# 北京交通大学
# 计算机实验报告

# 实验报告（lab3）

年　　级：大学一年级
学　　号：20281274
姓　　名：杜海玮
专　　业：计算机科学
任课老师：邹琪

二零二一年四月

# 目　　录

# 1）程序实现

## A）如何判断用户输入的数据记录条数合法

### 1.1 数据记录条数检查函数

```
int check(char* c) {//用户输入了数据记录条数值或文件名
    int flag = 0;
    for (int i = 0; c[i] != '\0'; i++) {
        if (!isdigit(c[i])) {
            flag = 1;//如果出现非数字字符，则所检查的char数组为文件名
        }
    }
    return flag;//flag为1表示为不只有数字的字符串，为文件名。flag为0表示只有数字的字符串，为文件条数。
}
//.....................................................
```

图 1-1 数据记录条数检查函数

输入字符串指针，利用 ctype 库函数中的 isdigit 函数检查是否是纯数字函数，按检查结果返回 1（被检查字符串指针不只有数字，应为文件名）或 0（被检查字符串指针只有数字，应为指定数据条数）。在上级函数中直接检查若为字符串情况时是否有且只有字符’r’，若存在则生成随机数据条数。

## B）生成数据记录文件函数

```
void CreateFile() {
    char FullPath[1000];
    int acc = _access(Stat.filesavepath, 0);
    if (acc == 0) {//目录存在
        _chdir(Stat.filesavepath);
        strcpy_s(FullPath, Stat.filesavepath);
        strcat_s(FullPath, Stat.filename);
        fopen_s(&fp, Stat.filename, "w");
        if (fp == NULL) {
            printf("文件打开失败");
        }
        fprintf(fp, "%d\n", Stat.number);
        for (int i = 0; i < Stat.number; i++) {
            int j = 0;
            fprintf(fp, "%d,%d,%d\n", Data[i][j], Data[i][j + 1], Data[i][j + 2]);
        }

        fclose(fp);
    }
    if (acc == -1) {//目录不存在
        CreateDir(Stat.filesavepath);
        _chdir(Stat.filesavepath);
        fopen_s(&fp, Stat.filename, "w");
        if (fp == NULL) {
            printf("文件打开失败");
        }
        fprintf(fp, "%d\n", Stat.number);
        for (int i = 0; i < Stat.number; i++) {
            int j = 0;
            fprintf(fp, "%d,%d,%d\n", Data[i][j], Data[i][j + 1], Data[i][j + 2]);
        }
        fclose(fp);
    }
```

图 1-2 生成数据记录文件函数

其应该输入配置结构体信息以读取其中的信息以生成数据记录文件。其返回值应该为生成结果的成功或失败以进行下一步对应操作。在本次实验中我使用了extern+全局变量的手段，所以图中该函数没有输入与返回值，其输入与反馈用全局变量实现。

## C）获取一个指定范围内的随机整数的函数

```
#define random(m,n) (rand()%(n-m+1)+m)
```

图 1-3 获取一个指定范围内的随机整数的函数

输入上 m 是取值下限，n 是取值上限。通过已有的随机数函数对上下限的差（取值范围）做取余操作首先生成一个范围内的随机整数，然后加上下限，即生成了指定范围内的随机整数。

## D）程序健壮性统计

本程序中用于保障健壮性的子函数包括输入检查函数，保障健壮性的代码包括各输入函数中的非法输入-再次输入循环与对应的提示文字、文件创建函数中为了对不同格式路径的支持而建立的多级多格式形式目录创建函数等。经统计，保障健壮性的代码约占总代码数量的 73%（455/624 行）。

我认为足够的健壮性或鲁棒性是写出来的程序或代码拥有足够长生存周期的保证，也是代码编写者实力的体现。健壮性支持代码可以显著拓展程序的输入输出范围与用户使用体验，相比之下核心功能则显得体量没有考虑大量情况的健壮性代码大。

## E）结构体变量的作用

实验中的结构体变量分别装载了随机数据数值的上下限，随机生成数据条数数量的上下限，指定数据条数数量，文件名及存储路径。这样，这个结构体其实就是一个文件的目录，它上面记载有生成文件所需的所有信息，相较于 lab2 时期代码的“单打独斗”如果需要临时调用则直接用结构体变量回方便直观许多。

## F）子函数

在 lab2 中我已经将二级功能部分封装为子函数（见 lab2 实验报告），本次实验中我将合法文件路径检验、文件路径分离以及多级目录展开函数封装为子函数。

合法文件路径检验函数将首先检测文件路径中是否含有非法字符，之后再检查文件后四位是否为.txt 或.dat 的字符组合以符合要求。它的输入值就是一个字符串指针（指向待测字符串），返回值则反映检测结果，1 代表通过检测，0 代表不通过检测。

```
int CheckFilePath(char* input) {
    for (int i = 0; i < strlen(input); i++) {
        if (/*input[i] == ':' || */input[i] == '*' || input[i] == '?' || input[i] == '\"' || input[i] == '<' || input[i] == '>' || input[i] == '|') {
            return 0;//非法字符排查，出现非法字符返回0
        }
    }
    int Last4Digit = strlen(input) - 4;
    if (input[Last4Digit] == '.') {
        if (input[Last4Digit + 1] == 't' || input[Last4Digit + 1] == 'd') {
            if (input[Last4Digit + 2] == 'x' || input[Last4Digit + 2] == 'a') {
                if (input[Last4Digit + 3] == 't') {
                    return 1;
                }
                else return 0;
            }
            else return 0;
        }
        else return 0;
    }
    else return 0;//检查末尾是否是.txt或.dat格式，如果是则返回1，不是则返回0
}
```

图 1-4 合法文件路径检验函数

文件路径分离函数使用 strrchr 函数将分隔目录的/或\检测并读取位置，通过限界复制函数 strncpy 将目录分割，最终将文件名与文件路径分别输入对应的结构体变量中。实现难点是对所有情况的应对，出现了数个分支处理路径。该函数输入目录字符串指针，直接处理全局结构体变量。

```
void FillPath(char* a) {
    char* name;
    char* name2;
    char def[20] = { "../OutputData/" };
    name = (char*)malloc(sizeof(char) * 100);
    name2 = (char*)malloc(sizeof(char) * 100);
    if (a[1] == ':') {
        name = strrchr(a, '\\') + 1;
        strcpy_s(Stat.filename, name);
        strncpy_s(Stat.filesavepath, sizeof(Stat.filesavepath), a, strlen(a) - strlen(name));
        PathForm = 1;//用户使用绝对路径形式
    }
    else {
        name = strrchr(a, '\\');
        name2 = strrchr(a, '/');
        if (name == NULL && name2 == NULL) {
            //name = strrchr(a, '/') + 1;
            strcpy_s(Stat.filename, a);
            strcpy_s(Stat.filesavepath, def);
            PathForm = 0;//用户使用最简相对路径形式
        }
        else if (name2 == NULL) {
            name = strrchr(a, '\\') + 1;
            strcpy_s(Stat.filename, name);
            strncpy_s(Stat.filesavepath, sizeof(Stat.filesavepath), a, strlen(a) - strlen(name));
            PathForm = 2;//用户使用只含有\的相对路径形式
        }
```

```
        PathForm = 0;//用户使用最简相对路径形式
    }
    else if (name2 == NULL) {
        name = strrchr(a, '\\') + 1;
        strcpy_s(Stat.filename, name);
        strncpy_s(Stat.filesavepath, sizeof(Stat.filesavepath), a, strlen(a) - strlen(name));
        PathForm = 2;//用户使用只含有\的相对路径形式
    }
    else if(name == NULL){
        name = strrchr(a, '/') + 1;
        strcpy_s(Stat.filename, name);
        strncpy_s(Stat.filesavepath, sizeof(Stat.filesavepath), a, strlen(a) - strlen(name));
        PathForm = 2;//用户使用只含有/的相对路径形式
    }
    else {
        int pos1 = 0, pos2 = 0;//pos1代表\的位置，pos2代表/的位置
        for (int i = 0; i < strlen(a); i++) {
            if (a[i] == '\\') {
                pos1 = i;
            }
            if (a[i] == '/') {
                pos2 = i;
            }
        }

        if (pos1 > pos2) {
            name = strrchr(a, '\\') + 1;
            strcpy_s(Stat.filename, name);
            strncpy_s(Stat.filesavepath, sizeof(Stat.filesavepath), a, strlen(a) - strlen(name));
            PathForm = 2;//用户使用只含有\的相对路径形式
        }
        if (pos1 < pos2) {
            name = strrchr(a, '/') + 1;
            strcpy_s(Stat.filename, name);
            strncpy_s(Stat.filesavepath, sizeof(Stat.filesavepath), a, strlen(a) - strlen(name));
            PathForm = 2;//用户使用只含有/的相对路径形式
        }
    }
}
```

图 1-5、6、7 文件路径分离函数

目录多级展开创建函数用于创建多级目录，难点是用不同分支区分用户给予的路径信息，当确认分隔后将目录依次创建即可（循环结构）。该函数接收之前的路径分离函数的结果，输入目录字符串指针，输出创建结果。

```
*****************************************/
int CreateDir(char* path)
{
    char str[1000];
    char* c = path;
    int lenth = strlen(path), flag = 0;
    int i, count;             //count记录/或者\的位置
    if (path == NULL) return 0;
    if (PathForm == 0) return 1;
    else
    {
        if (PathForm == 1)//处理绝对路径
        {
            strncpy_s(str, sizeof(str), path, 2);
            if (_chdir(str))//返回值为0，符合if条件，即磁盘不存在
            {
                printf("磁盘不存在\n");
                return 0;
            }
            else
            {
                for (i = 0, count = 0; i < lenth; i++)
                {
                    if (path[i] == '/' || path[i] == '\\')
                    {
                        count = i;
                        strncpy_s(str, sizeof(str), path, count);//将/或\之前的路径复制给str
                        if (_chdir(str))//如果不存在则创建目录
                        {
                            mkdir(str);

                        }
                    }
```

```
    else//处理相对路径
    {
        for (i = 0, count = 0; i < lenth; i++)//先处理..的情况，将其处理完成后再处理别的
        {
            if (path[i] == '.' && path[i + 1] == '.' && path[i + 2] != '.')
            {
                _chdir("..");
                c = path + i + 2;
                flag = 1;
            }//找到最右端的“../”
        }
        if (flag) c++;
        strcpy_s(str, sizeof(str), c);
        lenth = strlen(str);
        for (i = 0; i < lenth; i++)
        {
            if (str[i] == '/' ) {
                str[i] = '\0';
                mkdir(str);
                str[i] = '/';
            }
            if ( str[i] == '\\') {
                str[i] = '\0';
                mkdir(str);
                str[i] = '\\';
            }
        }
        _chdir(str);
    }
}
printf("目录创建成功\n");
return 1;
```

图 1-8、9 多级目录展开创建函数

## 2）文件路径相关问题

A）当用户输入文件名参数时输入的相对路径或绝对路径中包含有未创建的文件夹，程序是否能够正常运行？

程序不可以正常运行，将返回文件指针找不到目标的提示。我的程序中通过配置文件夹创建函数，当_access 函数返回文件夹不存在的结果时将调用文件夹多级展开创建函数进行文件夹创建，解决了这个问题。（文件夹多级展开创建函数见上）

B）如果程序强制要求文件名参数只能允许用户输入文件名参数，不能输入任何含有路径的文件名，那么在对文件名参数合法性校验的程序逻辑部分，需要做怎样的改动？

如图，参考 windows 系统中的提示，则不能含有具有文件路径提示性质的字符，例如/\:。只需要再合法性校验的逻辑中用逻辑或关系添加这些字符串即可。

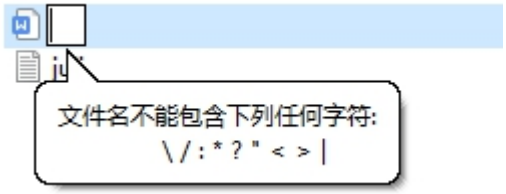


图 2-1 Windows 对于合法文件名的提示

## 3）实验总结

### A）实验总结

通过本次实验我深刻理解了 cpp 文件与.h 头文件之间的关系。通过一组 cpp 文件与头文件的搭配可以灵活的实现代码复用与程序的模块化应用。同时我还学习并大量应用了限界复制函数处理字符串，应用目录创建函数与循环搭建多级目录展开函数等等。

实验刚开始我便遇到了麻烦，在实验 2 中我使用了大量全局变量方便处理数据，在实验 3 中我为了避免全局变量重复声明的问题新建了一个头文件并应用 extern 变量处理这些全局变量，避免了头文件之间的冲突。在实验中，字符串处理涉及许多字符位置的确认，我使用限界复制函数重新创建新字符串的方式来方便的调整需要处理的字符串。在实验中我学习了函数的封

装，认识到如果将输入限定为结构体变量与需求的变量则可以方便的调用函数，但遗憾的是实验中我还是部分保留了实验 2 中的全局变量（以外部变量的形式实现），争取再实验 4 中解决这个问题。

在具体实现方面，我认识了文件路径的组成，实现了多级展开文件路径创建函数的构建。认识了文件的写入，_assess 函数,_chdir,_mkdir 函数等等，收获颇丰。

## B) 对小组其他成员的文档审查报告

陈明强同学：他的函数设置了结构体接口，相对于延续 lab2 使用全局变量的我他的代码显得更加健壮，更加灵活。

高原同学：他的代码中注释得更加全面，更加易读。他的实验报告在需要放大的地方用了放大到具体代码的截图，值得学习，但是可能格式需要再修缮一下。