

# 程序分组设计训练 实验 2

## 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：张龢泮

学号：22281052

班级：计算机 2202 班

编制日期：2023 年 3 月 20 日

# 目 录

## 实验 实现文件自动生成

1 实现文件自动生成 .....	1
1.1 结果展示 .....	1
1.1.1 软件调试结果 <sup>①</sup> .....	1
1.1.2 cmd 窗口运行结果 .....	2
1.2 相关问题回答 .....	2
1.2.1 问题 a & b .....	2
1.2.2 问题 c .....	5
1.2.3 问题 d .....	7
1.2.4 问题 e .....	9

## 实验总结

2 遇到的问题及解决办法 .....	10
2.1 程序与调试相关问题 .....	10
2.1.1 内存不足导致爆栈 .....	10
2.1.2 命令行参数调用 .....	11
2.2 文档排版相关问题 .....	12
2.2.1 插入注释 .....	12
2.2.2 表格跨页 .....	13
3 实验的收获与心得 .....	14
参考文献 .....	15
相关附件: .....	15

# 实验 实现文件自动生成

## 1 实现文件自动生成

### 1.1 结果展示

#### 1.1.1 软件调试结果<sup>①</sup>

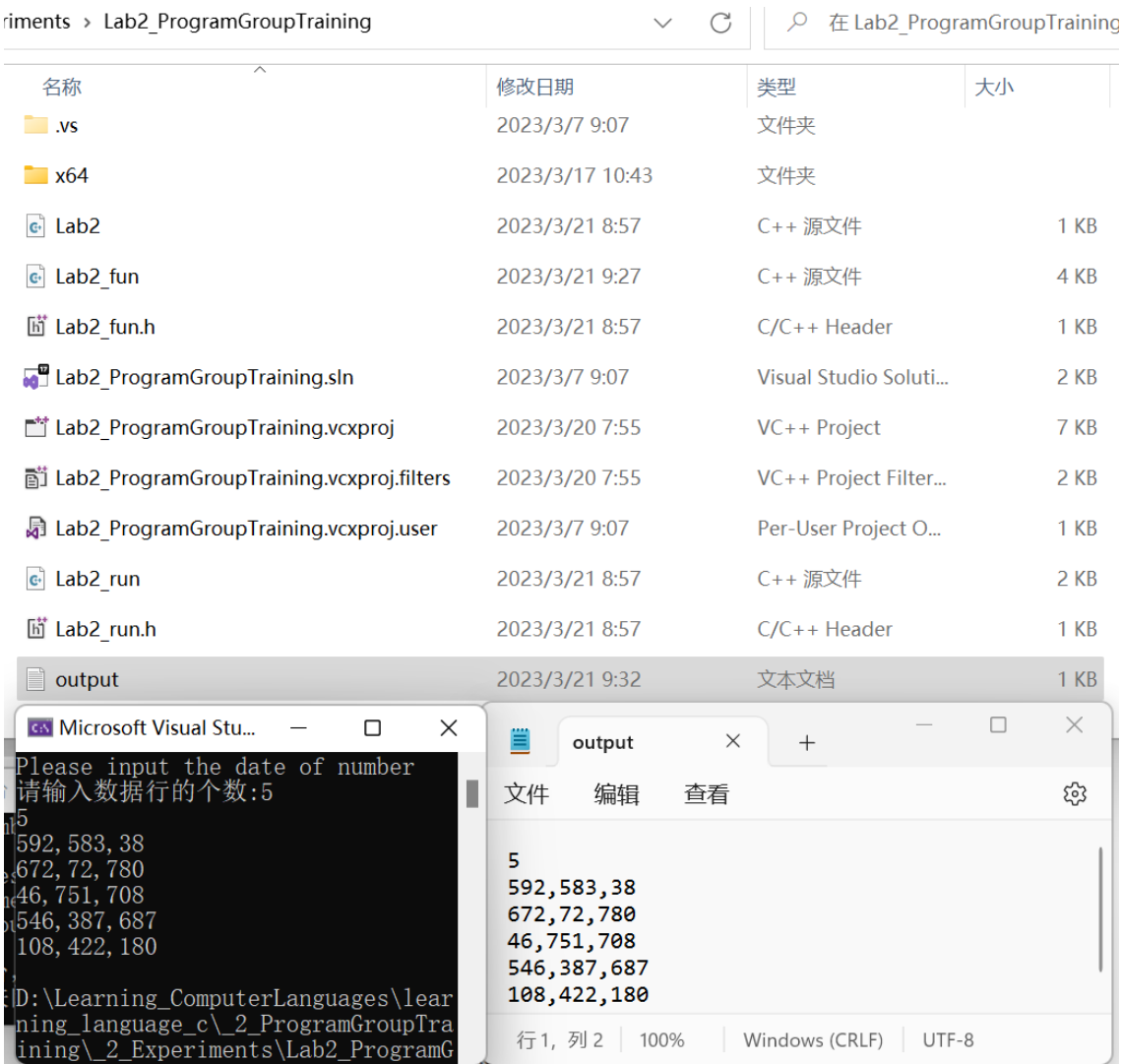
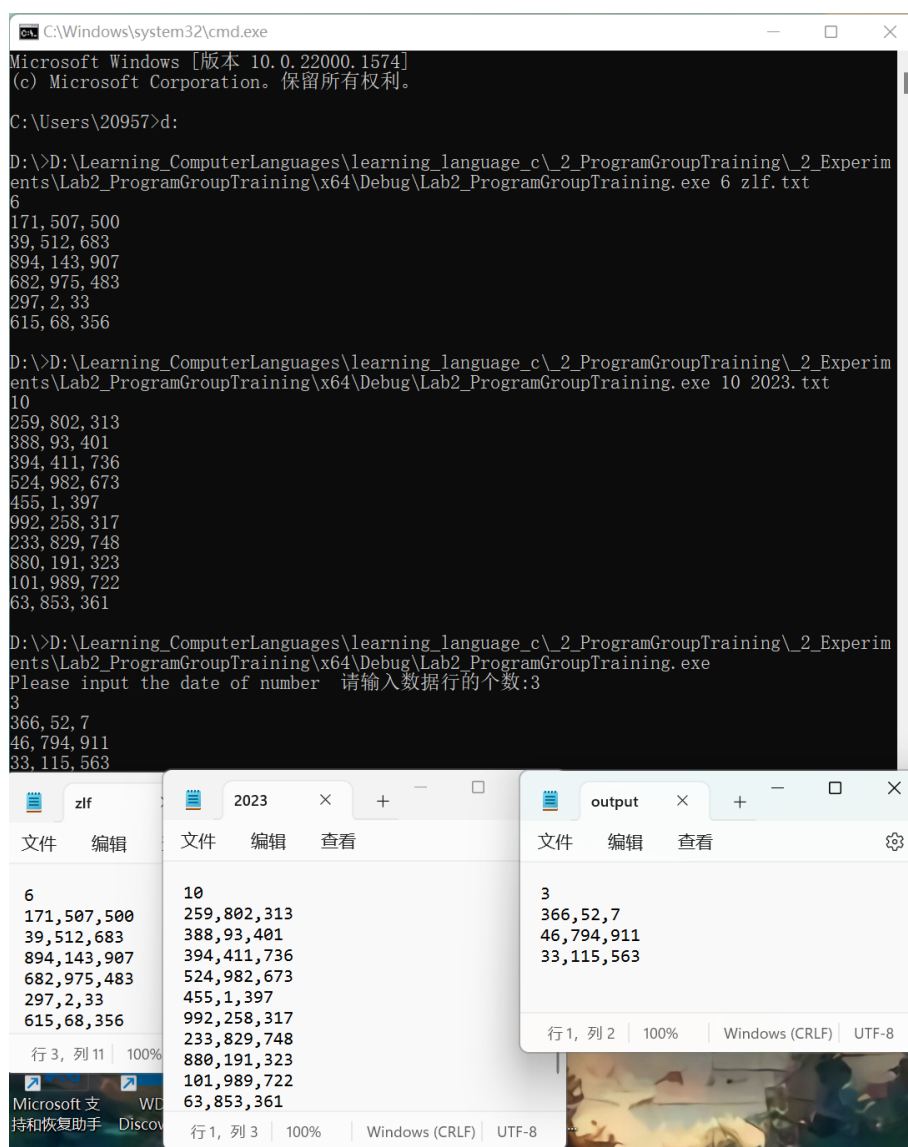


图 1-1-1 Visual Studio 2022 运行结果

① 调试软件为 Visual Studio 2022。

## 1.1.2 cmd 窗口运行结果



```
C:\Windows\system32\cmd.exe
Microsoft Windows [版本 10.0.22000.1574]
(c) Microsoft Corporation。保留所有权利。

C:\Users\20957>d:

D:\>D:\Learning_ComputerLanguages\learning_language_c\_2_ProgramGroupTraining\_2_Experiments\Lab2_ProgramGroupTraining\x64\Debug\Lab2_ProgramGroupTraining.exe 6 zlf.txt
6
171,507,500
39,512,683
894,143,907
682,975,483
297,2,33
615,68,356

D:\>D:\Learning_ComputerLanguages\learning_language_c\_2_ProgramGroupTraining\_2_Experiments\Lab2_ProgramGroupTraining\x64\Debug\Lab2_ProgramGroupTraining.exe 10 2023.txt
10
259,802,313
388,93,401
394,411,736
524,982,673
455,1,397
992,258,317
233,829,748
880,191,323
101,989,722
63,853,361

D:\>D:\Learning_ComputerLanguages\learning_language_c\_2_ProgramGroupTraining\_2_Experiments\Lab2_ProgramGroupTraining\x64\Debug\Lab2_ProgramGroupTraining.exe
Please input the date of number 请输入数据行的个数:3
3
366,52,7
46,794,911
33,115,563
```

图 1-1-2 cmd 窗口运行结果

## 1.2 相关问题回答

### 1.2.1 问题 a & b

a) 程序中哪些程序流程被你封装成了函数，函数实现什么功能，输出参数和返回值是什么，为什么要考虑将这个功能封装为函数。

b) 请在实验报告中用表格罗列您设计的程序函数，说明函数实现代码所在的源文件。

答：本程序中，将以下程序流程封装成了函数：<sup>①</sup>

表 1 被封装的函数程序流程

函数名称	程序流程	实现功能	输出参数	返回值	所在源文件	为什么封装
<code>int run (int argc, char* argv[])</code>	由 Lab2. cpp 入口程序进入到主运行程序	实验 2 函数大体框架	用户输入 1 个或 3 个级以上命令行参数，数据有问题，进行报错	<code>int</code> (无实际含义)	Lab2_run. cpp	题目要求，将接口程序与主程序分开
<code>void create_1 (FILE*, int*, char* argv[]);</code>	当用户未输入相关数据，提醒用户输入数据，之后创建 output. txt 文件	用户未输入数据情况、创建文件	如果没有成功创建文档，进行报错提示	<code>void</code>	Lab2_fun. cpp	使主程序更加清晰明了，并且易辨别函数功能

<sup>①</sup> 此处函数指除 main 函数之外的函数。

续 表 1

void create_2 (FILE*, int*, char* argv[]);	当用户输入相关 数据后, 创建 (argv[2]).txt 文件	用户已输入数据 情况、创建文件	如果没有成功创 建文档, 进行报 错提示	void	Lab2_fun. cpp	使主程序更加清 晰明了, 并且易 辨别函数功能
void data (FILE*, int*, char* argv[]);	在创建文件内写 入记录条数数目 与 n 行数据	进行随机数输出	对文件进行写入 相关数据, 如果 没有成功关闭文 档, 进行报错提 示	void	Lab2_fun. cpp	对于 2 种情况, 需要重复使用此 功能。分装后可 以减少不必要的 代码
int clearstdin();	对 scanf 输入进 行缓存清除, 防 止爆栈	对 scanf 输入进 行缓存清除	无	int (无实际含义)	Lab2_fun. cpp	分装后是程序更 加清晰明了
int check (char*)	对 create_2 函 数输入的 argv[1]检查是 否合法	对 argv[1]数据 是否合法进行检 查	无	int 1: 数据合法 2: 数据非法	Lab2_fun. cpp	分装后是程序更 加清晰明了

1.2.2 问题 c

c) 用户在使用你的程序的时候，如果用户进行以下操作，你的程序是否还能正常运行，如果不能正常运行，原因是什么：

- ① 用户输入了两个命令行参数，但是这两个命令行参数都是数字，  
如：>Lab2.exe 1111 2222;
- ② 用户输入了两个命令行参数，但是这两个命令行参数都是字符，  
如：>Lab2.exe aaaa bbbb;
- ③ 用户没有输入命令行参数，当程序提示用户输入记录条数时，用户没有输入有效数字，而是输入了字母。

答：

- ① 程序能够还能正常运行，生成了文件。

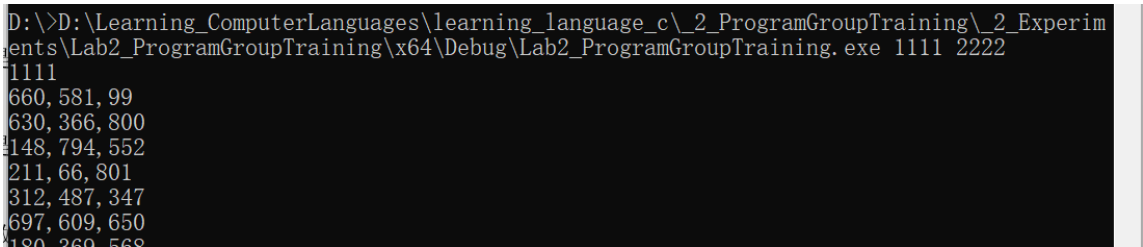


图 1-2-1 第①问程序正常运行截图

但是生成文件需要选择方式打开。

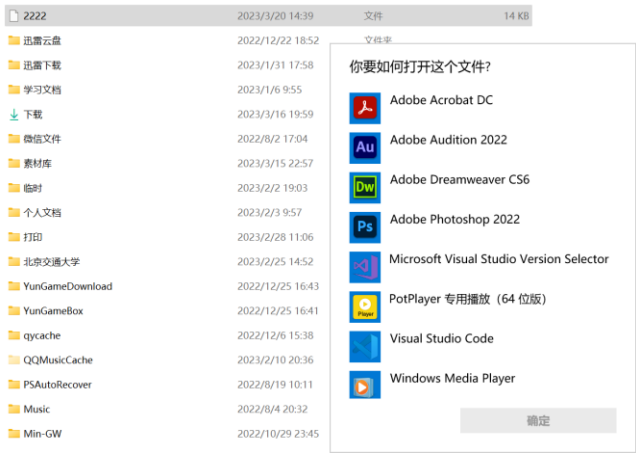


图 1-2-2 第①问程序正常运行截图

选择“记事本”方式打开后，能够正常看到文件内容。

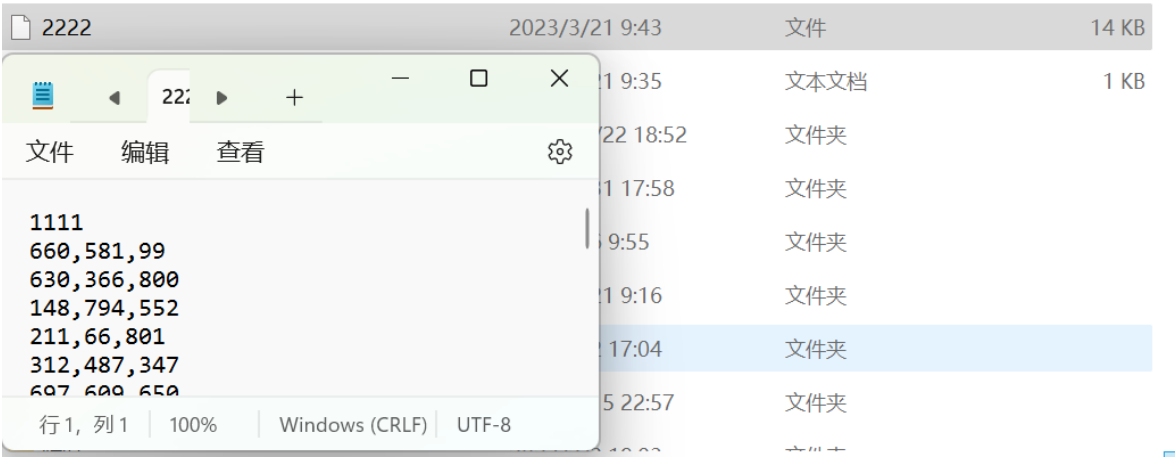


图 1-2-3 第①问文件内容

② 程序能够还能正常运行，未生成文件。在设计函数时已经考虑非法情况。

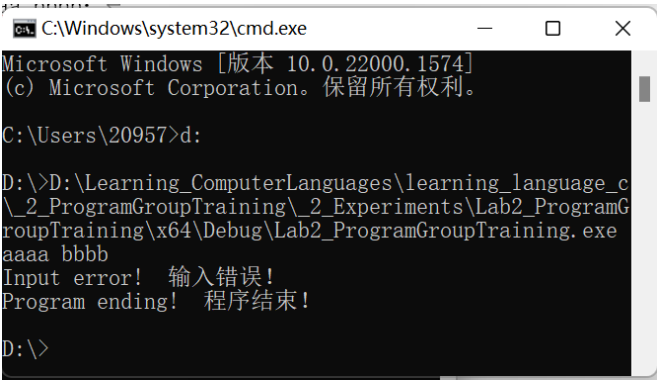


图 1-2-4 第②问程序正常运行截图

③ 程序能够还能正常运行，生成了“output.txt”文件。生成了随机数。

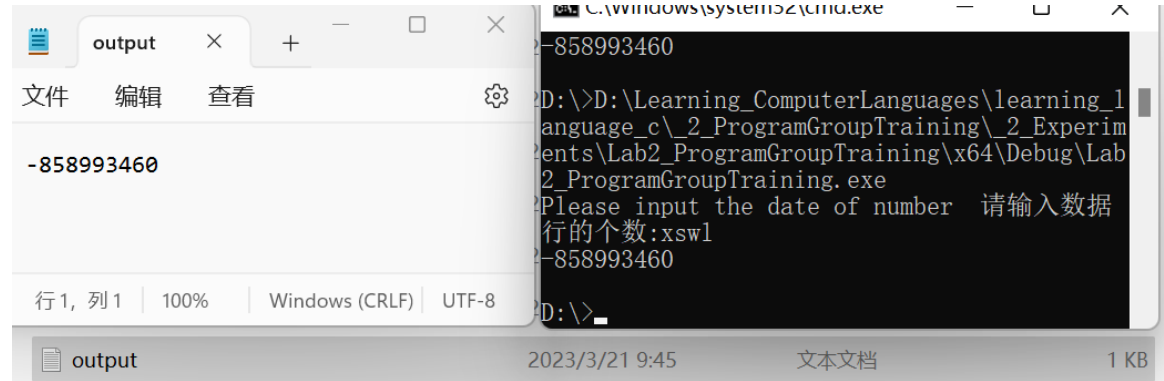


图 1-2-5 第③问程序正常运行截图及文件内容



### 1.2.3 问题 d

d) 用户的输入会千奇百怪，当命令行参数或用户输入的数据不合法时，程序需要给出错误提示，因此程序需要考虑如何检查和处理输入参数不合法的情况，请在实验报告中论述你设计的程序可能需要哪些输入不合法情况。

答：

① 用户输入 1 个或 3 个级以上命令行参数，数据有问题。

此时在 `int run(int argc, char* argv[])` 函数中，当 `argc` 数目非法时，结束程序。

```
switch (argc)           // 检查输入参数数目是否正确
{
case 1: // 用户无数据处理时
    create_1(fp, num, argv);
    break;

case 3: // 用户数据输入合理时
    create_2(fp, num, argv);
    break;

default: // 用户输入1个或3个级以上命令行参数，数据有问题
    printf("Input error! 输入错误! \nProgram ending! 程序结束! \n");
}
return 0;
```

图 1-2-6 命令行参数输入非法判断

② 用户输入 2 个命令行参数，输入数字时输入字母等非法字符。

此时在 `int check(char* str)` 函数中检查，当发现非法时，`check` 函数返回返回值 0。之后程序报错，结束程序。

```
if (check(argv[1]) == 0) // 用户输入第一个的非数字参数，数据有问题
    printf("Input error! 输入错误! \nProgram ending! 程序结束! \n");
```

图 1-2-7 命令行参数数字输入非法判断图 1

```

/*
*函数名称: check
*函数功能: 对argv[1]数据是否合法进行检查
*输入参数: 无
*返回值: int, 如果检查后合法, 返回值为1, 否则为0
*版本信息: create by Lifeng Zhang, 2022-03-20
*/
int check(char* str)
{
    int len = strlen(str);
    int i;
    for (i = 0; i < len; i++)
    {
        if (str[i] < '0' || str[i] > '9')
            return 0;
    }
    return 1;
}

```

图 1-2-8 命令行参数数字输入非法判断图 2

### ③ 用户未输入命令行参数, 进行再次提示。

```

/*
*函数名称: create_1
*函数功能: 用户未输入数据情况、创建文件
*输入参数: FILE*fp:文件指针、int* num:argv[1]文件条数、char* argv[]:命令行参数值
*返回值: void
*版本信息: create by Lifeng Zhang, 2022-03-17
*/
void create_1(FILE* fp, int* num, char* argv[])
{
    int number;
    printf("Please input the date of number 请输入数据行的个数:");
    scanf("%d", &number);
    clearstdin(); // 清缓存, 防止爆栈
}

```

图 1-2-9 命令行参数未输入 (argc==1) 时再次提示

### 1.2.4 问题 e

e) 考虑程序合法性校验逻辑, 修改程序主函数流程图, 给出新的流程图并用红色字体将你增加的逻辑标识出来。

答: 程序图如下:

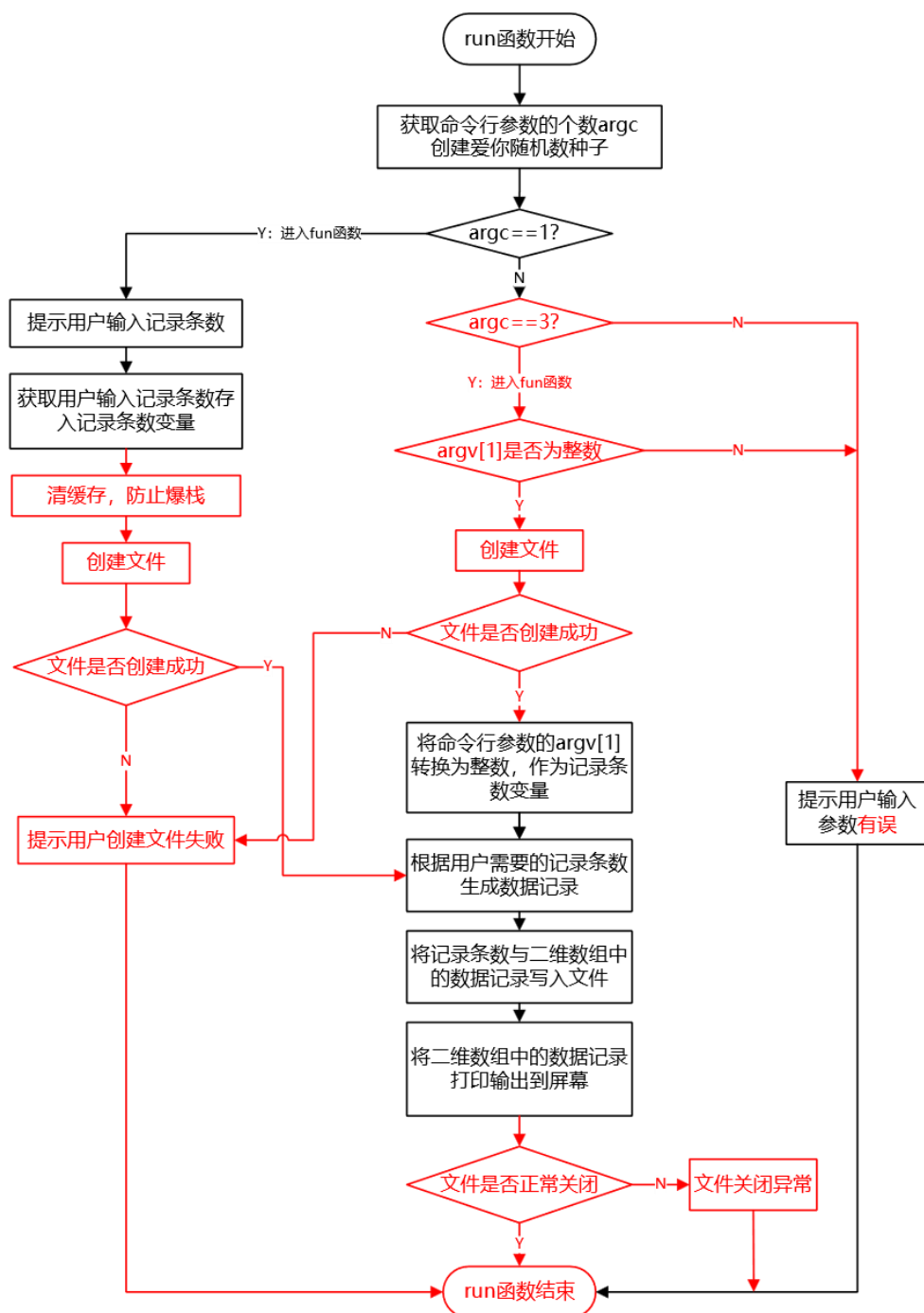


图 1-2-10 修改后的程序主函数流程图

# 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 内存不足导致爆栈

编写程序时，发现部分时候程序出现错误，并下方报错相关栈堆问题。

查询相关网页解释如下：

栈溢出是指函数中的局部变量造成的溢出（注：函数中形参和函数中的局部变量存放在栈上）栈的大小通常是 1M-2M，所以栈溢出包含两种情况，一是分配的的大小超过栈的最大值，二是分配的的大小没有超过最大值，但是接收的 buf 比原 buf 小。函数调用层次过深,每调用一次,函数的参数、局部变量等信息就压一次栈局部变量体积太大。<sup>[1]</sup>

根据编写程序情况，本人判断出是由于分配内存过少导致栈堆溢出。于是本人在相关程序后面加入了清理缓存语句与函数。

```
printf("Please input the date of number 请输入数据行的个数:");  
scanf("%d", &number);  
clearstdin(); // 清缓存, 防止爆栈
```

图 2-1-1 清理缓存语句图 1

```
void data(FILE* fp, int* num, char* argv[])  
{  
    int i, data_3[10000][3];  
    fprintf(fp, "%d\n", *num); // 输出总的数字个数  
    fflush(fp);                // 清缓存, 防止爆栈
```

图 2-1-2 清理缓存语句图 2

## 2.1.2 命令行参数调用

进行程序相关测试，首先用“Win+R”键，调用运行窗口，输入 cmd。

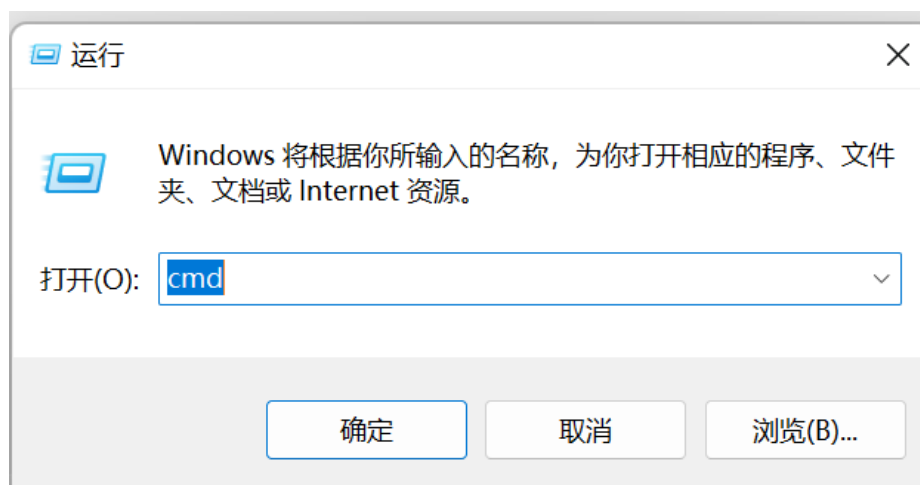


图 2-1-3 运行窗口输入 cmd

由于程序在 D 盘，因此输入“d:”，之后 cmd 窗口就会转到 D 盘进行后续操作。用户输入相关文件地址，之后就可以开始相关程序运行。

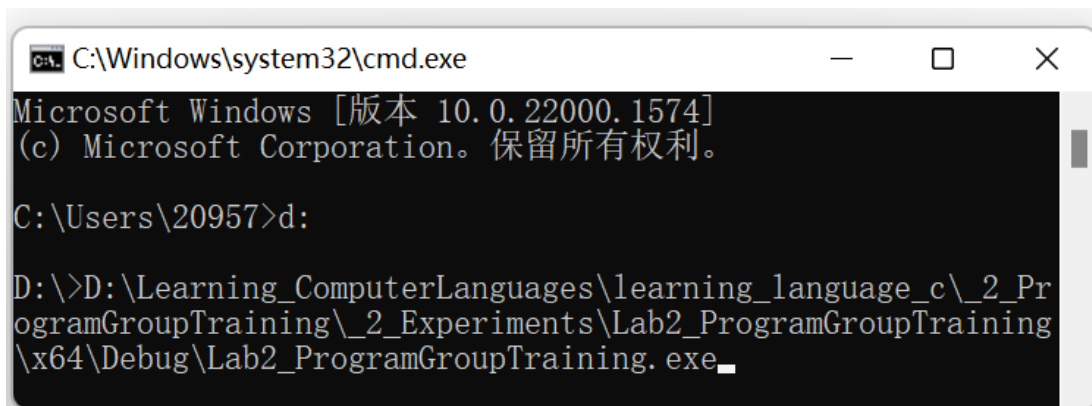


图 2-1-4 cmd 窗口

或者可在软件内进行调试。以 Visual Studio 2022 为例，选择上方工具栏的“调试”中的“xxx 调试属性”。点入后在“配置属性-调试”中的“命令参数”中输入相关数据，即可进行程序编译。

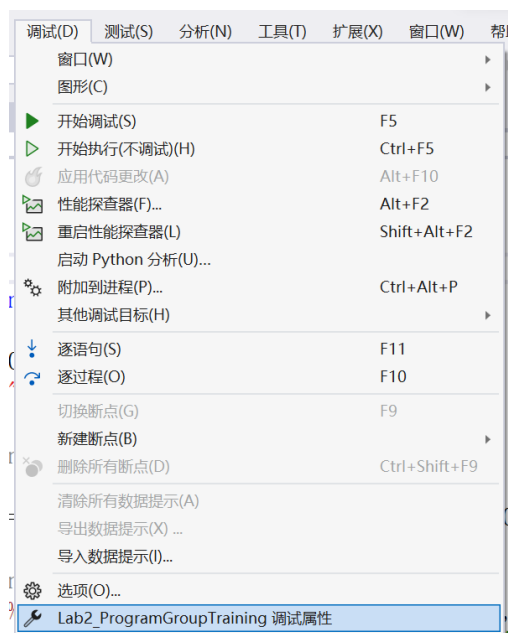


图 2-1-5 Visual Studio 2022 调试栏

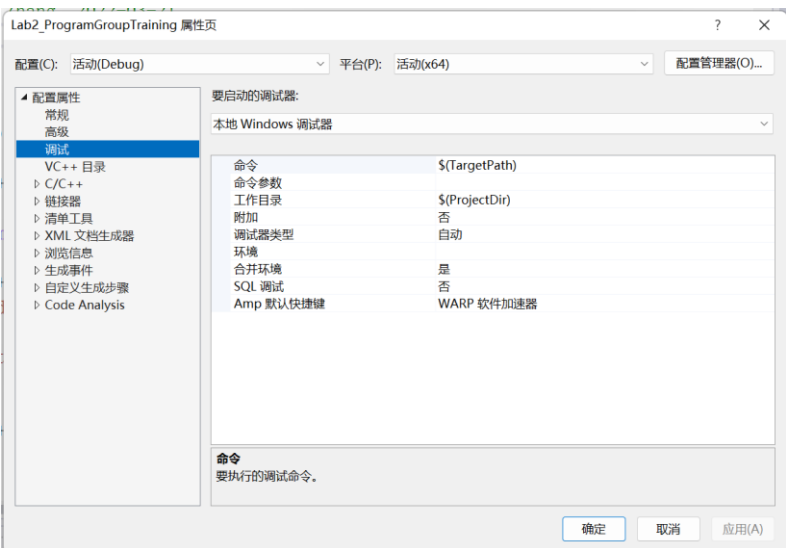


图 2-1-6 配置属性-调试窗口

2.2 文档排版相关问题

2.2.1 插入注释

编写报告时，部分语句需要进行解释，如图 2-2-1（可见正文第 1、3 页）：

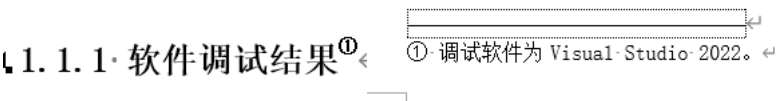


图 2-2-1 “①”注释插入

首先应当在所需插入文字后方，将光标移动到此。之后点击工具栏中的“引用-插入脚注”，进行相关设置即可。

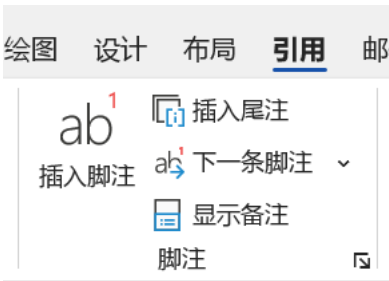


图 2-2-2 引用-插入脚注

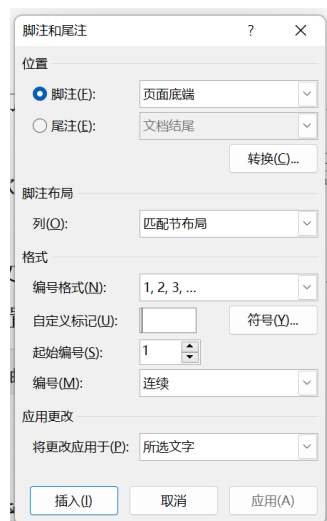


图 2-2-3 脚注与尾注设置界面

2.2.2 表格跨页

制作表格时，表格数据过多会出现表格跨页现象，如图 2-2-4。



图 2-2-4 表格跨页

此时，我们应当在表格第二页上方插入“续表”等类似字样。

此时，应当在第一行第一个字放置光标，键盘上输入“ctrl+shift+enter”，之后就会空出一行，靠右对齐数据相关内容即可。如图 2-2-5。

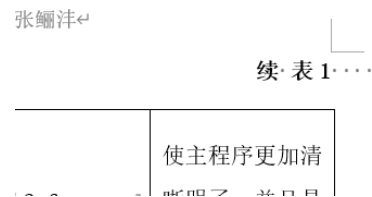


图 2-2-5 表格跨页

### 3 实验的收获与心得

#### (1) 初步运用 cmd 窗口打开运行程序

通过田媚老师课堂演示，本人尝试使用 cmd 窗口，进行了相关实验。了解了相关命令行参数，使用方法，书写规范等。

#### (2) 多文件编程、工程文件相关编写

本次实验是第一次用“程序工程组织与多文件程序开发调试”思想进行程序设计，运用“函数的封装与函数设计”，对部分功能进行了函数分装。了解了头文件、源文件、资源文件等。

#### (3) 代码规范与注解

本次实验要求按照代码格式进行注解。本人通过这次实验，对写注释的习惯有初步的养成。

#### (4) 指针与函数接口

本次实验涉及较多的指针运用。本次程序设计在进行函数调用时多次运用了各类指针，进行相关操作。在设计函数接口，需要考虑头文件、函数参数、传址指针的先关问题。在本次实验对于以上的应用有较大的提升

本次实验受益良多，为后面实验 3、4、5 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] kdyonly. 栈溢出的原因以及解决方法.

<https://cloud.tencent.com/developer/article/2232688#:~:text=%E6%A0%88%E6%BA%A2%E5%87%BA%E6%98%AF%E6%8C%87%E5%87%BD%E6%95%B0%E4%B8%AD,f%E6%AF%94%E5%8E%9Fbuf%E5%B0%8F%E3%80%82> . 2023-03-03

## 相关附件:

附件 1: Lab2 文件夹。内含源文件代码与编译程序

附件 2: Lab2\_仅代码 文件夹。内仅含源文件代码（即 xxx.cpp、xxx.h 文件）

附件 3: Lab2\_程序流程图.jpg

附件 4: Lab2\_程序流程图.vsd

附件 5: Lab2\_运行程序.exe