实验报告

院 系 数据科学与计算机学院　班 级 17　学 号 17341097　姓 名 廖永滨

【实验题目】数据表示实验

【实验目的】掌握结构数据的保存和读取的方法。

【实验说明】

- 把源程序和可执行文件放在相应的**上交源码**目录中。
- **截屏**用按键(Ctrl+Alt+PrintScreen)截取当前窗口
- **把每段具有独立功能的代码单独写入一个函数有助于编码和调试。（新增）**

【参考资料】

- **C 语言函数分类：**http://msdn.microsoft.com/zh-cn/library/2aza74he(v=vs.110).aspx
- **C 语言字符串函数：**http://msdn.microsoft.com/zh-cn/library/f0151s4x(v=vs.110).aspx
- **C 语言程序设计：**http://www.runoob.com/cprogramming/　**（新增）**

【实验环境】

- Windows + VS 2012　http://172.18.187.9/netdisk/default.aspx?vm=17net
- Linux + gcc

【实验内容】

(1)结构数据保存和读出(StructSave.cpp)

- **实验要求：**

  循环输入员工(Person)的信息，每输入一个员工的信息，立即写入文件(Persons.stru)，直到输入的姓名为 exit 时跳出循环。然后，读出该文件，显示每个 Person 的信息。

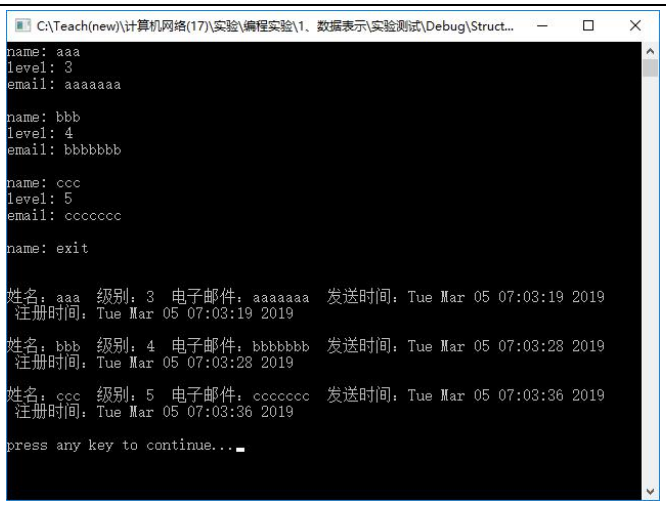
  Person 的信息表示：

```
struct Person {
    char username[USER_NAME_LEN];      // 员工名
    int level;                         // 工资级别
    char email[EMAIL_LEN];             // email 地址
    DWORD sendtime;                    // 发送时间
    time_t regtime;                    // 注册时间
};
```

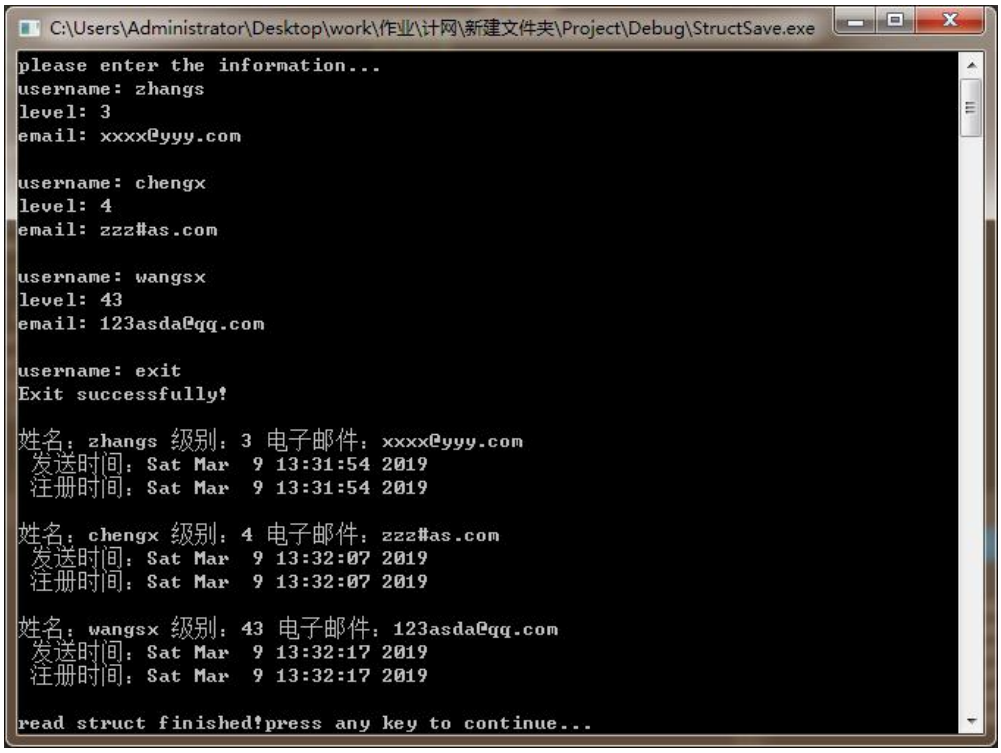
- **老师用到的字符串函数和自定义函数(仅作参考)：（新增）**

```
printf(),scanf(),strcpy()      (VS 2017 要求使用 scanf_s,strcpy_s)
int inputOnePerson(Person *personSent){...}
```

- **参考运行结果：（新增）**



- **截屏运行结果：**



- **源代码：**

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

#define BUF_LEN 100
#define USER_NAME_LEN 20
#define EMAIL_LEN 80
#define TIME_BUF_LEN 30

struct Person {
```

```
    char username[USER_NAME_LEN];       // 员工名
    int level;                          // 工资级别
    char email[EMAIL_LEN];              // email 地址
    time_t sendtime;                     // 发送时间
    time_t regtime;                     // 注册时间
};

int inputOnePerson(Person *personSent) {    // 输入用户信息
    char pts[TIME_BUF_LEN];                 /* pointer to time string          */
    time_t   now;
    (void)time(&now);                                       // 取得系统时间
    ctime_s(pts, TIME_BUF_LEN, &now);                        // 把时间转换为字符串
    //printf(pts);
    char inputBuf[100];
    int inputNumber;
    printf("username: ");
    scanf_s("%s", inputBuf, USER_NAME_LEN);
    if (inputBuf[0] == 'e'&&inputBuf[1] == 'x'&&inputBuf[2] == 'i') {
        if (inputBuf[3] == 't'&&inputBuf[4] == '\0') {
            printf("Exit successfully!\n\n");
            return 0;
        }
    }

    strcpy_s(personSent->username, inputBuf);
    printf("level: ");
    scanf_s("%d", &inputNumber, sizeof(int));
    personSent->level = inputNumber;
    printf("email: ");
    scanf_s("%s", inputBuf, EMAIL_LEN);
    printf("\n");
    strcpy_s(personSent->email, inputBuf);
    personSent->sendtime = now;
    personSent->regtime = now;
    return 1;
}

int main() {
    FILE * pFile;
    errno_t err;
    if ((err = fopen_s(&pFile, "Persons.stru", "wb")) != 0) {
        printf("cant open the file\n");
        getchar();
        exit(0);
    }
    printf("please enter the information...\n");
```

```
    struct Person personSent;
    while (inputOnePerson(&personSent)) {
        if (fwrite(&personSent, sizeof(Person), 1, pFile) != 1)
            printf("file write error\n");
    }
    fclose(pFile);

    FILE * fp;
    Person perBuf;
    char pts1[TIME_BUF_LEN];
    char pts2[TIME_BUF_LEN];
    if ((err = fopen_s(&fp, "Persons.stru", "rb")) != 0) {
        getchar();
        printf("can't open the file");
        exit(0);
    }
    while (fread(&perBuf, sizeof(Person), 1, fp) == 1) {
        ctime_s(pts1, TIME_BUF_LEN, &perBuf.regtime);
        ctime_s(pts2, TIME_BUF_LEN, &perBuf.sendtime);
        printf("姓名：%s 级别：%d 电子邮件：%s\n 发送时间：%s 注册时间：%s\n",
            perBuf.username, perBuf.level, perBuf.email, pts2, pts1);
    }
    printf("read struct finished!");
    printf("press any key to continue...");
    getchar();
    getchar();

}
```

(2) 多文件合并保存和读出(FilePack.cpp)

- **实验要求：**

  循环输入多个文件名（不超过 200MB，可以自己确定），每输入一个，就把该文件的文件名（最多 300 字节）、文件大小(long)和文件内容写入文件 FileSet.pak 中，输入文件名为 exit 时跳出循环。然后，读 FileSet.pak，每读出一个文件就把它保存起来，有重名文件存在时文件名加上序号（从 2 开始）。

  * 合并时可以先取得文件大小，然后边读边写。

- **老师用到的字符串函数和自定义函数(仅作参考)：（新增）**

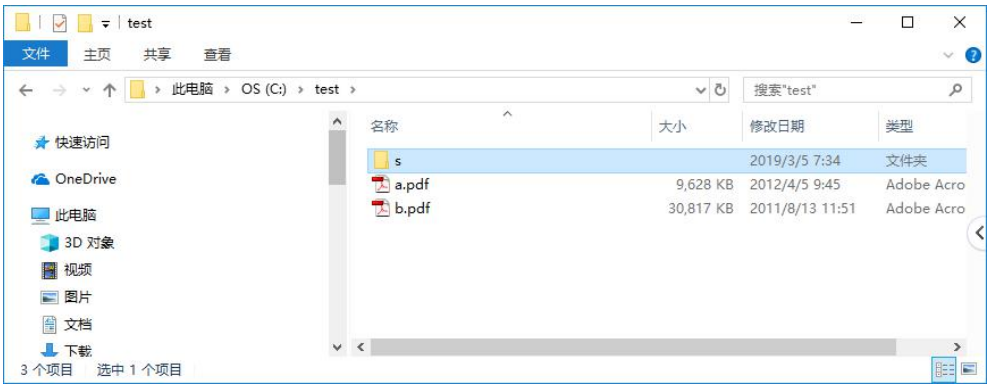
  strcpy()，scanf()，printf()
  sprintf()—用于多个字符串和整数合并成一个字符串
  strrchr()—反向查找字符

```
struct FileStruct {
```

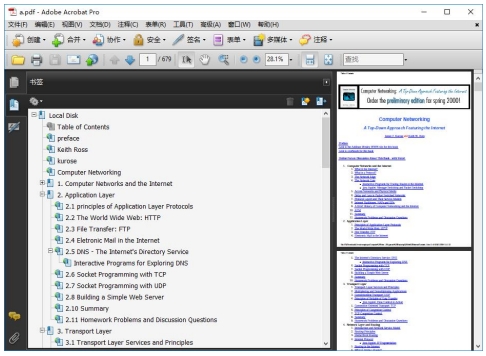
```
    char fileName[300];
     __int64 fileSize;

};
__int64 getFileSize(char * fileName){...}
char * getFileName(char *pathName){...}
int packFile(char *srcFileName,FILE * destFile){...}
// 拷贝filePathName中前面长度为len的字符串到fileFullName
int mystrcpy(char * fileFullName,int len,char * filePathName){...}
void getUniqueName(char *newFileName,char *filePathName){...}
int unpackFile(FILE *srcFile,char *Path){...}
```

- **参考运行结果：（新增）**

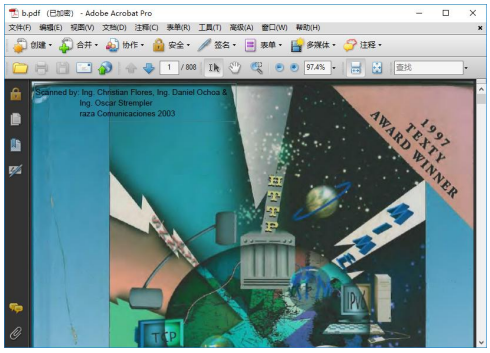
c:\test 下的两个文件：a.pdf,b.pdf，c:\test\s 为空文件夹。



打开 a.pdf 和 b.pdf

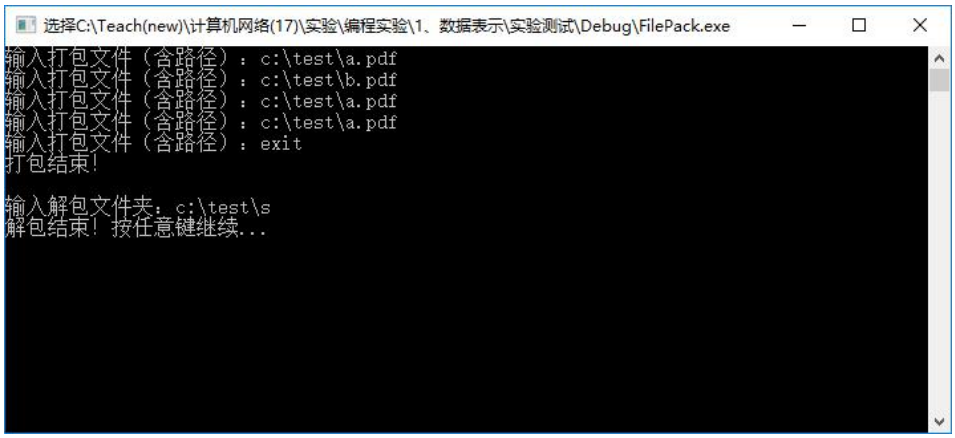


a.pdf

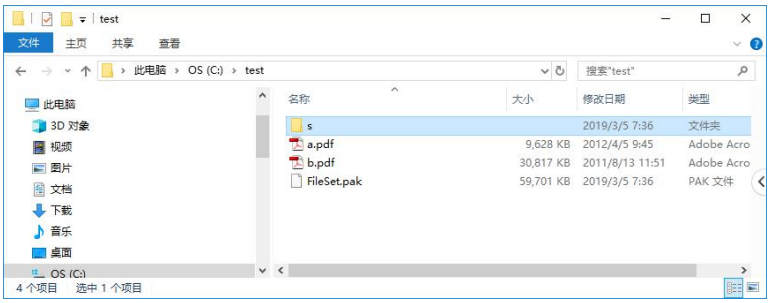


b.pdf

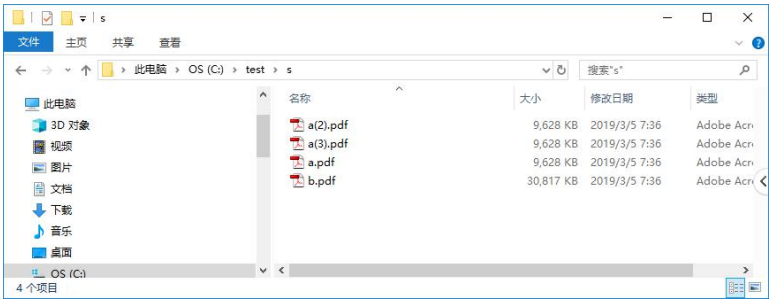
运行程序：



c:\test 的内容变为：



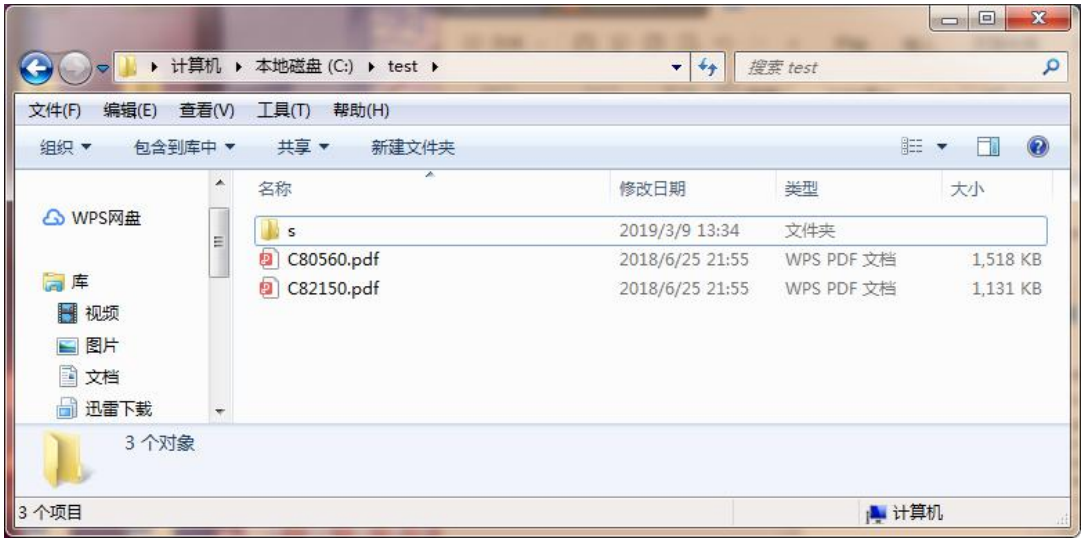
c:\test\s 的内容变为：

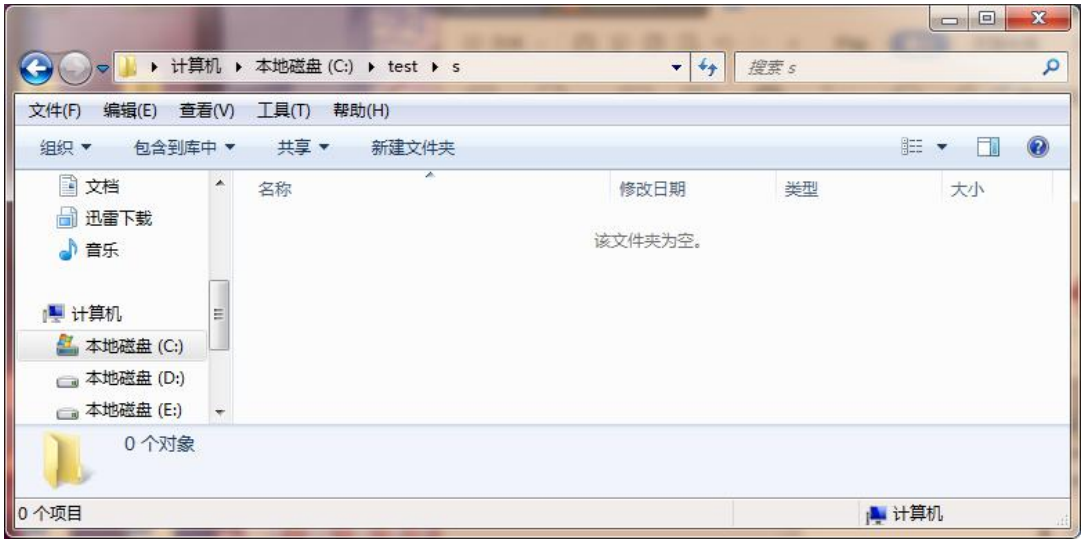


c:\test\s 下的四个文件均可打开，显示结果同上，字节数与源文件相同。

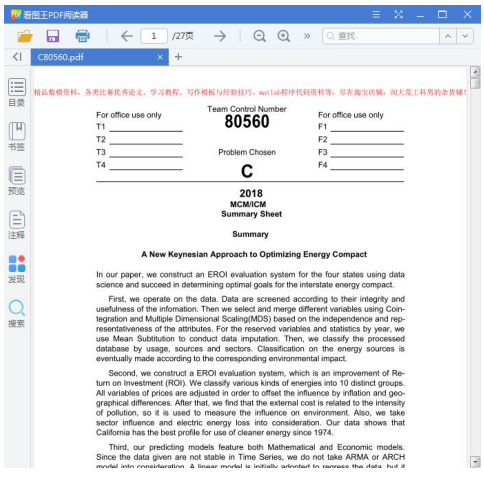
- **截屏运行结果：**

c:\test 下的两个文件：a.pdf,b.pdf，c:\test\s 为空文件夹。





打开 C80560.pdf 和 C82150.pdf



For office use only
Team Control Number
80560
For office use only
Problem Chosen
C
2018
MCM/ICM
Summary Sheet

Summary

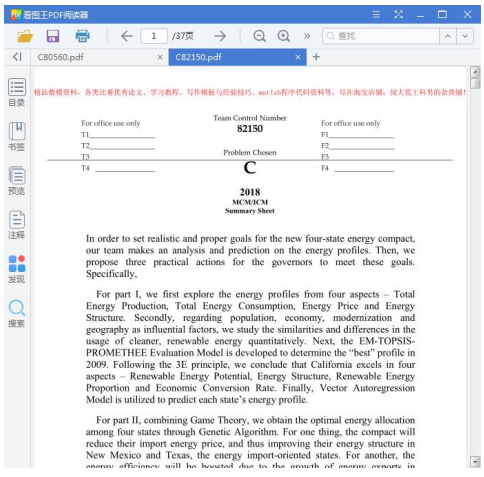
A New Keynesian Approach to Optimizing Energy Compact

In our paper, we construct an EROI evaluation system for the four states using data science and succeed in determining optimal goals for the interstate energy compact.

First, we operate on the data. Data are screened according to their integrity and usefulness of the infomation. Then we select and merge different variables using Cointegration and Multiple Dimensional Scaling(MDS) based on the independence and representativeness of the attributes. For the reserved variables and statistics by year, we use Mean Subtitution to conduct data imputation. Then, we classify the processed database by usage, sources and sectors. Classification on the energy sources is eventually made according to the corresponding environmental impact.

Second, we construct a EROI evaluation system, which is an improvement of Return on Investment (ROI). We classify various kinds of energies into 10 distinct groups. All variables of prices are adjusted in order to offset the influence by inflation and geographical differences. After that, we find that the external cost is related to the intensity of pollution, so it is used to measure the influence on environment. Also, we take sector influence and electric energy loss into consideration. Our data shows that California has the best profile for use of cleaner energy since 1974.

Third, our predicting models feature both Mathematical and Economic models. Since the data given are not stable in Time Series, we do not take ARMA or ARCH



For office use only
Team Control Number
82150
For office use only
Problem Chosen
C
2018
MCM/ICM
Summary Sheet

In order to set realistic and proper goals for the new four-state energy compact, our team makes an analysis and prediction on the energy profiles. Then, we propose three practical actions for the governors to meet these goals. Specifically,

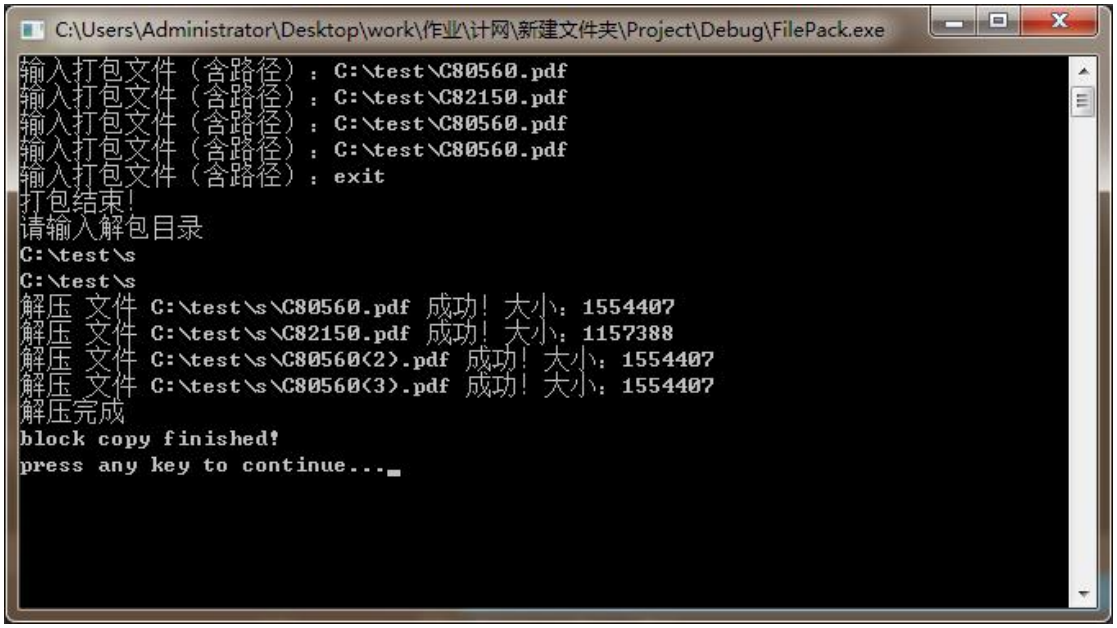
For part I, we first explore the energy profiles from four aspects – Total Energy Production, Total Energy Consumption, Energy Price and Energy Structure. Secondly, regarding population, economy, modernization and geography as influential factors, we study the similarities and differences in the usage of cleaner, renewable energy quantitatively. Next, the EM-TOPSIS-PROMETHEE Evaluation Model is developed to determine the "best" profile in 2009. Following the 3E principle, we conclude that California excels in four aspects – Renewable Energy Potential, Energy Structure, Renewable Energy Proportion and Economic Conversion Rate. Finally, Vector Autoregression Model is utilized to predict each state's energy profile.

For part II, combining Game Theory, we obtain the optimal energy allocation among four states through Genetic Algorithm. For one thing, the compact will reduce their import energy price, and thus improving their energy structure in New Mexico and Texas, the energy import-oriented states. For another, the

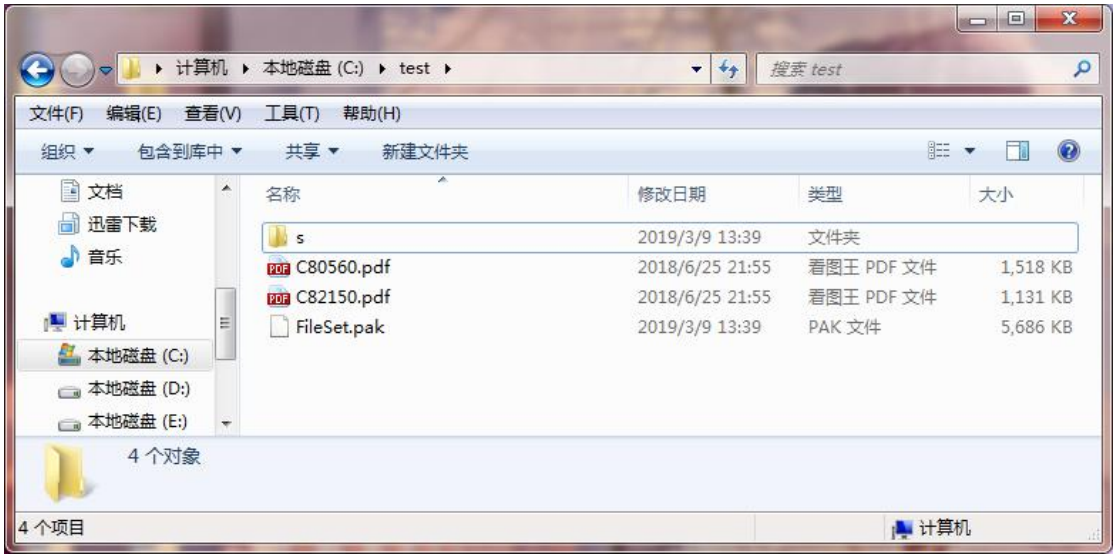
C80560.pdf

C82150.pdf

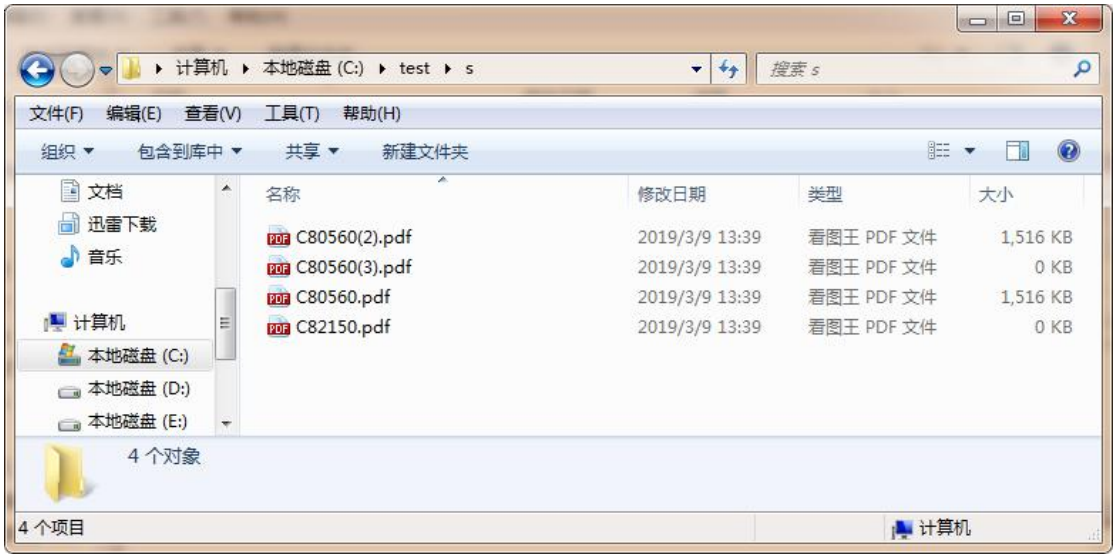
运行程序：



c:\test 的内容变为：



c:\test\s 的内容变为：



c:\test\s 下的四个文件均可打开，显示结果同上，字节数与源文件相同。

- **源代码：**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <io.h>
#define FILE_NAME_LEN 300
#define FILE_MAX_SIZE 1024

char* getFileName(char *pathName) {
    if (strrchr(pathName, '\\') != NULL) {
        pathName = strrchr(pathName, '\\');
        pathName = pathName + 1;
    }
    return pathName;
}

void getUniqueName(char *newFileName, char *filePathName) {
    int i = 2;
    while (!_access(newFileName, 0)) {//判断是否存在
        int flag = 0;
        int j = 0;
        for (j = 0; filePathName[j] != '\0'; j++) {
            if (filePathName[j] == '.') {
                newFileName[j] = '(';
                if (i > 9) {
                    newFileName[j + 1] = (i/10) + 48;
                    newFileName[j + 2] = (i%10) + 48;
                    newFileName[j + 3] = ')';
                    flag = 4;
```

```
                }else {
                    newFileName[j + 1] = i + 48;
                    newFileName[j + 2] = ')';
                    flag = 3;
                }

            }
            newFileName[j+flag] = filePathName[j];
        }
        newFileName[j + flag] = '\0';
        i++;
    }
}

struct FileStruct {
    char fileName[FILE_NAME_LEN];
    __int64 fileSize;
};

int packFile(char *srcFileName, FILE * destFile) {
    char buf[FILE_MAX_SIZE];//文件内容
    FileStruct srcfname_st;
    FILE *srcFile;
    if (srcFileName[0] == 'e'&&srcFileName[1] == 'x'&&srcFileName[2] == 'i') {
        if (srcFileName[3] == 't'&&srcFileName[4] == '\0') {
            printf("打包结束！\n");
            return 1;
        }
    }
    errno_t err = fopen_s(&srcFile, srcFileName, "rb");//打开要读取的二进制文件
    if (err != 0) {
        printf("打包文件未找到！\n");
        printf("按任意键继续...");
        getchar();
        getchar();
        exit(1); //中止程序
    }
    fseek(srcFile, 0, SEEK_END);   ///将文件指针移动文件结尾
    srcfname_st.fileSize = ftell(srcFile); ///求出当前文件指针距离文件开始的字节
    fseek(srcFile, 0, SEEK_SET);   ///将文件指针移动文件头部
    strcpy_s(srcfname_st.fileName, getFileName(srcFileName));

    if (fwrite(&srcfname_st, sizeof(FileStruct), 1, destFile) != 1)
        printf("file write error\n");
    int len = 0;
    while ((len = fread(buf, 1, FILE_MAX_SIZE, srcFile)) >= FILE_MAX_SIZE) {
```

```
            fwrite(buf, 1, FILE_MAX_SIZE, destFile);
        }
        fwrite(buf, 1, len, destFile);
        fclose(srcFile);
        return 0;
    }

    int unpackFile(FILE *srcFile, char *Path) {
        //fseek(srcFile, 0, SEEK_SET);   ///将文件指针移动文件头部
        FILE *destFile;
        char destFilename[FILE_NAME_LEN];
        char uniqueName[FILE_NAME_LEN];
        char buf[FILE_MAX_SIZE];
        FileStruct destFilename_st;
        errno_t err;
        printf("%s\n", Path);
        while (1) {
            if (fread(&destFilename_st, 1, sizeof(FileStruct), srcFile) == 0) {
                printf("解压完成\n");
                break;
            }
            sprintf_s(destFilename, "%s\\%s", Path, destFilename_st.fileName);

            strcpy_s(uniqueName, destFilename);
            getUniqueName(uniqueName, destFilename);

            if (err = fopen_s(&destFile, uniqueName, "wb") != 0) {
                printf("解包%s 文件失败,目录错误或编号过长（至多 99）\n", uniqueName);
                getchar();
                getchar();
                exit(0);
            }

            printf("解压 文件 %s 成功! 大小：%lld \n", uniqueName,
destFilename_st.fileSize);

            __int64 len = destFilename_st.fileSize;
            while (len >= FILE_MAX_SIZE) {   //可以一次读取
                fread(buf, 1, FILE_MAX_SIZE, srcFile);
                fwrite(buf, 1, FILE_MAX_SIZE, destFile);
                len -= FILE_MAX_SIZE;
            }
            fread(buf, 1, len, srcFile);
            fwrite(buf, 1, len, destFile);
        }
        return 1;
```

```
}

int main() {
    FILE *destFile;
    char destFilename[FILE_NAME_LEN] = "C:\\test\\FileSet.pak";  //按需求可修改
    char srcFileName[FILE_NAME_LEN]; //用于源文件路径+名字
    errno_t err;
    //如果要只需要解包，请注释从这里 116 到 127 之间的所有代码###################
    if (err = fopen_s(&destFile, destFilename, "wb") != 0) {
        printf("error\n");
    }
    while (1) {
        printf("输入打包文件（含路径）：");
        scanf_s("%s", srcFileName, FILE_NAME_LEN);
        if (packFile(srcFileName, destFile))
            break;
    }
    fclose(destFile);
    //如果要只需要解包，请注释从这里 116 到 127 之间的所有代码###################
    if (err = fopen_s(&destFile, destFilename, "rb") != 0) {
        printf("error\n");
    }
    printf("请输入解包目录\n");
    scanf_s("%s", srcFileName, FILE_NAME_LEN);
    unpackFile(destFile, srcFileName);
    fclose(destFile);

    printf("block copy finished!\r\n");
    printf("press any key to continue...");
    getchar();
    getchar();
    return 0;
}
```

- **与同学互测并截屏运行结果：**

  把打包的文件给同学，看他是否可以取出其中文件，同样测试是否可以读出并取出同学的打包文件。

  * 注意结构要相同

【完成情况】

是否完成以下步骤？（√完成 ×未做）

(1) [√] (2) [√]

是否与同学进行了互测？[√](√ ×)

互测同学的学号姓名：17341099 廖泽祥

【实验体会】

主要问题与解决方案：

1. VS2017 相较 DEV C++较为严格，体现在大量的语法警告与内存检查。

比如在文件读写上，如果创建一个较大的字符串空间，VS 会直接报错，解决方案就在老师提供的 BlockCopy.cpp 里，分为多次写入。这里又有一个问题，就是写入的总长度需要先记录下来，写入次数需要精密计算，多出或少于文件长度的读写都将破坏文件内容。

又如输入，字符串输入拼接，copy 上，VS 需要使用_s 的函数，且表达格式也略有不同。

2. 数据类型较多，需要严格同一。

在与同学互测的过程中，我发现，只有用统一的解包与打包格式，才能成功互测，主要原因在于同学用的是 DEV，我用的是 VS2017，他在打包时一开始用的是 long 作为文件长度，而我采用__int64 作为打包文件长度个数，出现了冲突。后面统一了就好了很多。

3. 文件读与写要严格区分，防止文件丢失或读写失败。

在开始的实验中，我使用 fread 和 fwrite 没有严格按要求来，一个文件打开了没有及时关闭或者文件位置指针没有及时回到文件头，导致出现了打包不报错，但是文件大小只有 0KB 的 bug 存在。后来，我 fopen 和 fclose 严格按要求来，不用的文件立即 fclose，用时再 fopen，就解决了。

4. 部分函数编写较麻烦，许多函数的调用需要自己上网查找。

有一部分函数老师已经给了出来，比如字符串的查找问题等等。但是，有些函数要自己写，比如重复命名时的解决函数，一方面需要用 access 函数查看是否重复，另一方面在重复后，需要重命名，这时修改字符串的难度略高，要在名字后面加（？），由于文件存在后缀名，所以需要插入在之间，这不是很好操作。

总的体会与收获的经验：

1. 熟悉了文件的读写以及路径相关的问题

在实验中，读写是最大一个部分，虽然有时候很麻烦，会遇到很多坑人的 bug 的，但是写好后还是蛮有成就感的。但事实上，还有一个大问题就是路径问题。程序对于相对路径与绝对路径都是能接受，需要按自己需求来，代码中的\是特殊字符，所以需要再加个\去除意义。

2. 熟悉了 VS2017 的操作

这是我第一次使用 VS2017，之前我都是用的 DEV 的，没想到，通过一次实验，我对 VS2017 的熟练度就已经和 DEV 差不多了，甚至觉得 VS 用起来比 DEV 更舒服，因为代码检测机制更完善，自带的函数库也比较全。

3. 模块化设计思想更进一步

虽然模块化思想在大一以及大二上学期就已经很有体会了，但这次实验，分函数，分模块的设计

实验报告

理念依旧发光发热，一方面，他让代码更规范更清晰更好理解，另一方面，他让 bug 变得更好处理。

【交实验报告】

每位同学单独完成本实验内容并填写实验报告。

交作业地点：http://172.18.187.11/netdisk/default.aspx?vm=17net

编程实验

截止日期：2019 年 3 月 9 日 23：00（周六）。

上传文件：学号_姓名_数据表示实验报告.doc

学号_姓名_数据表示实验源码.rar（源程序和可执行程序）