

南京林业大学

本科课程设计

学 院： 信智院

专 业： 计算机科学与技术

课程名称： 程序设计语言实习

学 号： 2351610105

学生姓名： 方泽宇

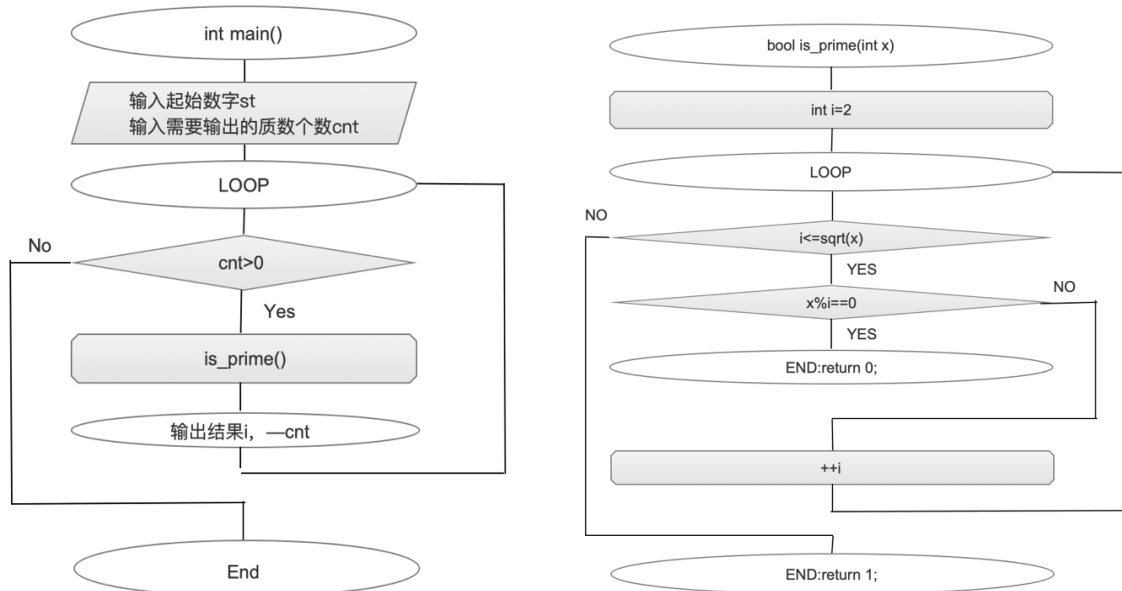
指导教师：

二〇二四年七月

模块一

[题目] 编写一个函数，将大于整数 m 且紧靠 m 的 k 个素数存入数组中，在命令行输入 m 和 k 。如输入 17 和 5，输出 19, 23, 29, 31, 37。

[流程图]



[代码]

```

#include<stdio.h>
bool is_prime(int x)//编写函数 is_prime, 判断一个数字是否是质数
{
    for(int i=2;i<=x/i;++i)
        //把条件写成 i<=x/i 而不是 i*i<=x, 是防止 i*i 爆 int 类型
        //把条件写成 i<=x/i 而不是 i<=sqrt(x), 是为了加快程序的运行速度
        if(x%i==0) return 0;
    return 1;//如果所有的 i 都不是 x 的因子, 那么说明他是一个质数
}

int main()
{
    int st,cnt;
    scanf("%d%d",&st,&cnt);
    for(int i=st+1;cnt;++i)//尝试判断数字 i 是否是质数
    {
        if(is_prime(i))//如果是质数, 打印, 再把 cnt 输出的总字符数-1
        {
            printf("%d ",i);
            --cnt;
        }
    }
    return 0;//程序结束, 别忘了返回 0
  
```

}

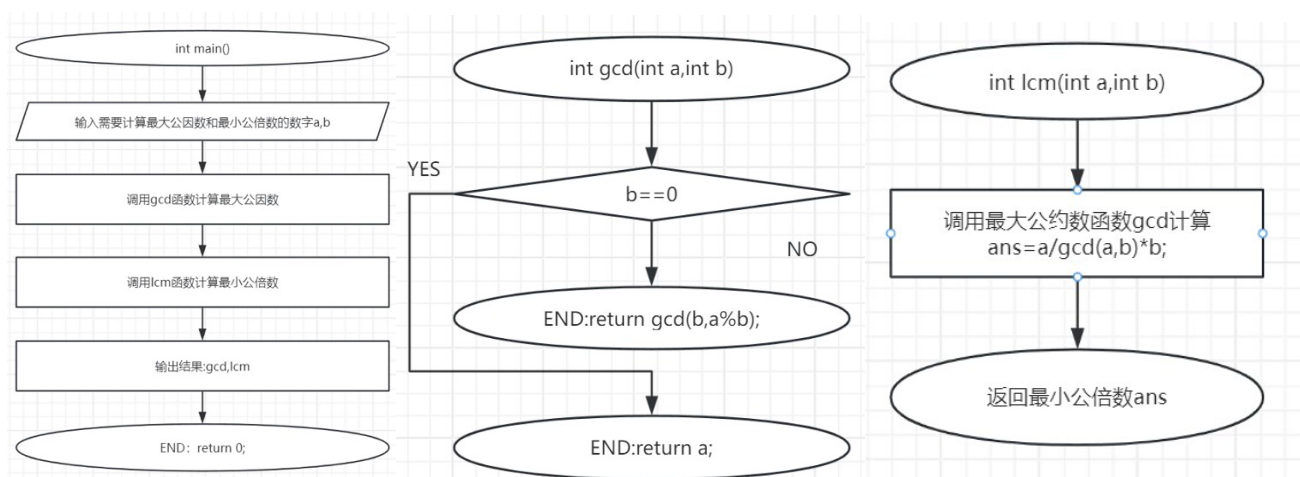
[运行结果]

● fang50253@MacBook-Pro 程序设计语言实习 % cd
 & "/Users/fang50253/Desktop/Files/Document:
 17 5
 19 23 29 31 37 %
 ○ fang50253@MacBook-Pro 程序设计语言实习 % █

[小结]

此题涉及到了条件语句、循环语句、判断质数算法、函数返回等内容。必须注意平均分`aver`的赋值情况。如不能在一开始赋值为0，则结果将不正确。

[题目] 求最大公约数和最小公倍数。输入两个正整数 m 和 n ，求其最大公约数和最小公倍数。

[流程图]**[代码]**

```

#include<stdio.h>
int gcd(int a,int b)
{
    return b?gcd(b,a%b):a;//使用辗转相除法计算最大公因数
}
int lcm(int a,int b)
{
    return a/gcd(a,b)*b;//这里写成 a/gcd(a,b)*b 而不是 a*b/gcd(a,b)是为了防止爆 int
}
int main()
{
    int a,b;
    scanf("%d%d",&a,&b);//读入两个数字，计算最大公因数和最小公倍数
  
```

```
printf("%d %d",gcd(a,b),lcm(a,b)); //调用函数, 输出结果
return 0;
}
```

[运行结果]

```
fang50253@MacBook-Pro:~/Programs/3求最大公约数和最小公倍数 && "/Us
8 12
4 24
fang50253@MacBook-Pro:~/Programs/3求最大公约数和最小公倍数 && "/Us
```

[小结]

此题涉及到了三目运算符条件语句、递归、辗转相除法求最大公因数、函数返回等内容。必须注意递归的终止条件 $b=0$ ，否则递归无法结束。

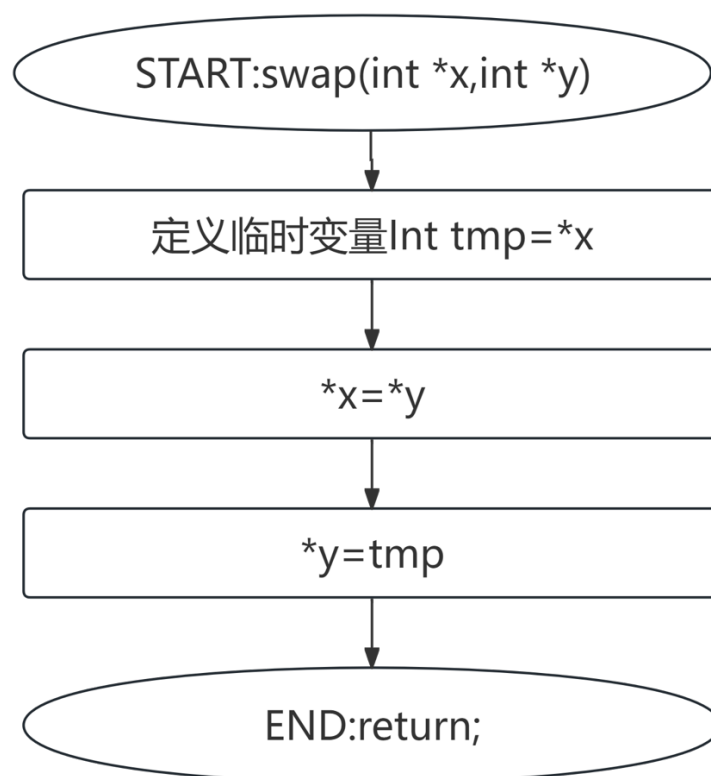
模块二

[题目]

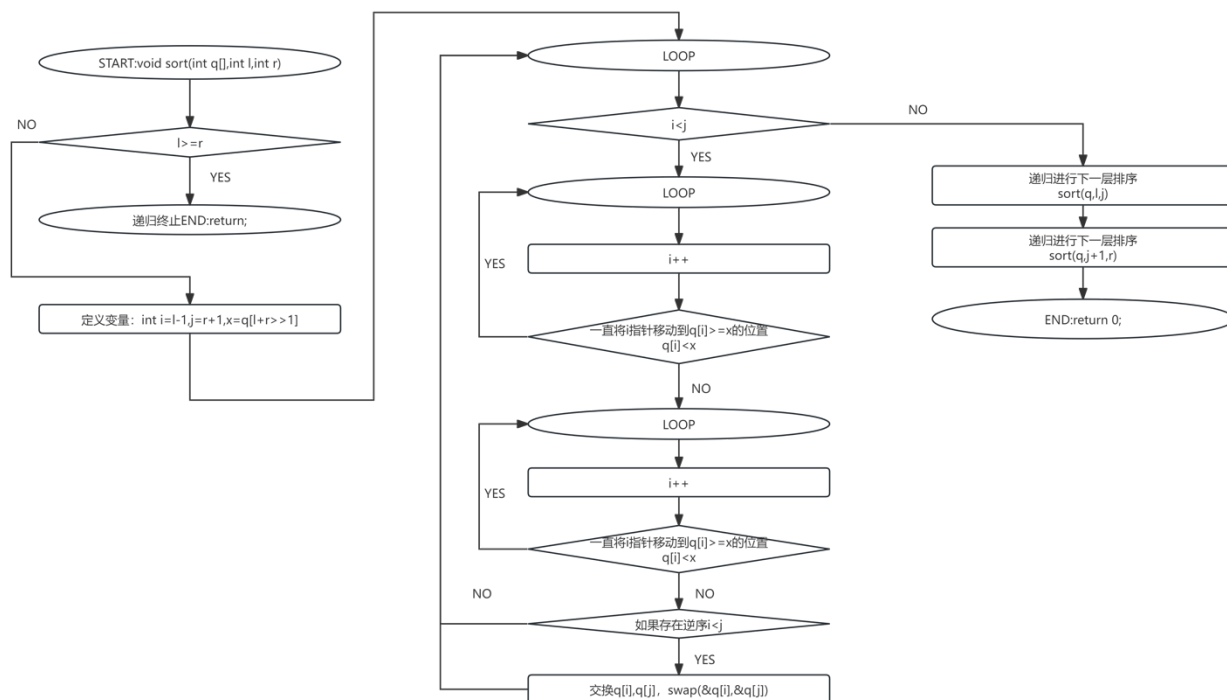
编写合并整数数组的函数。数组 $a[M]$ 中有 m 个元素 ($m < M$)，数组 $b[N]$ 中有 n 个元素 ($n < N$)，且 $m+n \leq M$ ，将 a 、 b 两个数组按存放的整数升序排序并合并放入数组 a 中。

[流程图]

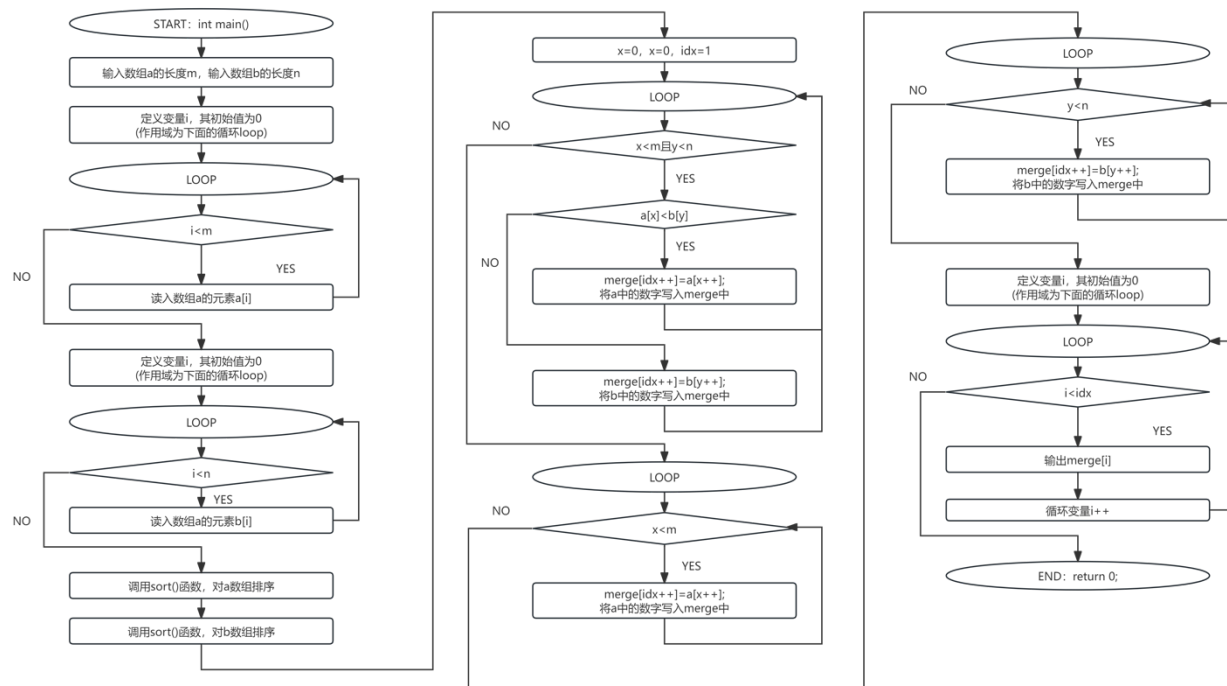
下面是 `void swap(int *x, int *y)` 函数的流程图：



下面是 `sort(int q[], int l, int r)` 快速排序函数的流程图：



下面是 `main()` 函数的流程图



[代码]

```

#include<stdio.h>
const int N=110;
int n,m,a[N],b[N],merge[2*N]; //定义数组 a, b, 用于存放归并之前的值, merge 用于存放归并以后的值
void swap(int *x, int *y) //定义 swap 函数用于交换两个值, 使用指针传入地址
{

```

```

    int tmp = *x;
    *x = *y;
    *y = tmp;
}
void sort(int q[],int l,int r)//定义快速排序函数 sort
{
    if(l>=r) return;//递归的终止条件: l>=r
    int i=l-1,j=r+1,x=q[l+r>>1]; //x 是快速排序的基准值
    while (i<j)//移动双指针, 让基准值左边全部小于 x, 右边全部大于 x
    {
        do i++; while(q[i]<x);
        do j--; while(q[j]>x);
        if(i<j) swap(&q[i],&q[j]);//交换
    }
    sort(q,l,j);//递归进行下一层排序
    sort(q,j+1,r);
}
int main()
{
    scanf("%d%d",&n,&m);//输入元素个数
    for(int i=0;i<m;++i) scanf("%d",&a[i]);//输入数组 a 的值
    for(int i=0;i<n;++i) scanf("%d",&b[i]);//输入数组 b 的值
    sort(a,0,m-1);//对 a 数组排序
    sort(b,0,n-1);//对 b 数组排序
    int x=0,y=0,idx=0;//归并前的准备
    while(x<m&&y<n)//在其中一个走到底之前, 归并 a, b 到 merge 中
    {
        if(a[x]<b[y]) merge[idx++]=a[x++];
        else merge[idx++]=b[y++];
    }
    while(x<m) merge[idx++]=a[x++]; //归并剩下的 a 数组元素到 merge 中
    while(y<n) merge[idx++]=b[y++]; //归并剩下的 b 数组元素到 merge 中
    for(int i=0;i<idx;++i) printf("%d ",merge[i]);//输出归并以后的结果
    return 0;
}

```

[运行结果]

```

5 5
128 238 127 349 129
328 173 11 23 49
11 23 49 127 128 129 173 238 328 349 %
○ fang50253@MacBook-Pro 程序设计语言实习 %

```

[小结]

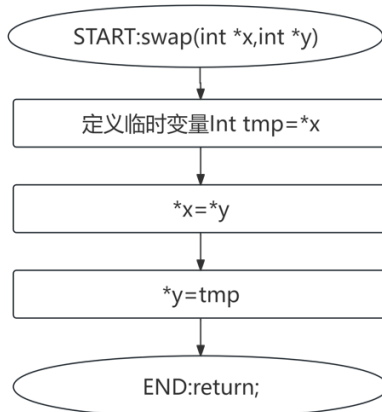
此题涉及到了快速排序算法、递归、双指针算法、函数返回等内容。该代码实现了两个已排序数组的合并，并通过快速排序对数组进行排序。使用了数组和输入输出、指针和函数调用、递归和快速排序、双指针及循环等 C 语言基础知识。其中，swap 函数用于交换元素，sort 函数实现快速排序，合并过程采用双指针技术，最后输出合并后的有序数组。

[题目]

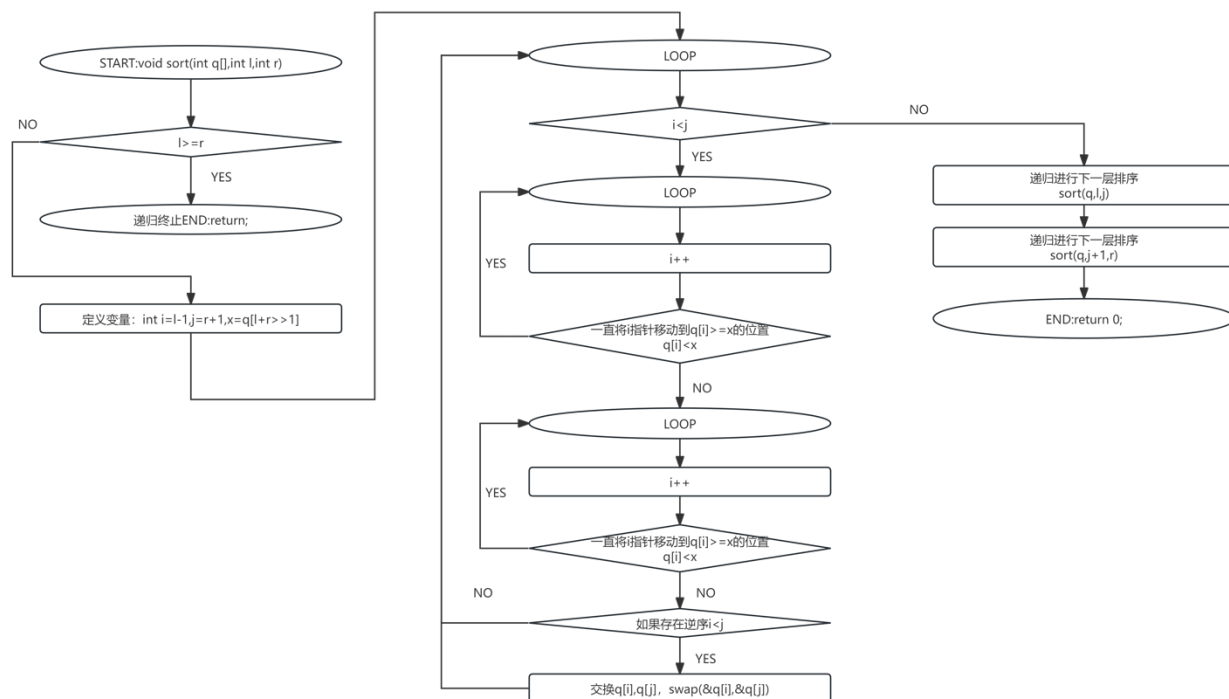
数组插入问题：生成一个 10 元素构成的一维数组，数组元素由用户随机输入。要求：先按照升序排列并输出。再输入一个数，按照升序的规律将其插入并输出。

[流程图]

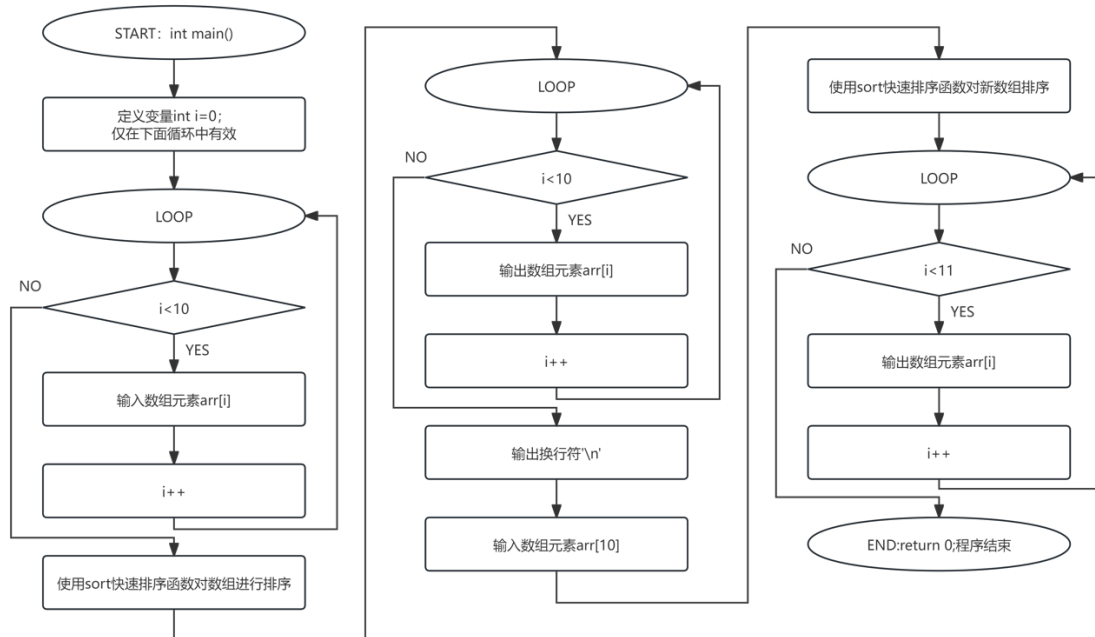
下面是 void swap(int *x, int *y) 函数的流程图：



下面是 sort(int q[], int l, int r) 快速排序函数的流程图：



下面是 int main() 函数的流程图：



[代码]

```

#include<stdio.h>
int arr[11];
void swap(int *x, int *y)//定义 swap 函数用于交换两个值，使用指针传入地址
{
    int tmp = *x;
    *x = *y;
    *y = tmp;
}
void sort(int q[],int l,int r)//定义快速排序函数 sort
{
    if(l>=r) return;//递归的终止条件: l>=r
    int i=l-1,j=r+1,x=q[l+r>>1]; //x 是快速排序的基准值
    while (i<j)//移动双指针，让基准值左边全部小于 x，右边全部大于 x
    {
        do i++; while(q[i]<x);
        do j--; while(q[j]>x);
        if(i<j) swap(&q[i],&q[j]);//交换
    }
    sort(q,l,j);//递归进行下一层排序
    sort(q,j+1,r);
}
int main()
{
    for(int i=0;i<10;++i) scanf("%d",&arr[i]);
    sort(arr,0,9);
}
  
```



```
for(int i=0;i<10;++i) printf("%d ",arr[i]);
printf("\n");
scanf("%d",&arr[10]);
sort(arr,0,10);
for(int i=0;i<11;++i) printf("%d ",arr[i]);
return 0;
}
```

[运行结果]

```
1 warning generated.
12 23 34 57 253 602 172 162 273 21
12 21 23 34 57 162 172 253 273 602
99
12 21 23 34 57 99 162 172 253 273 602
o fang50253@MacBook-Pro 程序设计语言实习 %
```

[小结]

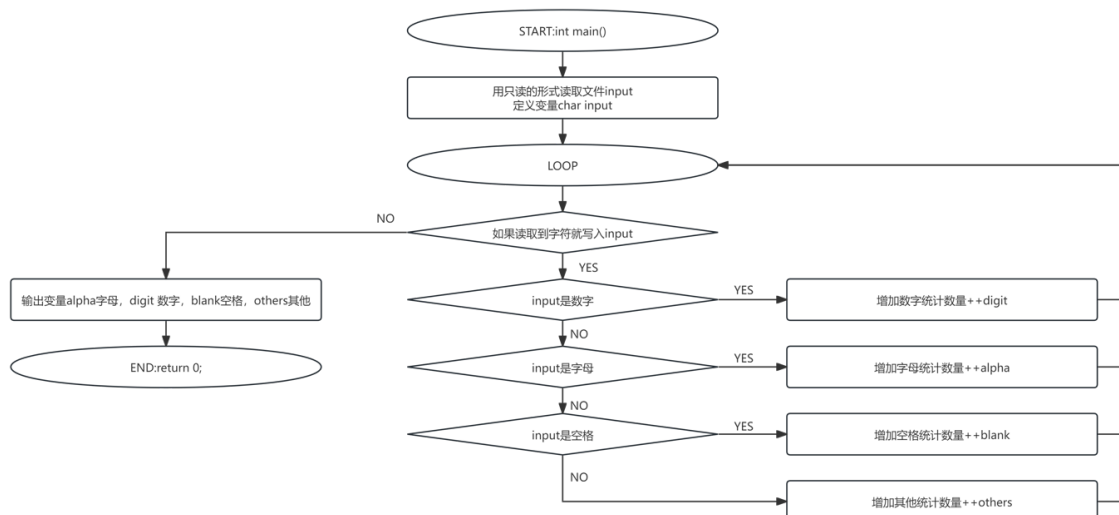
这段代码实现了一个快速排序和插入操作，主要用于对整数数组进行排序。首先，定义了一个数组 `arr` 来存储输入的整数，并通过 `scanf` 函数获取用户输入。为了实现排序，代码中定义了 `swap` 和 `sort` 两个函数，其中 `swap` 函数通过指针参数交换两个整数的值，`sort` 函数则采用递归的方式实现快速排序算法。在 `sort` 函数中，选取数组的中间值作为基准值，通过双指针技术将数组分为两部分，使得基准值左边的元素都小于基准值，右边的元素都大于基准值。然后，递归地对两部分分别进行排序。主函数首先对前 10 个输入的整数进行排序并输出结果，接着读取第 11 个整数并将其插入数组中，再次调用 `sort` 函数对新的数组进行排序并输出排序后的结果。这段代码展示了数组操作、指针传递、递归调用、双指针以及快速排序算法的应用。

模块三

[题目]

统计频率：读入一个文件，文件中包含字母，数字，空格，标点符号等。请统计文件中的字母，数字，空格和其他符号的数目，在屏幕上显示。

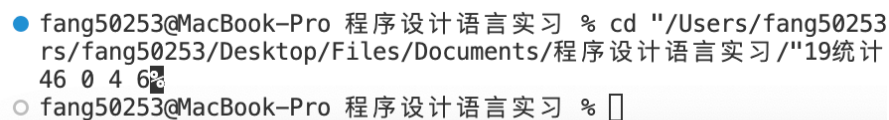
[流程图]



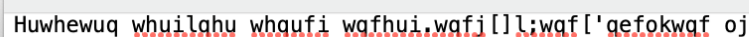
[代码]

```
#include<stdio.h>
#include<stdlib.h>
#include<ctype.h> //引入字符类型判断头文件
int digit,others,alpha,blank; //初始化每一个桶
int main()
{
    FILE *fp=fopen("input","r"); //以只读形式打开一个文件
    char input; //读取字符
    while((input=fgetc(fp))!=EOF) //逐个读取字符，直到文件结束
    {
        if(isdigit(input)) {++digit;continue;} //数字
        if(isalpha(input)) {++alpha;continue;} //字母
        if(input==' ') {++blank;continue;} //空格
        ++others; //其他
    }
    printf("%d %d %d %d",alpha,digit,blank,others); //输出统计个数
    fclose(fp); //关闭文件
    return 0;
}
```

[运行结果]



```
● fang50253@MacBook-Pro 程序设计语言实习 % cd "/Users/fang50253
rs/fang50253/Desktop/Files/Documents/程序设计语言实习/"19统计
46 0 4 6
○ fang50253@MacBook-Pro 程序设计语言实习 %
```



HuwHewuq whuilaHu whauFi wafhui.wafi[];waf['aefokwaf oj

[小结]

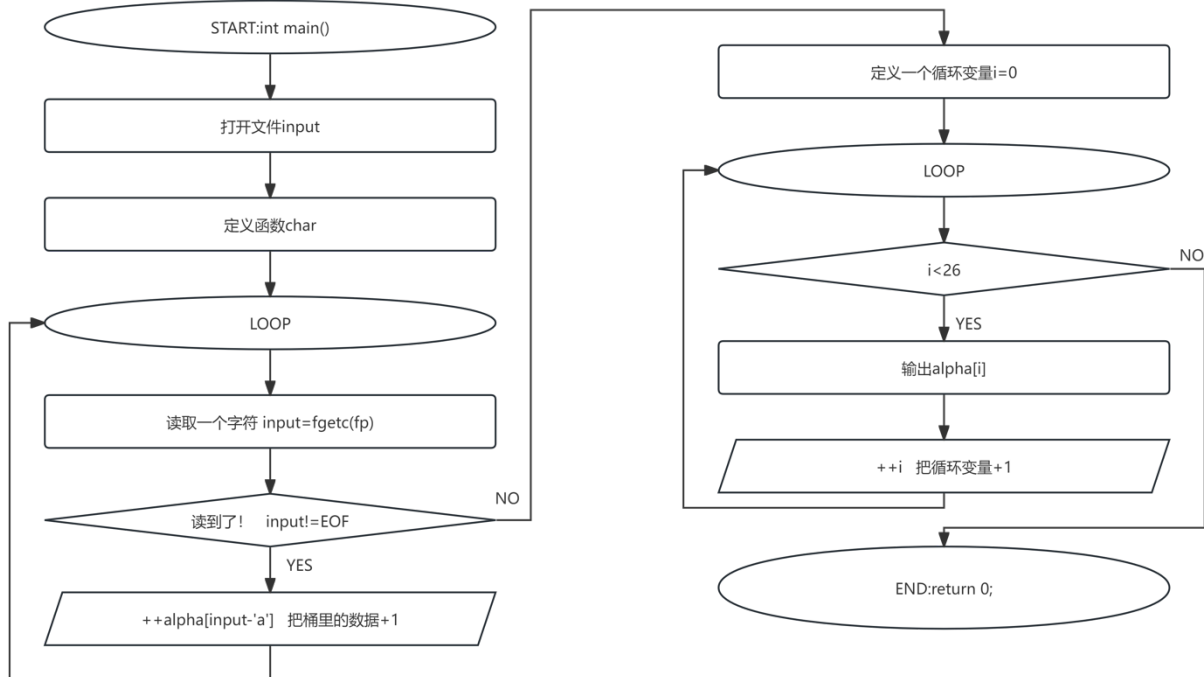
这段代码展示了使用 C 语言进行文件读取和字符分类的基本方法。首先，通过 `#include` 指令引入必要的头文件，如 `stdio.h` 用于输入输出操作，`stdlib.h` 提供了标准库函数，`ctype.h` 用于字符类型判断。变量 `digit`、`others`、`alpha`、`blank` 被声明并初始化为 0，用于计数不同类型的字符。在 `main` 函数中，使用 `fopen` 函数打开名为“input”的文件，并使用 `fgetc` 函数逐个读取字符。读取的字符通过 `isdigit` 函数判断是否为数字，通过 `isalpha` 函数判断是否为字母，通过比较判断是否为空格，分别计数到对应的变量中，所有其他字符计数到 `others` 中。读取结束后，使用 `printf` 函数输出各类别字符的计数结果，最后使用 `fclose` 函数关闭文件。

[题目]

统计字母个数。读取文件中的字符串，统计从“a”到“z”26 个字母各自出现的次数，并将结果放

入数组中。如文件中有字符串abcde f gabcdeabc，输出33322110000000000000000000。

[流程图]



[代码]

```

#include<stdio.h>
int alpha[26];
int main()
{
    FILE *fp=fopen("input","r+");
    char input;
    while((input=fgetc(fp))!=EOF)
    {
        if(input==' ') continue;
        ++alpha[input-'a'];
    }
    for(int i=0;i<26;++i) printf("%d",alpha[i]);
    fclose(fp);
    return 0;
}

```

[运行结果]

```

● fang50253@MacBook-Pro 程序设计语言实习 % cd
  言实习/"21统计字母个数
  33322110000000000000000000%

```

```

| abcde f gabcdeabc

```

[小结]

这段代码实现了统计文本文件中每个小写字母出现次数的功能。首先,定义了一个大小为 26 的数组 `alpha` 用于存储每个字母的计数值。打开文件 `input` 并逐个读取字符。若字符是空格,则跳过。否则,根据字符的 ASCII 值更新对应的计数器。最后,打印 `alpha` 数组中每个字母的计数值。关闭文件并返回。代码使用 `fgetc` 读取字符,`fopen` 和 `fclose` 进行文件操作,并通过 `input - 'a'` 计算数组索引来更新字母计数。

模块四

[题目]

分数比较问题:比较两个分数的大小。

[代码]

```
#include<stdio.h>
struct D//定义一个分数的结构体
{
    int a,b;//a 代表分子, b 代表分母
}x,y;//定义两个数字
int main()
{
    scanf("%d/%d %d/%d",&x.a,&x.b,&y.a,&y.b);//读取分数, 格式类似 3/5 2/7
    if(1.0*x.a/x.b>1.0*y.a/y.b) printf("x is larger than y");//x 比 y 大
    else if(1.0*x.a/x.b<1.0*y.a/y.b) printf("y is larger than x");//y 比 x 大
    else printf("x is smae as y");//两者一样大
    return 0;
}
```

[运行结果]

```
● fang50253@MacBook-Pro 程序设计语言实习 % cc
  && "/Users/fang50253/Desktop/Files/Document
  3/5 7/3
  y is larger than x%
○ fang50253@MacBook-Pro 程序设计语言实习 % █
```

[小结]

这段代码展示了 C 语言中结构体的定义与使用、输入输出操作以及条件判断。结构体 D 定义了两个整数成员 a 和 b,并声明了两个结构体变量 x 和 y。scanf 函数用于读取两个分数,并分别存储到结构体变量中。通过计算两个分数的浮点数值并进行比较,printf 函数输出比较结果。代码使用了基本的结构体操作、浮点数计算和条件判断。

模块五

[题目]

饭卡管理程序。

- (1). 建立饭卡信息：添加若干人的饭卡号、姓名、金额，要求饭卡号是唯一的；
- (2). 买饭：要求用户输入饭卡号、饭费，系统自动从该人的饭卡中减去饭钱，并分别显示买饭前后的金额，如果原来饭卡中的余额不足 5 元，则不能买饭，显示“余额不足，请充值”；
- (3). 充值：输入饭卡号、充值金额，充值完成后显示充值前后的金额。

[代码]

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#define CLS "clear"//在win系统中改成"cls" 调试端为macOS
const int MAXN=1e3+10;//数据量大上限
int data_open;//日志模式开关
int idx=0;//数据量顶
int add_users(char num[],char name[]);//用于添加用户
int buy(char num[],int fee);//用于消费
void main_tab();//首页
int recharge(char num[],int fee);//用于充值
int read();//用于读取数据
int write();//用于写入数据
void add_users_input();
void buy_input();
void recharge_input();
//代码中所有的 input 均为前端处理函数
int query_residue(char num[]);
struct DataBase//数据库结构体
{
    char num[30];
    char name[30];
    int money;//这边使用 int 类型存储是防止 double 类型存储的金钱存在偏移，扩大 100 倍显示
}info[MAXN];
void query_residue_output()
{
    char num[100];
    printf("请输入需要查询的卡号，回车键结束\n");
    scanf("%s",num);
    int v=query_residue(num);//记录返回值等待处理
    if(v==-1)
    {
        if(data_open) system(CLS);
        printf("用户名不存在\n");//处理特殊情况
        main_tab();//返回主菜单
    }
}
```

```
        return;
    }
    if(data_open) system(CLS);
    printf("余额为%.2lf\n",v/100.0);//输出余额
    main_tab();
    return;
}

int read()
{
    FILE *fp=fopen("data.fzy","r");//以制度模式打开文件
    if(fp==NULL) return 0;//如果文件读取失败那么不再继续执行，代表程序为新程序
    fscanf(fp,"%d",&data_open);//读取日志模式状态
    char num[100],name[100];//文件读取缓冲区
    int money;//文件读取缓冲区
    memset(info,0,sizeof info);//清空原有数据
    idx=0;//清空数据顶
    while(fscanf(fp,"%s%s%d",num,name,&money)==3)
    {
        ++idx;
        if(idx>=MAXN-10) printf("Error,数据量过大\n");
        //数据最大规模可以在代码顶部全局变量区进行调整
        strcpy(info[idx].num,num);//将文件写入内存
        strcpy(info[idx].name,name);//将文件写入内存
        info[idx].money=money;//将文件写入内存
    }
    fclose(fp);//关闭文件，停止占用
    return 0;
}

int write()
{
    FILE *fp=fopen("data.fzy","w+");//写文件
    fprintf(fp,"%d\n",data_open);//写入日志模式配置文件
    for(int i=1;i<=idx;++i)
        fprintf(fp,"%s %s %d\n",info[i].num,info[i].name,info[i].money);
    fclose(fp);//关闭 fp 文件停止占用
    printf("数据保存成功\n");//宿处保存成功提示
    return 0;
}

int query_residue(char num[])
//返回的数组是余额的 100 倍
//返回值-1 代表用户名不存在
{
    int users_num=-1;//进行标记防止数据读入失败
    for(int i=1;i<=idx;++i)
```

```
{
    struct DataBase output=info[i];
    if(strcmp(output.num,num)==0)//找到用户名所在的位置
    {
        users_num=i;
        break;
    }
}
if(users_num==-1) return -1;//用户名不存在
return info[users_num].money;//返回结果
}
void recharge_input()
{
    char num[100];//用户名缓冲区
    double drawin;//存入的钱(原始金额)
    printf("请输入卡号和充值金额, 使用空格分开\n");
    scanf("%s%lf",num,&drawin);
    int v=recharge(num,(int)(drawin*100));
    //通过 recharge()函数获取一个返回值
    if(v==-1)
    {
        if(data_open) system(CLS);
        printf("操作失败(返回值-1): 用户名不存在\n");//处理异常情况
        main_tab();//返回主界面
        return;
    }
    printf("用户名%s 充值后卡内余额剩余%.2lf 元\n",num,v/100.0);//前端提示
    main_tab();//返回主页面
    return;
}
void buy_input()
{
    char num[100];
    double fee;
    printf("请输入卡号和消费金额, 使用空格分开\n");
    scanf("%s %lf",num,&fee);
    int v=buy(num,(int)(fee*100));
    switch(v)
    {
        case -1://处理买饭的第一个异常: 用户名不存在
        {
            if(data_open) system(CLS);
            printf("操作失败(返回值-1): 用户名不存在\n");
            main_tab();
        }
    }
}
```

```
        return;
    }
    case -2://处理买饭的第二个异常：余额不足 5 元
    {
        if(data_open) system(CLS);
        printf("操作失败(返回值-2)：原始卡内余额不足 5 元，请充值\n");
        main_tab();
        return;
    }
    case -3://处理买饭的第三个异常：余额有 5 元但不足以支付
    {
        if(data_open) system(CLS);
        printf("操作失败(返回值-3)：无法完成此次消费，因为卡内余额不足以支付饭费\n");
        main_tab();
        return;
    }
    case 0://没有异常
    {
        if(data_open) system(CLS);
        int residue=query_residue(num);
        //这边的 residue 不可能为-1，如果返回值为-1，用户不存在，则上面的 case -1 已经解决了这个问题
        printf("操作成功，卡号%s 已完成消费%.2lf 元,余额%.2lf 元\n",num,fee,residue/100.0);
        //返回消费的金额，剩余的余额等信息
        main_tab();//返回主屏幕
        return;
    }
}
}

void add_users_input()//新增用户的前端页面
{
    char num[100],name[100];//用户名和卡号的缓冲区
    printf("请输入卡号和用户，使用空格分开，要求不超过 25 个字符\n");
    scanf("%s %s",num,name);//输入想要添加的用户名
    int v=add_users(num,name);//获取一个返回值
    switch(v)
    {
        case -1://处理用户名添加的第一个异常：用户名重复
        {
            if(data_open) system(CLS);
            printf("操作失败(返回值-1)：用户重复添加\n");
            main_tab();
            return;
        }
        case -2://获取用户名添加的第二个异常：卡号溢出
```



```
{
    if(data_open) system(CLS);
    printf("操作失败(返回值-2): 卡号超长\n");
    main_tab();
    return;
}
case -3://获取用户名添加的第三个异常: 用户名溢出
{
    if(data_open) system(CLS);
    printf("操作失败(返回值-3): 用户名超长\n");
    main_tab();
    return;
}
case 0://没有异常
{
    if(data_open) system(CLS);
    printf("操作成功, 已添加用户%s %s\n", num, name);
    main_tab();
    return;
}
}
}
void main_tab()
{
    //下面是主界面
    for(int i=1;i<=50;++i) printf("#");
    for(int i=1;i<=3;++i) printf("\n");
    printf("    fzy 饭卡管理程序@2024-07-28\n");
    printf("    请输入以下数字, 选择您所要进行的操作: \n");
    printf("    1.从磁盘中读取记录\n");
    printf("    2.向磁盘中写入记录\n");
    printf("    3.新增用户\n");
    printf("    4.买饭消费\n");
    printf("    5.卡片充值\n");
    printf("    6.查询卡内余额\n");
    printf("    7.打开或关闭日志开关\n");
    printf("    8.退出系统\n");
    for(int i=1;i<=2;++i) printf("\n");
    for(int i=1;i<=50;++i) printf("#");
    printf("\n 请输入您的选择(回车键确认): ");
    //主界面打印完成, 供用户选择
    while(1)
    {
        int choose;
```

```
scanf("%d",&choose);//不断重复询问选择
switch(choose)
{
    case 1:read();break;//读文件
    case 2:write();break;//写文件
    case 3:add_users_input();break;//增加用户
    case 4:buy_input();break;//买饭
    case 5:recharge_input();break;//充值
    case 6:query_residue_output();break;//查询余额
    case 7:data_open=!data_open;break;//打开或关闭数据日志
    case 8:return;//退出软件
}
}
}
int add_users(char num[],char name[])
//返回值为-1, 代表程序执行出现错误:用户重复添加
//返回值为-2:卡号超长
//返回值为-3:姓名超长
//返回值为 0, 代表用户添加成功
{
    for(int i=1;i<=idx;++i)
    {
        struct DataBase output=info[i];
        if(strcmp(output.num,num)==0) return -1;
    }
    //执行到这一步, 代表用户没有被重复添加
    ++idx;
    if(strlen(num)>25) return -2;//卡号超长判断
    if(strlen(name)>25) return -3;//姓名超长判断
    strcpy(info[idx].num,num);//将数据写入
    strcpy(info[idx].name,name);//将数据读出
    info[idx].money=0;
    write();
    return 0;
}
int buy(char num[],int fee)
//返回值为-1, 代表用户不存在
//返回值为-2, 代表余额不足
//返回值为-3, 代表余额不足以支付饭费
//返回值为 0, 代表程序正常运行结束
{
    int users_num=-1;
    for(int i=1;i<=idx;++i)
    {
```

```
    struct DataBase output=info[i];
    if(strcmp(output.num,num)==0)
    {
        users_num=i;
        break;
    }
}
if(users_num==-1) return -1;//用户名不存在
if(info[users_num].money<500) return -2;//卡内原有余额不足5元，需充值
if(info[users_num].money-fee<0) return -3;//卡内余额不足以消费，需充值
info[users_num].money-=fee;
write();
return 0;
}
int recharge(char num[],int fee)
//返回-1: 用户名不存在
//返回到数字为充值后到余额
{
    int users_num=-1;//搜索是否存在用户
    for(int i=1;i<=idx;++i)
    {
        struct DataBase output=info[i];
        if(strcmp(output.num,num)==0)
        {
            users_num=i;
            break;
        }
    }
    if(users_num==-1) return -1;//用户名不存在
    info[users_num].money+=fee;
    write();//写入文件，保存信息
    return info[users_num].money;//返回余额(余额的100倍)
}
int main()
{
    read();//读取数据
    if(data_open) system(CLS);
    main_tab();//进入主页面
    printf("已退出系统\n");//退出提示
    return 0;
}
```

[运行结果]

```
#####

fzy饭卡管理程序@2024-07-28
请输入以下数字，选择您所要进行的操作：
1.从磁盘中读取记录
2.向磁盘中写入记录
3.新增用户
4.买饭消费
5.卡片充值
6.查询卡内余额
7.打开或关闭日志开关
8.退出系统

#####

fzy饭卡管理程序@2024-07-28
请输入以下数字，选择您所要进行的操作：
1.从磁盘中读取记录
2.向磁盘中写入记录
3.新增用户
4.买饭消费
5.卡片充值
6.查询卡内余额
7.打开或关闭日志开关
8.退出系统

#####

请输入您的选择(回车键确认)：3
请输入卡号和用户，使用空格分开，要求不超过25个字符
234 teacher
数据保存成功
操作成功，已添加用户234 teacher

#####

fzy饭卡管理程序@2024-07-28
请输入以下数字，选择您所要进行的操作：
1.从磁盘中读取记录
2.向磁盘中写入记录
3.新增用户
4.买饭消费
5.卡片充值
6.查询卡内余额
7.打开或关闭日志开关
8.退出系统

#####

请输入您的选择(回车键确认)：5
请输入卡号和充值金额，使用空格分开
123 25
数据保存成功
用户名123充值后卡内余额剩余131.56元
.....

#####

fzy饭卡管理程序@2024-07-28
请输入以下数字，选择您所要进行的操作：
1.从磁盘中读取记录
2.向磁盘中写入记录
3.新增用户
4.买饭消费
5.卡片充值
6.查询卡内余额
7.打开或关闭日志开关
8.退出系统

#####

请输入您的选择(回车键确认)：4
请输入卡号和消费金额，使用空格分开
123 23.45
数据保存成功
操作成功，卡号123已完成消费23.45元,余额108.11元
```

[小结]

这段代码实现了一个简易的饭卡管理系统，包括新增用户、用户充值、用户消费、查询余额等功能。代码首先定义了常量和全局变量，例如清屏命令、最大数据量、日志模式开关和数据索引等。然后定义了一系列函数以实现具体功能，包括用户新增、充值、消费、查询余额、数据读写等。程序使用结构体 `DataBase` 存储用户信息，每个用户包含卡号、姓名和余额等属性。数据保存在 `info` 数组中。`main()` 函数首先调用 `read()` 函数读取数据文件 `data.fz`，初始化 `info` 数组和数据索引 `idx`。然后调用 `main_tab()` 函数显示主菜单，用户可以选择不同功能选项 `read()` 和 `write()` 函数负责从文件中读取数据和将数据写入文件。文件操作使用 `fopen`、`fscanf` 和 `fprintf` 等标准库函数进行。`add_users_input()` 函数是用户新增功能的前端处理函数，提示用户输入卡号和姓名，并调用 `add_users()` 函数进行用户添加操作 `add_users()` 函数首先检查卡号是否重复，然后检查卡号和姓名长度是否合法，最后将新用户信息添加到 `info` 数组，并调用 `write()` 函数保存数据。`buy_input()` 和 `recharge_input()` 函数分别处理用户消费和充值的前端输入，通过提示用户输入卡号和金额，并调用相应的 `buy()` 和 `recharge()` 函数处理具体业务逻辑。`buy()` 函数首先检查用户是否存在，然后检查余额是否充足，最后更新用户余额并保存数据 `recharge()` 函数首先检查用户是否存在，然后更新余额并保存数据。`query_residue_output()` 函数处理查询余额的前端输入，提示用户输入卡号，并调用 `query_residue()` 函数查询余额。`query_residue()` 函数遍历 `info` 数组查找用户，并返回用户余额。代码中的所有输入处理函数均调用 `main_tab()` 函数返回主菜

单，保持用户体验的一致性。日志模式开关通过全局变量 `data_open` 控制，可以通过主菜单中的选项打开或关闭。代码使用了标准库函数 `strcmp` 进行字符串比较，`strcpy` 进行字符串复制，`system` 调用系统命令进行清屏，`fscanf` 和 `fprintf` 进行文件读写等。代码结构清晰，功能模块化，每个函数职责单一，易于维护和扩展。