BÀI THỰC HÀNH SỐ 06

1. MỤC TIÊU

- Hiểu một số khái niệm về tập tin;
- Biết các bước thao tác với tập tin;
- Biết sử dụng một số hàm truy xuất đến tập tin văn bản;
- Biết sử dụng một số hàm truy xuất đến tập tin nhị phân.

2. LÝ THUYẾT CẦN GHI NHỚ

2. 1. Các bước thao tác trên tập tin

Muốn thao tác trên tập tin, ta phải lần lượt làm theo các bước:

- Khai báo biến tập tin.
- Mở tập tin bằng hàm *fopen()*.
- Thực hiện các thao tác xử lý dữ liệu của tập tin bằng các hàm đọc/ghi dữ liệu.
- Đóng tập tin bằng hàm *fclose()*.

Ở đây, ta thao tác với tập tin nhờ các hàm được định nghĩa trong thư viện stdio.h.

*Bước 1.**Khai báo biến tập tin***

- ***Cú pháp:*** `FILE <Danh sách các biến con trỏ>`

Các biến trong danh sách phải là các con trỏ và được phân cách bởi dấu phẩy(,).

- ***Ví dụ 2.1*** `FILE *f1,*f2;`

*Bước 2.**Mở tập tin***

- **Cú pháp:** `FILE *fopen(char *Path, const char *Mode)`

Trong đó:

- *Path*: chuỗi chỉ đường dẫn đến tập tin trên đĩa.
- *Mode*: chuỗi xác định cách thức mà tập tin sẽ mở. Các giá trị có thể của *Mode*:

Chế độ	*Ý nghĩa*
r	Mở tập tin văn bản để đọc
w	Tạo ra tập tin văn bản mới để ghi
a	Nối vào tập tin văn bản
rb	Mở tập tin nhị phân để đọc
wb	Tạo ra tập tin nhị phân để ghi

ab	Nối vào tập tin nhị phân
r+	Mở một tập tin văn bản để đọc/ghi
w+	Tạo ra tập tin văn bản để đọc ghi
a+	Nối vào hay tạo mới tập tin văn bản để đọc/ghi
r+b	Mở ra tập tin nhị phân để đọc/ghi
w+b	Tạo ra tập tin nhị phân để đọc/ghi
a+b	Nối vào hay tạo mới tập tin nhị phân

Hàm *fopen* trả về một con trỏ tập tin. Chương trình của ta không thể thay đổi giá trị của con trỏ này. Nếu có một lỗi xuất hiện trong khi mở tập tin thì hàm này trả về con trỏ *NULL*.

Ví dụ 2.2 Mở một tập tin tên c:\\ TEST.txt để ghi.

```
FILE *f;
f = fopen("c:\\TEST.txt", "w");
if (f!=NULL)
{
     /* Các câu lệnh để thao tác với tập tin*/
     /* Đóng tập tin*/
}
```

Trong ví dụ trên, ta có sử dụng câu lệnh kiểm tra điều kiện để xác định mở tập tin có thành công hay không?

Khi mở tập tin để ghi (chế độ *w*), nếu tập tin đã tồn tại rồi thì nội dung của tập tin sẽ bị xóa và một tập tin mới được tạo ra.

Nếu ta muốn ghi nối dữ liệu, ta phải sử dụng chế độ "*a*".

Khi mở với chế độ đọc, tập tin phải tồn tại rồi, nếu không một lỗi sẽ xuất hiện.

Bước 3. *Các thao tác đọc/ghi dữ liệu trên file*

Tùy thuộc chế độ (mode) được dùng khi mở tập tin (văn bản hay nhị phân) mà ta có thể sử dụng các hàm đọc ghi tương ứng. Các hàm đọc/ghi trên file văn bản hoặc nhị phân sẽ được giới thiệu trong các mục 11.3. Truy cập tập tin văn bản và 11.4. Truy cập tập tin nhị phân.

Bước 4. *Đóng tập tin*

i. Hàm fclose()

- Được dùng để đóng tập tin được mở bởi hàm *fopen()*.
- Hàm này sẽ ghi dữ liệu còn lại trong vùng đệm vào tập tin và đóng lại tập tin.

- Cú pháp: `int fclose (FILE *f)`

  Trong đó *f* là con trỏ tập tin được mở bởi hàm *fopen()*.

- Giá trị trả về của hàm:
  - Là *0* báo rằng việc đóng tập tin thành công.
  - Là *EOF* nếu có xuất hiện lỗi.

ii. Hàm fcloseall()

- Để đóng tất cả các tập tin lại.
- Cú pháp: `int fcloseall()`
- Kết quả trả về của hàm:
  - Là tổng số các tập tin được đóng lại.
  - Nếu không thành công, kết quả trả về là *EOF*.

iii.Kiểm tra đến cuối tập tin hay chưa?

- Cú pháp: `int feof (FILE *f)`
- Ý nghĩa: Kiểm tra xem đã chạm tới cuối tập tin hay chưa và trả về *EOF* nếu cuối tập tin được chạm tới, ngược lại trả về *0*.

iv. Di chuyển con trỏ tập tin về đầu tập tin

- Khi ta đang thao tác một tập tin đang mở, con trỏ tập tin luôn di chuyển về phía cuối tập tin. Muốn cho con trỏ quay về đầu tập tin như khi mở nó, ta sử dụng hàm *rewind()*.
- Cú pháp: `void rewind (FILE *f)`

2. 2. Truy cập tập tin văn bản

2.2.1. Ghi dữ liệu lên tập tin văn bản

2.2.1.1. Hàm putc()

- Hàm này được dùng để ghi một ký tự lên một tập tin văn bản đang được mở để làm việc.
- Cú pháp: `int putc (int c, FILE *f)`

Trong đó, tham số *c* chứa mã ASCII của một ký tự nào đó. Mã này được ghi lên tập tin liên kết với con trỏ *f*. Hàm này trả về *EOF* nếu gặp lỗi.

2.2.1.2. Hàm fputs()

- Hàm này dùng để ghi một chuỗi ký tự chứa trong vùng đệm lên tập tin văn bản.
- Cú pháp: `int puts (const char *buffer, FILE *f)`

Trong đó, buffer là con trỏ có kiểu char chỉ đến vị trí đầu tiên của chuỗi ký tự được ghi vào. Hàm này trả về giá trị *0* nếu buffer chứa chuỗi rỗng và trả về *EOF* nếu gặp lỗi.

2.2.1.3. Hàm fprintf()

- Hàm này dùng để ghi dữ liệu có định dạng lên tập tin văn bản.
- Cú pháp: **`fprintf (FILE *f, const char *format, varexpr)`**

Trong đó:

- *format*: chuỗi định dạng (giống với các định dạng của hàm *printf()*).
- *varexpr*: danh sách các biểu thức, mỗi biểu thức cách nhau dấu phẩy (,).

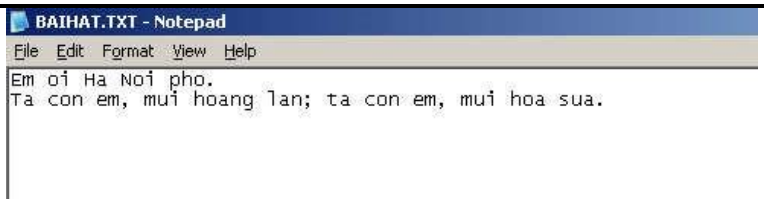
Định dạng	***Ý nghĩa***
`%d`	Ghi số nguyên
`%[`.số chữ số thập phân`]` **`f`**	Ghi số thực có <số chữ số thập phân> theo quy tắc làm tròn số.
`%o`	Ghi số nguyên hệ bát phân
`%x`	Ghi số nguyên hệ thập lục phân
`%c`	Ghi một ký tự
`%s`	Ghi chuỗi ký tự
`%e` hoặc `%E` hoặc `%g` hoặc `%G`	Ghi số thực dạng khoa học (nhân 10 mũ x)

2.2.1.4. Ví dụ 2.3

Viết chương trình ghi chuỗi ký tự lên tập tin văn bản D:\\Baihat.txt

```
#include<stdio.h>
#include<conio.h>
int main ()
{
     FILE *f;
     clrscr ();
     f = fopen ("D:\\Baihat.txt","r+");
     if (f!=NULL)
     {
          fputs("Em oi Ha Noi pho.\n",f);
          fputs("Ta con em, mui hoang lan; ta con em, mui hoa
          sua.",f);
          fclose(f);
     }
     return 0;
}
```

Nội dung tập tin Baihat.txt khi được mở bằng trình soạn thảo văn bản Notepad.



2.2.2. *Đọc dữ liệu từ tập tin văn bản*

2.2.2.1. *Hàm getc()*

- Hàm này dùng để đọc dữ liệu từ tập tin văn bản đang được mở để làm việc.
- Cú pháp: **`int getc (FILE *f)`**
- Hàm này trả về mã ASCII của một ký tự nào đó (kể cả *EOF*) trong tập tin liên kết với con trỏ *f*.

2.2.2.2. *Hàm fgets()*

- Cú pháp: **`char *fgets (char *buffer, int n, FILE *f)`**
- Hàm này được dùng để đọc một chuỗi ký tự từ tập tin văn bản đang được mở ra và liên kết với con trỏ *f* cho đến khi đọc đủ *n* ký tự hoặc gặp ký tự xuống dòng '\n' (ký tự này cũng được đưa vào chuỗi kết quả) hay gặp ký tự kết thúc *EOF* (ký tự này không được đưa vào chuỗi kết quả).

  Trong đó:

  - *buffer* (vùng đệm): con trỏ có kiểu char chỉ đến vùng nhớ đủ lớn chứa các ký tự nhận được.
  - *n*: giá trị nguyên chỉ độ dài lớn nhất của chuỗi ký tự nhận được.
  - *f*: con trỏ liên kết với một tập tin nào đó.
  - Ký tự *NULL* ('\0') tự động được thêm vào cuối chuỗi kết quả lưu trong vùng đệm.
- Hàm trả về địa chỉ đầu tiên của vùng đệm khi không gặp lỗi và chưa gặp ký tự kết thúc *EOF*. Ngược lại, hàm trả về giá trị *NULL*.

2.2.2.3. *Hàm fscanf()*

- Hàm này dùng để đọc dữ liệu từ tập tin văn bản vào danh sách các biến theo định dạng.
- Cú pháp: **`fscanf (FILE *f, const char *format, varlist)`**

  Trong đó:

  - *format*: chuỗi định dạng (giống hàm `scanf());`
  - *varlist*: danh sách các biến mỗi biến cách nhau dấu phẩy (,).

- ***Ví dụ 2.4***: Viết chương trình chép tập tin *D:\Baihat.txt* ở trên sang tập tin *D:\Baica.txt*.

```
#include<stdio.h>
#include<conio.h>
int main()
{
     FILE *f1,*f2;
     f1=fopen("D:\\Baihat.txt","rt");
     f2=fopen("D:\\Baica.txt","wt");
     if (f1!=NULL && f2!=NULL)
     {    int ch=fgetc (f1);
          while (! feof (f1))
               fputc(ch,f2); ch=fgetc(f1);
          fcloseall();
     }
     return 0;
}
```

2. 3. Truy cập tập tin nhị phân

2.3.1. *Ghi dữ liệu lên tập tin nhị phân*

- Cú pháp:

```
size_t fwrite(const void *ptr, size_t size, size_t n, FILE*f)
```

Trong đó:

- *ptr*: con trỏ chỉ đến vùng nhớ chứa thông tin cần ghi lên tập tin.
- *n*: số phần tử sẽ ghi lên tập tin.
- *size*: kích thước của mỗi phần tử.
- *f*: con trỏ tập tin đã được mở.

- Giá trị trả về của hàm này là số phần tử được ghi lên tập tin. Giá trị này bằng *n* trừ khi xuất hiện lỗi.

2.3.2. *Đọc dữ liệu từ tập tin nhị phân*

- Cú pháp:

```
size_t fread (const void *ptr, size_t size, size_t n, FILE *f)
```

Trong đó:

- *ptr*: con trỏ chỉ đến vùng nhớ sẽ nhận dữ liệu từ tập tin.
- *n*: số phần tử được đọc từ tập tin.
- *size*: kích thước của mỗi phần tử.
- *f*: con trỏ tập tin đã được mở.

- Giá trị trả về của hàm này là số phần tử đã đọc được từ tập tin. Giá trị này bằng *n* hay nhỏ hơn *n* nếu đã chạm đến cuối tập tin hoặc có lỗi xuất hiện.

2.3.3. Di chuyển con trỏ tập tin

- Việc ghi hay đọc dữ liệu từ tập tin sẽ làm cho con trỏ tập tin dịch chuyển một số byte, đây chính là kích thước của kiểu dữ liệu của mỗi phần tử của tập tin.
- Khi đóng tập tin rồi mở lại, con trỏ luôn ở vị trí ngay đầu tập tin.
- Nhưng nếu ta sử dụng kiểu mở tập tin là "*a*" để ghi nối dữ liệu, con trỏ tập tin sẽ di chuyển đến vị trí cuối cùng của tập tin này.
- Ta cũng có thể điều khiển việc di chuyển con trỏ tập tin đến vị trí chỉ định bằng hàm *fseek()*.
- Cú pháp: **`int fseek (FILE *f, long offset, int whence)`**

  Trong đó:

  - *fseek* di chuyển con trỏ f đến vị trí *offset* theo mốc *whence*. `fseek` trả về:
    - = 0 nếu thành công.
    - <>0 nếu di chuyển có lỗi.
  - *f* : con trỏ tập tin đang thao tác.
  - *offset*: số byte cần dịch chuyển con trỏ tập tin kể từ vị trí trước đó. Phần tử đầu tiên là vị trí *0*.
  - *whence*: vị trí bắt đầu để tính *offset*, ta có thể chọn điểm xuất phát là:
    - `#define SEEK_SET 0 //tính từ đầu tập tin`
    - `#define SEEK_CUR 1 //tính từ vị trí hiện hành của con trỏ`
    - `#define SEEK_END 2 // tính từ cuối tập tin`

3. BÀI TẬP CÓ HƯỚNG DẪN

1.1. Bài tập 1

- ***Yêu cầu***: Viết chương trình tạo tập tin chứa 10000 số nguyên ghi vào file *SONGUYEN.INP* mỗi dòng 10 số, với giá trị của các số nguyên được phát sinh ngẫu nhiên và nằm trong khoảng từ 0-999. Sau đó viết chương trình đọc file *SONGUYEN.INP*, sắp xếp theo thứ tự tăng dần và lưu kết quả vào file *SONGUYEN.OUT*. Thực hiện bằng 2 cách:
  - Thao tác trên tập tin văn bản.
  - Thao tác trên tập tin nhị phân.
- ***Thực hiện***

(i)- Thực hiện trên tập tin văn bản

```
//Khai báo thư viện cần dùng
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#include<conio.h>
//Định nghĩa các biến
#define MAX 10000
#define VALUE 1000
#define InFile "D:\\SoNguyen1.inp"
#define OutFile "D:\\SoNguyen1.out"
//Khai báo các nguyên mẫu hàm
void TaoFile();
void DocFile(int a[]);
void swap(int &a, int &b);
void Sort(int a[]);
void TaoFileKQ(int a[]);
void main()
{
     srand(time(NULL));
     int a[MAX];
     TaoFile();
     DocFile(a);
     XuatMang(a);
     TaoFileKQ(a);
}
void TaoFile()
{
     FILE *f;
     f = fopen(InFile, "wt");
     if (f == NULL)
          printf("Khong tao duoc file");
     else
     {    for (int i = 0; i < MAX / 10; i++)
          {      for (int j = 0; j < 10; j++)
               {
                    int so = rand() % VALUE;
                    fprintf(f,"%5d",so);
               }
          }
          fclose(f);
     }
}
void DocFile(int a[])
{
     FILE *f;
```

```
        f = fopen(InFile, "rt");
        if (f == NULL)
            printf("Khong tao duoc file");
        else
        {   for (int i = 0; i < MAX; i++)
                fscanf(f, "%d", &a[i]);
            fclose(f);
        }
    }
    void swap(int &a, int &b)
    {
        int temp = a;
        a = b;
        b = temp;
    }
    void Sort(int a[])
    {
        for (int i = 0; i < MAX - 1; i++)
            for (int j = i + 1; j < MAX; j++)
                if (a[i]>a[j])
                    swap(a[i], a[j]);
    }
    void TaoFileKQ(int a[])
    {   //B1: Sắp xếp mảng a tăng dần
        Sort(a);
        //B2: Ghi mảng a đã được sắp tăng dần vào file
        FILE *f;
        f = fopen(OutFile, "wt");
        if (f == NULL)
            printf("Khong tao duoc file");
        else
        {   for (int i = 0; i < MAX ; i++)
                fprintf(f, "%d", a[i]);
            fclose(f);
            printf("Da tao thanh cong file %s", OutFile);
        }
    }
```

(ii)- <u>Thực hiện trên tập tin nhị phân</u>

```
//Khai báo thư viện cần dùng
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#include<conio.h>
//Định nghĩa các biến
#define MAX 10000
#define InFile "D:\\SoNguyen2.inp"
```

```
#define OutFile "D:\\SoNguyen2.out"
//Khai báo các nguyên mẫu hàm
void TaoFile();
void DocFile(int a[]);
void swap(int &a, int &b);
void Sort(int a[]);
void TaoFileKQ(int a[]);

void main()
{
     srand(time(NULL));
     int a[MAX];
     TaoFile();
     DocFile(a);
     TaoFileKQ(a);
}
void TaoFile()
{
     FILE *f;
     f = fopen(OutFile, "wb");
     if (f == NULL)
          printf("Khong tao duoc file");
     else
     {
          for (int i = 0; i < MAX/10; i++)
          {    for (int j = 0; j < 10; j++)
               {    int so = rand()%1000;
                    fwrite(&so, sizeof(int), 1, f);
               }
          }
          fclose(f);
     }
}
void DocFile(int a[])
{
     FILE *f;
     f = fopen(OutFile, "rb");
     if (f == NULL)
          printf("Khong tao duoc file");
     else
     {    for (int i = 0; i < MAX; i++)
               fread(&a[i], sizeof(int), 1, f);
          fclose(f);
     }
}
void swap(int &a, int &b)
```

```
{
  int temp = a;
    a = b;
    b = temp;
}
void Sort(int a[])
{
    for (int i = 0; i < MAX - 1; i++)
      for (int j = i + 1; j < MAX; j++)
          if (a[i]>a[j])
              swap(a[i], a[j]);
}
void TaoFileKQ(int a[])
{
    //B1: Sắp xếp mảng a tăng dần
    Sort(a);
    //B2: Ghi mảng a đã được sắp tăng dần vào file
    FILE *f;
    f = fopen(OutFile, "wb");
    if (f == NULL)
        printf("Khong tao duoc file");
    else
    {   for (int i = 0; i < MAX; i++)
            fwrite(&a[i], sizeof(int), 1, f);
        fclose(f);
        printf("Da tao thanh cong file %s", OutFile);
    }
}
```

1.2. Bài tập 2

- ***Yêu cầu***

Mỗi sinh viên cần quản lý ít nhất 2 thông tin: mã sinh viên và họ tên. Viết chương trình cho phép lựa chọn các chức năng: nhập danh sách sinh viên từ bàn phím rồi ghi lên tập tin SinhVien.dat, đọc dữ liệu từ tập tin SinhVien.dat rồi hiển thị danh sách lên màn hình, tìm kiếm họ tên của một sinh viên nào đó dựa vào mã sinh viên nhập từ bàn phím.

Ta nhận thấy rằng mỗi phần tử của tập tin SinhVien.Dat là một cấu trúc có 2 trường: mã và họ tên. Do đó, ta cần khai báo cấu trúc này và sử dụng các hàm đọc/ghi tập tin nhị phân với kích thước mỗi phần tử của tập tin là chính kích thước cấu trúc đó.

- ***Thực hiện***

```
#include<stdio.h>
#include<conio.h>
```

```
#include<string.h>
typedef struct
{
    char Ma[10];
    char HoTen[40];
} SinhVien;
void WriteFile (char *FileName)
{   FILE *f;
    int n,i;
    SinhVien sv;
    f=fopen(FileName,"ab");
    printf("Nhap bao nhieu sinh vien? ");
    scanf("%d",&n); fflush(stdin);
    for(i=1;i<=n;i++)
    {
        printf("Sinh vien thu %i\n",i);
        printf(" - MSSV: ");
        gets(sv.Ma);
        printf(" - Ho ten: ");
        gets(sv.HoTen);
        fwrite(&sv,sizeof(sv),1,f);
        fflush(stdin);
    }
    fclose(f);
    printf("Bam phim bat ky de tiep tuc");
}
void ReadFile(char *FileName)
{   FILE *f;
    SinhVien sv;
    f=fopen(FileName,"rb");
    printf("   MSSV   | Ho va ten\n");
    fread (&sv,sizeof(sv),1,f);
    while (!feof(f))
    {
        printf(" %s  | %s\n",sv.Ma,sv.HoTen);
        fread(&sv,sizeof(sv),1,f);
    }
    fclose(f);
    printf("Bam phim bat ky de tiep tuc!!!");
}
void Search(char *FileName)
{
    char MSSV[10] ;
    FILE *f ;
    int Found=0;
    SinhVien sv;
```

```
        fflush(stdin);
        printf("Ma so sinh vien can tim: ");
        gets(MSSV);
        f=fopen(FileName,"rb");
        while (!feof(f) && Found==0)
        {
            fread(&sv,sizeof(sv),1,f);
            if (strcmp(sv.Ma,MSSV)==0)
                Found=1;
        }
        fclose(f);
        if (Found == 1)
            printf("Tim thay SV co ma %s. Ho ten la: %s", sv.Ma,
                                                        sv.HoTen);
        else
            printf("Tim khong thay sinh vien co ma %s",MSSV);
        printf("\nBam phim bat ky de tiep tuc!!!");
}
int main()
{   int c;
    for (;;)
    {
        printf("1. Nhap DSSV\n");
        printf("2. In DSSV\n");
        printf("3. Tim kiem\n");
        printf("4. Thoat\n");
        printf("Ban chon 1, 2, 3, 4: ");
        scanf("%d",&c);
        if(c==1)
            WriteFile("d:\\SinhVien.Dat");
        else
            if (c==2)
                ReadFile("d:\\SinhVien.Dat");
            else
                if (c==3)
                    Search("d:\\SinhVien.Dat");
                else
                    break;
    }
    return 0 ;
}
```

4. BÀI TẬP SINH VIÊN TỰ THỰC HIỆN

4. 1. Tập tin văn bản (Text file)

4.1.1. Thao tác trên dữ liệu đơn

(1)- Viết chương trình gồm các hàm sau:

a.- Hàm tạo file có chức năng tạo một file chứa 10000 số nguyên ngẫu nhiên có giá trị từ 1 đến 32767 với tên file là `"SoNguyen.inp"`.

b.- Hàm đọc file `" SoNguyen.inp"` vừa tạo ở trên và xuất ra màn hình trên nhiều dòng, mỗi dòng 10 số.

c.- Hàm đọc file `"SoNguyen.inp"` và ghi các số chẵn vào file `"SoChan.out"` và những số lẻ vào file `"SoLe.out"`.

4.1.2. Mảng 1 chiều

(2)- Cho file `"Array.inp"` chứa dữ liệu của 1 mảng các số nguyên A gồm n phần tử, với cấu trúc của file như sau:

- Dòng 1 chứa số nguyên n (0 < n <= 100).
- Dòng 2 chứa n số nguyên cách nhau bởi khoảng trắng.

a.- Viết hàm đọc file `"Array.inp"` và in n số có trong file này ra màn hình.

b.- Viết hàm đọc file `"Array.inp"` và lưu vào mảng 1 chiều A.

c.- Viết hàm các số nguyên tố có trong mảng A. Yêu cầu: các số nguyên tố được ghi tiếp theo nội dung đã có (không xóa nội dung cũ).

d.- Thực hiện tìm phần tử lớn nhất có trong mảng A, sau đó ghi kết quả sắp xếp vào file `"Array1.out"`, với cấu trúc của file `"Array1.out"` như sau:

- Dòng 1 ghi giá trị lớn nhất có trong mảng.
- Dòng 2 ghi chỉ số (vị trí) của chứa các số lớn nhất trong mảng (do trong mảng có thể chứa nhiều số cùng có giá trị lớn nhất).

e.- Thực hiện sắp xếp các số nguyên trong mảng A tăng dần sau đó ghi kết quả sắp xếp vào file `"Array2.out"`, với cấu trúc của file `"Array2.out"` như sau:

- Dòng 1 chứa n phần tử của mảng các số nguyên.
- Dòng 2 chứa n số nguyên đã được xếp tăng dần.

f.- Thực hiện đếm số lần xuất hiện của từng giá trị có trong mảng A, sau đó ghi kết quả thực hiện vào file `"Array3.out"`, với cấu trúc của file `"Array3.out"` gồm nhiều dòng, mỗi dòng chứa 2 số: giá trị và số lần xuất hiện của giá trị đó.

Ví dụ nội dung tập tin `"Array.inp"` như sau:

```
8
5 4 2 -2 4 5 9 5
```

Nội dung file "Array3.out" sau khi hoàn tất

```
-2   1
2    1
4    2
5    3
9    1
```

(3)- Viết chương trình lần lượt thực hiện các chức năng như sau:

a.- Cho người dùng nhập các số nguyên, việc nhập kết thúc khi người dùng nhập vào giá trị 0. Ghi vào file MANG.TXT có nội dung như sau:

- Dòng đầu chứa số n.
- Dòng kế tiếp chứa các số nguyên mà người dùng đã nhập (không ghi số 0 cuối cùng mà người dùng đã nhập).

Ví dụ nội dung tập tin MANG.TXT như sau:

```
6
5    36   2  -2   4  9
```

b.- Viết hàm đọc file Mang.txt và in giá trị của n số có trong file ra màn hình.

c.- Viết hàm đọc file Mang.txt và thực hiện:

- Đếm xem trong file có bao nhiêu số chính phương?
- In các số chính phương này ra màn hình.
- Tìm số chính phương nhỏ nhất.

d.- Viết hàm đọc file Mang.txt và thực hiện:

- Đếm xem trong file có bao nhiêu số nguyên tố palindrome?
- In các số nguyên tố palindrome này ra màn hình.
- Tìm số nguyên tố palindrome lớn nhất.

🏳 *Số nguyên tố Palindrome*: Là số nguyên tố vẫn giữ nguyên giá trị khi các chữ số của nó được đảo ngược. Ví dụ: 2, 3, 5, 7, 11, 101, 131, 151, 181, 191, 10301, 11311, 12421.

e.- Viết hàm đọc file Mang.txt và thực hiện:

- Đếm xem trong file có bao nhiêu số nguyên tố strobogrammatic?
- In các số nguyên tố strobogrammatic này ra màn hình.
- Tìm số nguyên tố strobogrammatic lớn nhất.

🏳 *Số strobogrammatic*: Là một số có giá trị không đổi khi xoay số đó 180 độ (180^o). Nhận xét: các số *strobogrammatic* chỉ chứa các số sau đây: 0, 1, 6, 8, 9. Ví dụ: 0, 1, 8, 11, 69, 88, 96, 101, 111, ...

(4)- Viết chương trình tạo một tập tin chứa các số nguyên có giá trị ngẫu nhiên không trùng nhau. Sắp xếp chúng theo thứ tự tăng dần và lưu trữ sang tập tin khác.

(5)- Viết chương trình thực hiện các yêu cầu sau:

a. Cho nhập n (1<n<=500), nếu sai yêu cầu nhập lại

b. Tạo file input.txt chứa n số ngẫu nhiên, với giá trị các số ngẫu nhiên nằm trong khoảng từ -1000 đến +1000. Cấu trúc file input.txt gồm 2 dòng: dòng đầu chứa số n, dòng kế tiếp chứa n số ngẫu nhiên đã tạo được.

c. Viết hàm đọc file input.txt. Tìm xem các số may mắn có trong file này và ghi kết quả vào file LuckyNumber.txt. Cấu trúc file LuckyNumber.txt gồm 2 dòng: dòng đầu chứa số lượng số may mắn, dòng kế tiếp chứa các số may mắn có trong file input.txt.

☞ Số may mắn có thể là số âm hoặc dương và các số này chỉ chứa các ký số 6 hoặc ký số 8. Ví dụ số may mắn như 6, - 86, 68, -888, ...; số không may mắn như 7, - 96, 65, -848, ...

4.1.3. Mảng 2 chiều

(6)- Cho file *Mang2Chieu.txt* có cấu trúc như sau: dòng đầu lưu giá trị của 2 số nguyên dương m, n. Các dòng còn lại lưu giá trị của 1 mảng hai chiều m dòng và n cột là các số nguyên.

a. Viết chương trình đọc file mảng hai chiều trên.

b. Xuất mảng hai chiều đó ra màn hình.

c. Tính tổng các số chẵn có trong mảng hai chiều.

d. Tìm phần tử lớn nhất có trong mảng hai chiều.

e. Đếm số phần tử lớn nhất có trong mảng hai chiều.

f. Sắp xếp mảng tăng dần.

g. Nếu mảng 2 chiều là ma trận vuông thì in các giá trị trên đường chéo chính và đường chéo phụ nối tiếp vào nội dung file *Mang2Chieu.txt* (ghi tiếp theo, không xóa nội dung cũ).

(7)- Viết chương trình tạo mới 1 file chứa dữ liệu của một ma trận 10 x10 với các phần tử của ma trận là số nguyên (0-100) được tạo ngẫu nhiên. Sau khi đóng file vừa tạo, mở lại file, đọc và in ma trận ra màn hình.

(8)- Viết chương trình gồm 2 hàm sau:

- Hàm TaoFile:
  - Nhận tham số là số dòng, số cột cần có của 1 mảng hai chiều.
  - File cần tạo có tên MaTran.txt. Cấu trúc của file có dạng như sau:
    - Dòng 1: chứa số dòng của mảng hai chiều
    - Dòng 2: chứa số cột của mảng hai chiều
    - Dòng 3 đến dòng cuối: mỗi dòng chứa giá trị các phần tử có trên từng dòng của mảng hai chiều. Biết rằng giá trị các phần tử này được phát sinh ngẫu nhiên và có giá trị trong khoảng từ 1-100.
- Hàm XoaDong:
  - Nhận tham số là số dòng (row) cần xóa.

- Thực hiện đọc file MaTran.txt. Tiến hành xóa dòng row. Lưu kết quả lại vào file MaTran.txt.

(9)- Viết chương trình tạo file văn bản có tên là "MATRIX.INP" có cấu trúc như sau:

- Dòng đầu ghi hai số *m*, *n*.
- Trong m dòng tiếp theo mỗi dòng ghi n số và các số các nhau một khoảng cách.

Hãy kiểm tra xem trong file đó có bao nhiêu số nguyên tố.

Kết quả cần ghi vào file "*MATRIX.OUT*" có nội dung là một số nguyên đó là số lượng các số nguyên tố trong file "*MATRIX.INP*".

(10)- Nhập một ma trận vuông nxn từ một tập tin văn bản MATRAN.TXT có nội dung như sau:

- Dòng đầu tiên chứa cấp n
- Mỗi dòng kế tiếp chứa n số tương ứng với mỗi dòng của ma trận.

Ví dụ nội dung tập tin *MATRAN.TXT* như sau:

```
3
5 8   2
4 7   6
8 0   0
```

Yêu cầu:

a. Đọc dữ liệu từ tập tin vào một mảng trong bộ nhớ.
b. Tính tổng đường chéo chính.
c. Tính tổng đường chéo phụ.
d. Sắp xếp các phần tử trên đường chéo chính tăng dần từ trên xuống dưới.
e. Sắp xếp các phần tử trên đường chéo phụ giảm dần từ trên xuống dưới.

(11)- Viết chương trình thực hiện các yêu cầu sau:

a. Cho nhập n (1 <*n* <=20), nếu sai yêu cầu nhập lại.

```
3
3   5   9
2   4   7
8   1   6
```
input.txt

```
3
          9
    4
8
```
CheoPhu.out

```
 3   5   9  17
 2   4   7  13
 8   1   6  15
13  10  21
```
TongDongCot.out

b. Tạo file *input.txt* chứa *n* X *n* số ngẫu nhiên, với giá trị các số ngẫu nhiên đều là số dương (nằm trong khoảng từ *0* đến *400*) và không trùng nhau. Cấu trúc file *input.txt* gồm *n+1* dòng: dòng đầu chứa số *n*, *n* dòng kế tiếp mỗi dòng chứa *n* số ngẫu nhiên đã tạo được.

c. Viết hàm đọc file *input.txt*. Tạo file *CheoPhu.out* như hình minh họa. Cấu trúc file *CheoPhu.out* gồm *n+1* dòng: dòng đầu chứa số *n*, *n* dòng kế tiếp mỗi dòng chứa số nằm trên đường chéo phụ của ma trận *n* X *n*, những số không nằm trên đường chéo phụ sẽ được in bằng khoảng trắng.

d. Viết hàm đọc file *input.txt*. Thực hiện tính tổng các dòng, tổng các cột, sau đó lưu kết quả vào file *TongDongCot.out* như hình minh họa.

4. 2. Tập tin nhị phân (Binary file)

(12)- Tạo sẵn file chứa nội dung tùy ý. Viết hàm tìm và thay thế một chuỗi con bằng một chuỗi khác.

(13)- Viết chương trình quản lý một tập tin văn bản theo các yêu cầu:

a. Nhập từ bàn phím nội dung một văn bản sau đó ghi vào đĩa

b. Đọc từ đĩa nội dung văn bản vừa nhập và in lên màn hình.

c. Đọc từ đĩa nội dung văn bản vừa nhập, in nội dung đó lên màn hình và cho phép nối thêm thông tin vào cuối tập tin đó.

(14)- Viết chương trình nhập tên một tập tin văn bản và thống kê tần số xuất hiện của một chữ cái từ A đến Z trong văn bản. Khi đếm tần số xuất hiện, không phân biệt chữ thường với chữ hoa (chẳng hạn a và A là như nhau). Kết quả thống kê ghi vào một tập tin văn bản trên đĩa, tần số xuất hiện của mỗi chữ cái được lưu trên một dòng của tập tin văn bản.

(15)- Viết chương trình đếm số từ và số dòng trong một tập tin văn bản.

(16)- Viết chương trình đọc một chuỗi tối đa 100 kí tự từ bàn phím. Lưu các ký tự là nguyên âm vào tập tin "NguyenAm.txt". Đọc các kí tự từ tập tin này và hiển thị lên màn hình console.

(17)- Viết chương trình đọc và hiện một tập tin văn bản. Sau đó trình bày những thống kê sau: số ký tự, số từ, số dòng của nó.

(18)- Viết chương trình tạo ra một tập văn bản chứa tên, tuổi, địa chỉ (mỗi thông tin chiếm một dòng). Sau đó chương trình sẽ đọc lại tập tin này và chép sang một tập tin khác nhưng trên một dòng cho mỗi người.

(19)- Viết chương trình tính số lần xuất hiện ký tự chữ cái trong một tập tin văn bản.

(20)- Viết chương trình tính số từ có trong một tập tin văn bản.

(21)- Viết chương trình nối hai tập tin văn bản lại với nhau thành một tập tin mới.

(22)- Viết chương trình lưu lại nội dung của màn hình vào tập tin SCREEN.DAT (tương tự lệnh COPY CON của DOS).

(23)- Viết chương trình hiện chương trình được lưu trong SCREEN.DAT (tương tự lệnh TYPE của DOS).

(24)- Tương tự bài trên nhưng những dòng trống sẽ được bỏ qua.

(25)- Viết chương trình cho phép nhập từ bàn phím và ghi vào 1 tập tin tên HocSinh.txt với mỗi phần tử của tập tin là một cấu trúc bao gồm các trường:

- Mã số học sinh (Ma, char [3]).
- Họ tên học sinh (HoTen, char[30]).
- Điểm lý thuyết (lt, float).
- Điểm thực hành (th, float).
- Điểm trung bình (dtb float). Biết dtb= (lt*60%)+(th*40%)

Yêu cầu cài đặt:

- Viết hàm cho nhập số nguyên dương n.
- Viết hàm cho nhập thông tin của n học sinh và ghi vào file HocSinh.txt.
- Viết hàm in ra danh sách các học sinh có điểm trung bình dưới 5.
- Viết hàm xuất các học sinh có trong file HocSinh.txt ra màn hình theo thứ tự giảm dần của điểm trung bình.

(26)- Để quản lý được hàng hóa, người ta cần biết được các thông tin sau về hàng hóa:

- Mã hàng (mh, char[5]).
- Số lượng (sl, int).
- Đơn giá (dg, float).
- Số tiền (st, float). Biết số tiền = số lượng * đơn giá

Yêu cầu cài đặt:

- Viết hàm cho nhập các hàng hóa cần ghi vào file DSHH.TXT. Kết thúc việc nhập bằng cách đánh ENTER vào mã hàng. Sau khi nhập xong yêu cầu in toàn bộ danh sách hàng hóa ra màn hình.
- Viết hàm xuất các hàng hóa có trong file DSHH.TXT ra màn hình.
- Viết hàm nhận tham số là mã hàng hóa (mh) cần tìm. Nếu tìm thấy mã mh sẽ in thông tin của hàng hóa có mã mh ra màn hình; ngược lại in ra "Không tìm thấy.