北京交通大学
计算机实验报告

实验报告（lab4）

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

二零二一年五月

目　　录

1）程序实现

A）配置文件读取函数

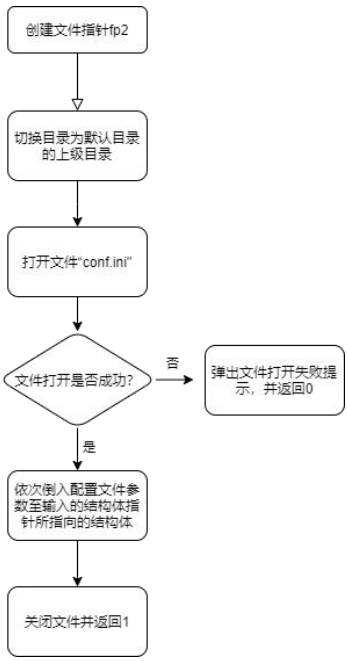
1.1 函数声明

```
/*********************************************
函数名称:
    配置文件读取函数
功能描述:
    读取程序exe文件同名目录下的文本配置文件并写入输入的结构体
函数参数:
    结构体configinof类型
返回值:
    1为输入成功，0为输入失败
模块历史:
    杜海玮于2021年4月13日创建本模块，email:20281274@bjtu.edu.cn
*********************************************/
int ReadConf(configinfo*input) {
```

图 1-1 配置文件读取函数

输入参数为 configinfo 型结构体指针（也就是存放配置文件的结构体），返回值 1 代表输入成功，返回值 0 代表输入失败。

1.2 函数流程图



B）文件分类型输出新增函数：

```
**********************************************/
void WriteFile(DataItem*a) {
    if (Stat.FileType == 't') {
        fprintf(fp, "%d\n", Stat.number);
        for (int i = 0; i < Stat.number; i++) {
            fprintf(fp, "%d,%d,%d\n", a[i].item1, a[i].item2, a[i].item3);
        }
    }
    if (Stat.FileType == 'd') {
        fwrite(&Stat.number, 4, 1, fp);
        fwrite(a, sizeof(DataItem), Stat.number, fp);
    }
}
```

图 1-2 文件写入函数

Lab4 中新增加了分类型输出的要求，为此我在生成文件函数中新增了根据用户指定类型将对应后缀名加入至文件名后方的语句。在文件写入函数中增加了用二进制写入的分支函数。

```
char txt[10] = ".txt";
char dat[10] = ".dat";
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);
    if (Stat.FileType == 't') {
        strcat_s(Stat.filename, txt);
    }
    if (Stat.FileType == 'd') {
        strcat_s(Stat.filename, dat);
    }
```

图 1-3 后缀名按用户指定加入

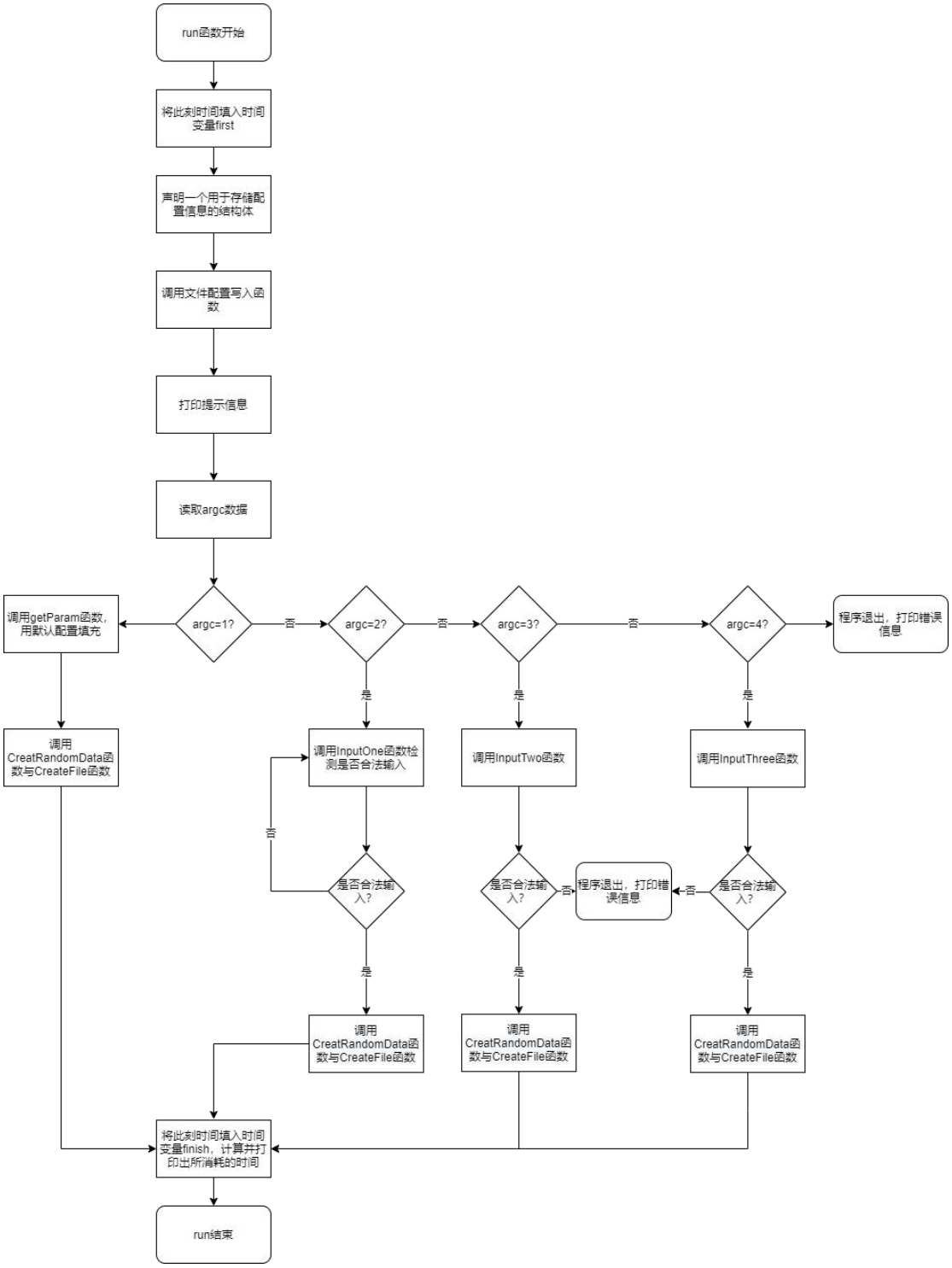
C）新增：计时代码：

加入 time 库，在 run 函数开头将当前时间赋值于初始时间变量，在程序最后将当前时间赋值给最终时间变量，用其之差除以单位时间变量得到总耗时。

```
finish = clock();
double TheTimes;
TheTimes = (double)(finish - start) / CLOCKS_PER_SEC;
printf("总共消耗时间:%lf秒", TheTimes);
```

图 1-4 计时代码

D) 修改后的 run 函数流程图：



2）综述与学习：字节序与两种写入方式的比较

A）字节序

字节序有两种配置模式，Motorola 的 PowerPC 系列 CPU 和 Intel 的 x86 系列 CPU。PowerPC 系列采用 big endian（大端存储）方式存储数据，而 x86 系列则采用 little endian（小端存储）方式存储数据。

Little Endian 是将低序字节存储在起始地址，Big Endian 是将高序字节存储在起始地址。

比如 int a = 0x08070605，在 Little Endian 的情况下存放为：

字节号: 0 1 2 3

数 据: 05 06 07 08

在 Big Endian 的情况下存放为：

字节号: 0 1 2 3

数 据: 08 07 06 05

我的电脑配备是 x86 系列 cpu，则默认使用小端存储模式，我选择保持默认选择，使用小端存储模式保存数据。实际上字节序往往只在不同读取方式（cpu）的机器上体现差别（与操作系统无关，例如计算机网络的应用等）。

B）使用时间函数分析两种储存方式的异同

采用三组数据条目两样共六个样本进行对比，如下表：

参数 \ 数据量	100	1000	10000	100	1000	1000
生成时间/s	0.003	0.003	0.027	0.002	0.003	0.005
生成类型	t	t	t	d	d	d

表 1-1 二进制与十进制写入方式的比较

从表可得，二进制写入方式在全数段的写入速度均快于或等于十进制，而在数据量偏大的情况，二进制写入速度显著快于十进制，快了一个数量级。

3）实验总结

A）实验总结

通过本次实验我实践了用结构体指针作为参数的函数（文件配置读入函数，见上），还首次实验了使用二进制的方式写入文件，编写了独立的二级制读取程序，验证了我输出二进制文件的正确性。

通过比较生成二进制文件的速度与大小，我有如下结论：生成速度方面，因为二进制文件直接写入二进制代码，无需进一步转换，其生成大规模数据（数据条目大于等于 1000）时速度显著快于 txt 文本文件。而在生成的文件大小方面，因为文本文件将 int 转化为对应 ASCII 码的字符（4 字节转化成 1 字节）导致同样数据规模下二进制文件反而比文本文件大。

时间函数方面，我学习了 time 库最粗浅的应用，调用 clock()函数，用系统自带的时间分量计算得到两个过程相隔的时间。

内存操作方面，我首次应用 free 函数，在变量使用完后将其所占用的内存空间释放，完善了代码的健壮性与安全性。

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，word 文档编辑的全面性覆盖不足。

高原同学：缺少 free 函数来释放内存，整合封装上面比我易读。