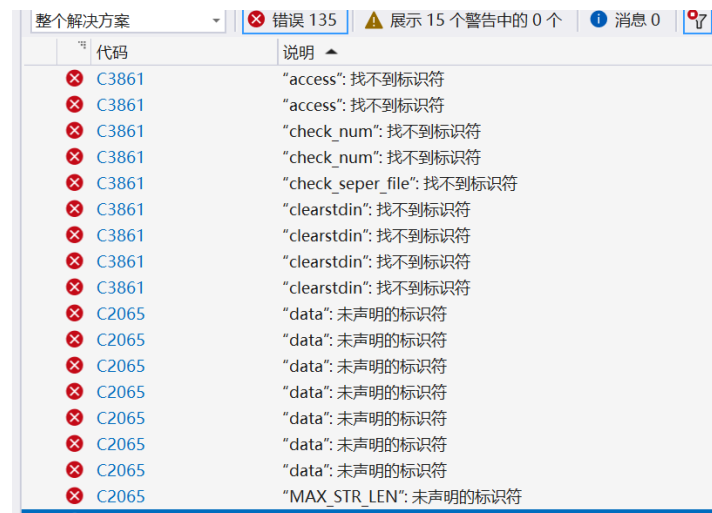


图 2-1-4 C3861、C2065 报错

随后，询问了 2020 级杨圣普学长，通过交流得出问题：使用 extern 导致动态链接有误，以至于无法链接报错 LINK1120、LNK2001。

后续进行程序修改，将所有变量改为静态变量，并且在全部程序函数中添加了传参接口。为方便储存，设定了结构体进行全局变量的声明，如图 2-1-5:

```
typedef struct Conf
{
    int sys_method;           // 实验4工作模式变量，1表示自动模式，2表示交互模式
    char filepath[MAX_ARRAY_LEN]; // 交互模式下储存用户输入的文件储存路径
    int record_num;          // 交互模式下存储用户输入的数据记录条数
    int* Array2D;            // 二维数组
    DATAITEM* StructArray;  // 结构体数组首指针
    DATAITEM** FingerArray; // 指针数组首指针
    LINKNODE* LinkHead;     // 链表头指针

    char user_filename[MAX_STR_LEN];
} confvod;
```

图 2-1-4 全局变量的声明

最后，程序不报错。

2.2 程序调试中发现的相关知识点

2.2.1 什么是动态链接与静态链接

由于动态链接与静态链接的问题，程序报错 LINK1120、LINK2001。因此学习了动态链接与静态链接相关知识。

1. 什么是静态链接？

静态链接是由链接器在链接时将库的内容加入到可执行程序中的做法。链接器是一个独立程序，将一个或多个库或目标文件（先前由编译器或汇编器生成）链接到一块生成可执行程序。这里的库指的是静态链接库。

2. 什么是动态链接？

动态链接把链接这个过程推迟到了运行时再进行，在可执行文件装载时或运行时，由操作系统的装载程序加载库。这里的库指的是动态链接库。

3. 静态链接与动态链接的优缺点？

（1）静态链接的优缺点：

优点：

代码装载速度快，执行速度略比动态链接库快；

只需保证在开发者的计算机中有正确的.lib 文件，在以二进制形式发布程序时不需考虑在用户的计算机上.lib 文件是否存在及版本问题。

缺点：

使用静态链接生成的可执行文件体积较大，包含相同的公共代码，造成浪费。

（2）动态链接的优缺点：

优点：

生成的可执行文件较静态链接生成的可执行文件小；

适用于大规模的软件开发，使开发过程独立、耦合度小，便于不同开发者和开发组织之间进行开发和测试；

不同编程语言编写的程序只要按照函数调用约定就可以调用同一个 DLL 函数；

DLL 文件与 EXE 文件独立，只要输出接口不变（即名称、参数、返回值类型和调用约定不变），更换 DLL 文件不会对 EXE 文件造成任何影响，因而极大地提高了可维护性和可扩展性；

缺点：

使用动态链接库的应用程序不是自完备的，它依赖的 DLL 模块也要存在，如果使用载入时动态链接，程序启动时发现 DLL 不存在，系统将终止程序并给出错误信息；

速度比静态链接慢。^[4]

2.3 程序设计文档

谈一下你对程序设计文档的理解，结合你的体会论述一下程序设计文档应该怎么写、什么时候写、要写清楚的内容是什么、在程序开发中起到什么作用。

程序设计文档是用于记录软件开发过程中的程序设计思路和实现细节的文档。它是在设计程序与软件至关重要的一环，可以帮助开发人员了解系统的整体构架、设计思想以及技术细节，使得团队更加高效协作，提高代码的可读性和可维护性。

程序设计文档应该在需求分析、设计和编码之前编写，以确保开发团队在开发过程中始终清晰地了解所开发的软件系统的结构和功能，并且能够遵循事先确定的规则 and 标准进行开发。在写程序设计文档时，需要注意以下几个方面：

1.结构合理：文档应该按照模块化的方式组织，每个模块都应该包含必要的信息，例如，**接口、参数、返回值**等。

2.内容详尽：文档应该详细记录每个模块的功能描述、接口定义、算法流程和数据结构等内容，包括准确的输入和输出格式、异常处理机制、运行时间复杂度、空间复杂度等。

3.语言简洁明了：文档应该使用简单、易懂的语言描述，避免使用专业术语，使得团队成员可以快速理解和使用。

程序设计文档在程序开发中起到了非常重要的作用。它可以帮助开发团队了解整个系统的架构、功能以及具体实现细节，从而更好地进行协调与合作，避免错误和重复工作，并且对于后期维护和升级也提供了必要的依据。

3 实验的收获与心得

(1) 链表的使用

本次实验是第一次结合动态内存分配与链表，进行随机数生成。通过这次训练，体会到了链表的相关使用规则。

(2) 快速排序的使用

本次实验涉及较多的快速排序的运用，尤其是关于链表的快排，无法调用库中的 `qsort` 函数，得要自行理解快排的原理之后写出来。

(3) 外部程序调用

本次实验重点之一是外部程序的调用。本次程序设计中，运用了 `sprintf()`、`system()` 等函数对 Lab3 的程序进行调用，对相关知识有了一定的了解。

本次实验受益良多，为后面实验 5 与期末考试打下了坚实的基础。希望在以后的学习中有更多的收获。

参考文献

[1] Maccy37. C++ `access()` 函数. https://blog.csdn.net/qq_37611824/article/details/108386923 . 2020-09-03

[2] corob-msft、v-kents、Mikejo5000、mikeblome、ghogen、Saisang. 链接器工具错误 LNK2005. <https://learn.microsoft.com/zh-cn/cpp/error-messages/tool-errors/linker-tools-error-lnk2005?view=msvc-170> . 2022-09-27

[3] corob-msft、Taojunshen、v-kents、Mikejo5000、Mikeblome、ghogen、Saisang. 链接器工具错误 LNK2001. <https://learn.microsoft.com/zh-cn/cpp/error-messages/tool-errors/linker-tools-error-lnk2001?view=msvc-170> . 2022-09-27

[4] wx60b650682e725. C 语言 | 什么是动态链接与静态链接? . https://blog.51cto.com/u_15244533/2845292. 2021-06-02

相关附件：

附件 1: Lab4_ProgramGroupTraining 文件夹 (Lab4 文件夹)。内含源文件代码与编译程序

附件 2: Lab4_22281052_张颢沅_部分函数详细设计.vsd

附件 3: Lab4_22281052_张颢沅_程序设计分组训练实验 4—程序设计说明书.pdf

附件 4: Lab4_运行程序.exe