西安電子科技大學

XIDIAN UNIVERSITY

计算训练项目

姓名：王梦祥

学号：20009100716

指导老师：王斌

实验日期：2021.11

1、实验目的

为了提高学生代码能力和自学能力，提高学生 C 语言的能力，提高学生对批量数据的读取和处理能力，故提出本次计算训练项目。

2、实验内容

2.1 实验环境

电脑操作系统：windows11
编译环境：visual studio 2019
CPU：AMD Ryzen 7 5800H with Radeon Graphics 3.20 GHz
内存：16GB

2.2 实验思路

（1）通过程序打开对应文件夹下的 50 个 dat 文件。

为了实现这个功能，我首先对一些基本数据储存的数据结构进行了定义，分别用结构体链表、结构体数组以及结构体数组表示储存原始数据、储存所需结果文件以及输出结果文件。随后，我将这个功能封装到 Input()函数中。具体实现是通过循环函数进行读取文件，其中涉及 FILE、sprintf_s、fopen_s 等函数。之后，我将读出的数据库放到结构体链表中储存，其中涉及 fscanf_s、atof()等函数。

（2）将文件中 v、u 进行运算，分别计算其方差以及协方差。

这个部分总体思想就是对所得数据进行计算，考虑到不同数据的处理方式，这个部分就封装到了 Operation 函数中。具体代码就是实现方差和协方差的具体过程。

（3）将结果保存到一个 dat 文件中。

这个部分和打开文件具有类似的思想，我将其封装为 Output()函数。具体实现过程涉及 fopen_s、fprint_s、fclose()等函数，找到文件储存的地址，随后即可进行文件输出。值得注意的是，具体赋值文件地址时，要注意字符串长度，放置地址过长，超过定义储存范围。

3、代码展示

```
//代码运行时间：2021年11月26日
//代码作用：将文件中v、u进行运算，分别计算其方差以及协方差
```

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

//存储原始数据
struct value {
    char x[5], y[5], xval[20], yval[20], u[20], v[20], l[20];
    struct value* next;
}*head;

//储存所需结果文件
struct values {
    double x[10336], y[10336], u[10336], v[10336];
}vals[50];

//输出结果文件
struct Restress {
    double a;
    double b;
    double c;
}result[10336];

char fileloc[400];

//函数初始化
void Input(void);
void Operation(void);
void Output(void);

//主函数
int main()
{

    printf("请输入读取文件的路径：(例如：D:\\Origdata）\n");
    gets_s(fileloc);

    //读取数据
    Input();
    //处理数据
    Operation();
    //输出数据
    Output();

    return 0;
}

//输入函数
void Input(void) {
    FILE* fp;
    char filename[400];
    int flag, i, m = 0, j;
    struct value* p1, * p2;

    //依次读入每个文件

    printf("\n开始读取文件名字信息\n\n");
```

```
while (m < 50) {
    if (m < 10) {
        sprintf_s(filename, "%s\\Export-dat.6rw80dwa.00000%d.dat", fileloc, m);
    }
    else {
        sprintf_s(filename, "%s\\Export-dat.6rw80dwa.0000%d.dat", fileloc, m);
    }
    printf("已读取文件:   ");
    printf("%s\n", filename);

    //打开文件
    fopen_s(&fp, filename, "r");

    //将文件信息存入数据库，将读出的信息存放在链表中
    if (fp == NULL) {
        printf("未能初始化第%d个文件的信息\n", m);
        return;
    }

    else {
        flag = 0;
        i = 1;
        j = 0;
        p1 = (struct value*)malloc(sizeof(struct value));
        head = p1;
        p2 = NULL;

        while (!feof(fp) && flag < 10337) {
            //前三行读取
            //interaction：跳过n行，则多输入n行如下代码
            if (flag == 0) {
                fscanf_s(fp, "%*[^\n]%*c");
                fscanf_s(fp, "%*[^\n]%*c");
                fscanf_s(fp, "%*[^\n]%*c");
                flag = 1;
            }

            //之后每行进行读取
            //interaction：对应文件有m个数据，则对应读取相应的数据
            fscanf_s(fp, "%s", p1->x, 5);
            fscanf_s(fp, "%s", p1->y, 5);
            fscanf_s(fp, "%s", p1->xval, 20);
            fscanf_s(fp, "%s", p1->yval, 20);
            fscanf_s(fp, "%s", p1->u, 20);
            fscanf_s(fp, "%s", p1->v, 20);
            fscanf_s(fp, "%s\n", p1->l, 20);

            p2 = p1;
            p1 = (struct value*)malloc(sizeof(struct value));
            p2->next = p1;

            vals[m].x[j] = atof(p2->x);     //字符串转浮点
            vals[m].y[j] = atof(p2->y);
            vals[m].u[j] = atof(p2->u);
            vals[m].v[j] = atof(p2->v);

            j++;
            flag++;
```

```
            }

            p2->next = NULL;
            p2 = NULL;

            free(p1);
            fclose(fp);
        }
        m++;
    }
    return;
}

//操作函数
void Operation(void) {
    //interaction：可以依据具体计算结构进行调整计算
    double sumu[10336] = { 0 }, sumv[10336] = { 0 }, sumuv[10336] = { 0 };
    double sumuu[10336] = { 0 }, sumvv[10336] = { 0 }, averu[10336] = { 0 },
averv[10336] = { 0 };

    for (int i = 0; i < 10336; i++) {
        for (int m = 0; m < 50; m++) {
            sumu[i] = sumu[i] + vals[m].u[i];
            averu[i] = sumu[i] / 50;

            sumv[i] = sumv[i] + vals[m].v[i];
            averv[i] = sumv[i] / 50;

            sumuv[i] = sumuv[i] + vals[m].u[i] * vals[m].v[i];
            sumuu[i] = sumuu[i] + vals[m].u[i] * vals[m].u[i];
            sumvv[i] = sumvv[i] + vals[m].v[i] * vals[m].v[i];
        }

        result[i].a = averu[i] * averv[i] - (sumuv[i] / 50);
        result[i].b = averu[i] * averu[i] - (sumuu[i] / 50);
        result[i].c = averv[i] * averv[i] - (sumvv[i] / 50);
        //printf("%lf %lf %lf\n", result[i].a, result[i].b, result[i].c);
    }
}

//输出函数
void Output(void){
    //interaction：可以依据具体计算结果进行调整输出格式，以及调整文件保存位置

    //以“写”方式打开结果文件
    FILE* fp;
    fopen_s(&fp, "D:\\Origdata\\output.dat", "w");

    if (fp == NULL) {
        printf("\n保存文件不正常，请核对文件名！\n");
        fclose(fp);
    }

    else {
        rewind(fp);  //使得指针指向文件开头

        //输出文件开头的内容
```

```
        fprintf(fp, "TITLE=DynamicStudio Exported Data\n");
        fprintf(fp, "VARIABLES = \"averageU[-]\" \"averageV[-]\" \"averageUV[-]\"
\n");
        fprintf(fp, "ZONE T = \"DynamicStudio Data\" I = 136 J = 76 F = POINT\n");

        for (int i = 0; i < 10336; i++) {
            fprintf(fp, "%lf %lf %lf\n", result[i].a, result[i].b, result[i].c); //输
出一行，换一行
        }

        fclose(fp);
    }
    return;
}
```

4、实验结果

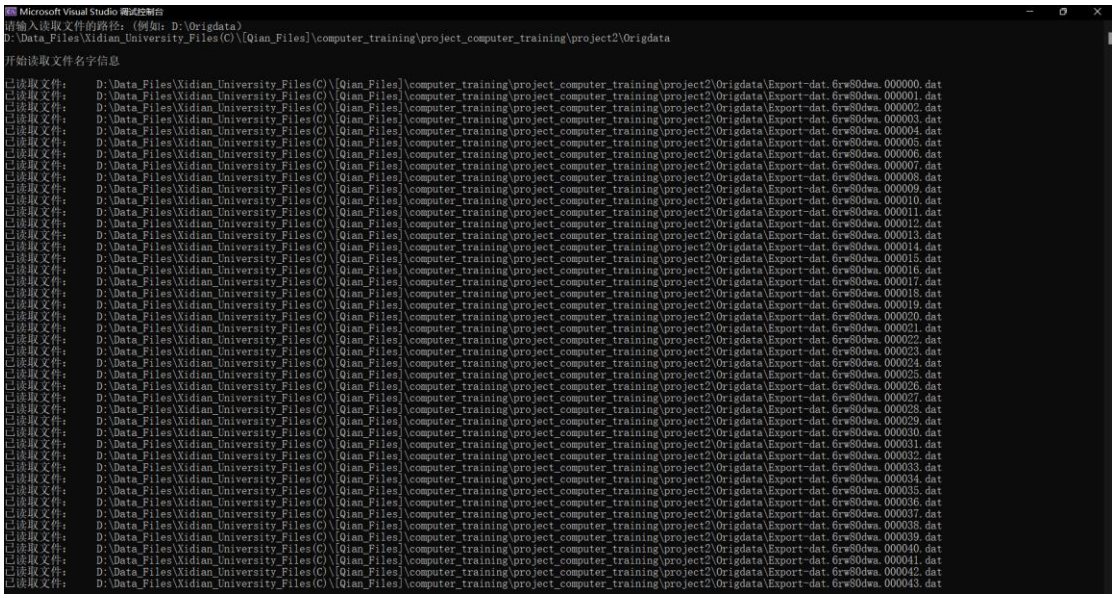

图 1 调试控制台运行过程

输出到文件中的结果（详细结果见附录）：

```
TITLE=DynamicStudio Exported Data
VARIABLES = "averageU[-]" "averageV[-]" "averageUV[-]"
ZONE T = "DynamicStudio Data" I = 136 J = 76 F = POINT
-230123.707417 -637227.314736 -1832719.617835
-97600.455110 -724735.229581 -2391803.102271
-678715.837929 -1366798.712300 -2030805.397542
-114775.043678 -1634077.459096 -1719162.424462
-750756.187033 -2487785.946827 -1649112.474839
-283822.675594 -1694419.884195 -1840548.675351
……
```

5、感想与收获

总体来说，通过一个比较全面的程序编写经历，对一个有实践性系统的代码有了比较全面的把握，为之后多场景代码编写奠定了一个初步的基础。

（1） 学会了对多文件处理。通过一些基本的 C 语言函数，学会了批量打开、处理、输出某些特定位置的数据。

（2） 提高了用户交互的代码意识，通过输入输出端、多文件代码管理等方式，为之后代码的移植和代码的可读性。

附录：输出结果详细数据（部分展示）

```
TITLE=DynamicStudio Exported Data
VARIABLES = "averageU[-]" "averageV[-]" "averageUV[-]"
ZONE T = "DynamicStudio Data" I = 136 J = 76 F = POINT
-230123.707417 -637227.314736 -1832719.617835
-97600.455110 -724735.229581 -2391803.102271
-678715.837929 -1366798.712300 -2030805.397542
-114775.043678 -1634077.459096 -1719162.424462
-750756.187033 -2487785.946827 -1649112.474839
-283822.675594 -1694419.884195 -1840548.675351
-633658.296413 -1259147.099274 -1960605.509137
-476353.865984 -1624055.914898 -2090532.662351
-318087.486110 -2307892.402413 -1933979.491389
-7061.566552 -2053778.006709 -2229036.772395
152931.009285 -2550172.023663 -2675435.670912
-180394.743493 -2240639.307039 -2503136.530402
554822.612267 -1690090.504790 -2804069.758015
535751.972182 -1403785.501856 -2633711.203794
869130.092779 -2052104.435816 -3301541.279280
-96566.249984 -1401230.594836 -3300017.213112
248991.854890 -1209892.260142 -4123745.877404
378383.244929 -1064443.479114 -2826527.464123
512994.281583 -1198958.778735 -3519443.358837
302250.532159 -1297840.804001 -2409856.617494
542662.909060 -1482168.771687 -2379375.386488
227839.506549 -1408966.003852 -2372985.669115
322561.796995 -1129299.744776 -2422915.275762
-159561.347614 -1291640.534815 -2635240.434488
```

-631115.625261 -1608023.341005 -2476168.274113
-475056.350955 -1172928.046443 -3001354.928622
36007.173802 -1379798.256090 -1967379.084440
-32317.979498 -1009700.892591 -2371346.297022
485355.719668 -1128966.486270 -3351319.656256
-178255.906628 -1346153.972518 -3271479.869613
108732.816064 -1458651.868872 -3444874.752530
-120567.260526 -1803335.612331 -2762302.042006