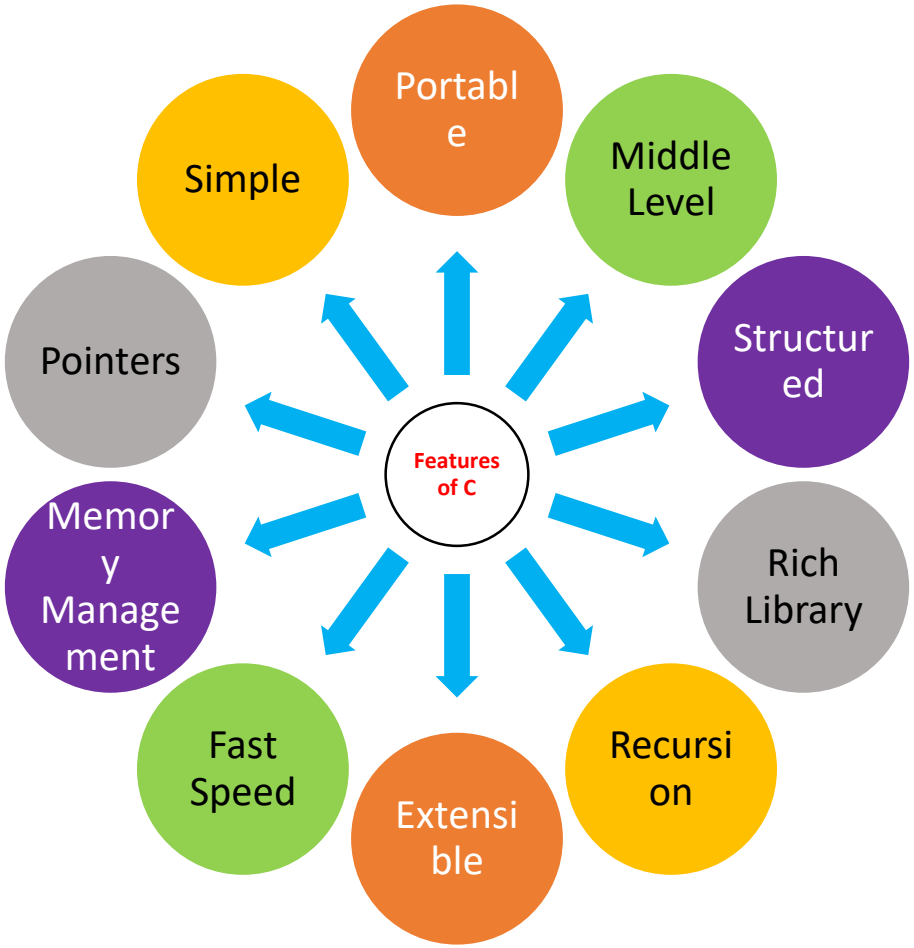
PHƯƠNG PHÁP LẬP TRÌNH

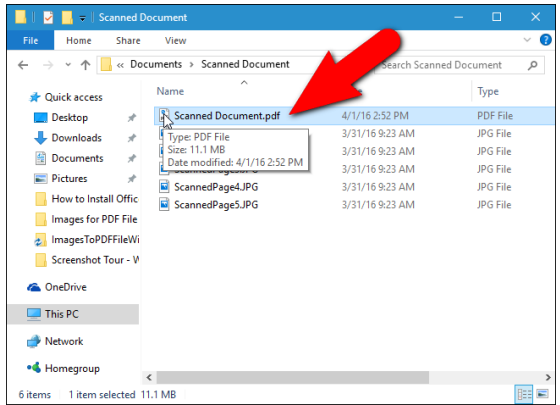


NỘI DUNG MÔN HỌC

1. Tổng quan về giải thuật
2. Ngôn ngữ lập trình C
3. Cấu trúc điều khiển (*Control structures and statements*)
4. Hàm (*Function*)
5. Mảng 1 chiều (*One array dimension*)
6. Chuỗi ký tự (*String*)
7. Mảng hai chiều (*Two array dimension*)
8. Kiểu dữ liệu cấu trúc (*Struct data type*)
9. Kiểu dữ liệu con trỏ (*Pointer data type*)
10. Tập tin (*File*)
11. Đệ quy (Recursive)

CHƯƠNG 10

KIỂU DỮ LIỆU TẬP TIN (*FILE*)



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.

NỘI DUNG

1. Khái niệm
2. Tổng quan khi thao tác với tập tin
3. Các thao tác trên tập tin
4. Truy cập tập tin văn bản
5. Truy cập tập tin nhị phân
6. Các hàm thao tác chung trên tập tin văn bản & Tập tin nhị phân
7. Thực hành

1. KHÁI NIỆM

1.1. Tại sao phải sử dụng tập tin?

- *Dữ liệu lưu trữ trong bộ nhớ của máy tính (RAM)*:
  - Thông thường: nhập dữ liệu – biến … (gồm kiểu số, kiểu mảng, kiểu cấu trúc, …) từ bàn phím ⇒ thao tác ⇒ xuất ra màn hình. Dữ liệu được lưu trữ trên RAM (bộ nhớ lưu trữ tạm thời).
  - ***Ưu điểm***: xử lý trên RAM có tốc độ cao do tốc độ truyền dữ liệu cao.
  - ***Khuyết điểm***
    - RAM giá thành đắt
    - Không lưu trữ dài hạn dữ liệu để sử dụng cho các lần sau:
      - Sau khi kết thúc chương trình
      - mất điện sẽ mất dữ liệu, …

1.1. Tại sao phải sử dụng tập tin?

- *Dữ liệu lưu trữ trong bộ nhớ ngoài của máy tính (FDD, HDD, USB, ...).*
  - *Đặc điểm*:
    - Khi kết thúc chương trình, dữ liệu vẫn còn $\Rightarrow$ có thể sử dụng nhiều lần.
    - Kích thước của kiểu tập tin là lớn với số lượng các phần tử không hạn chế (chỉ bị hạn chế bởi dung lượng của bộ nhớ ngoài).

1.2. Khái niệm về tập tin

- Tập hợp thông tin (dữ liệu) được tổ chức theo một dạng xác định với tên được định danh
- Một dãy byte liên tục (dưới góc độ lưu trữ)
- Được lưu trữ trong thiết bị lưu trữ ngoài: USB, HDD, SSD, …
- Cho phép đọc dữ liệu (thiết bị nhập) và ghi dữ liệu (thiết bị xuất).

1.3. Phân loại kiểu tập tin

– ***Mục đích trong sử dụng***: quan tâm đến nội dung tập tin sẽ phân loại theo phần mở rộng tập tin (đuôi tập tin): *.EXE*, *.DOCX*, *.TXT*, *.PPTX*, …

– ***Mục đích trong lập trình***: tự tạo các stream tường minh để kết nối với tập tin xác định nên sẽ phân loại theo cách sử dụng stream.

– ***Hai dạng tập tin cơ bản***

- Tập tin dạng văn bản (tương ứng với *stream văn bản*)
- Tập tin dạng nhị phân (tương ứng với *stream nhị phân*).

1.3. Phân loại kiểu tập tin

1.3.1. Tập tin văn bản (Text file)

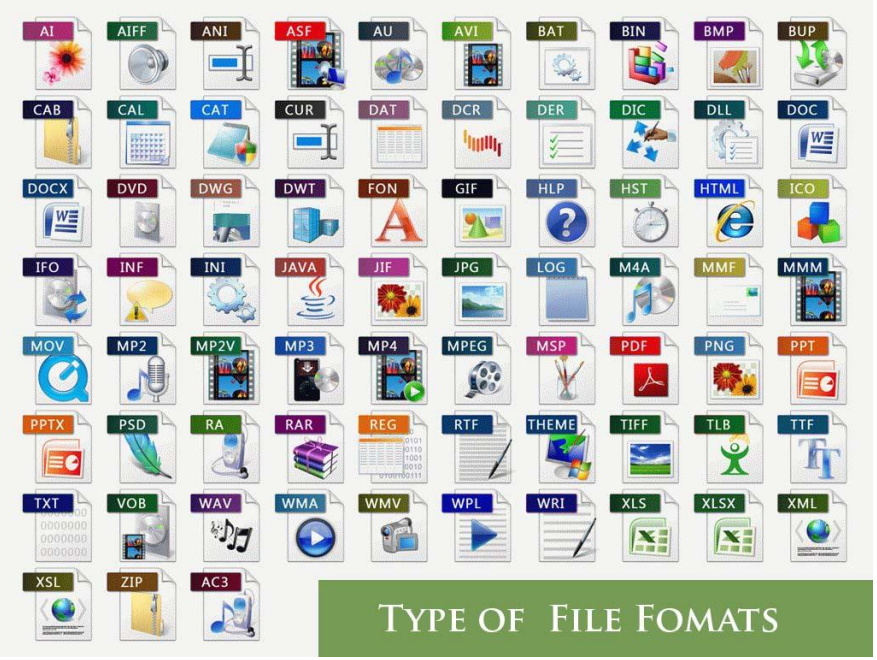
- Là loại tập tin dùng để ghi các ký tự lên đĩa, các ký tự này được lưu trữ dưới dạng mã ***ASCII***.
- Dữ liệu của tập tin được lưu trữ thành các dòng, mỗi dòng được kết thúc bằng ký hiệu xuống dòng (*new line*).
- Ký hiệu xuống dòng là sự kết hợp của 2 ký tự:
  - Ký tự '**\r**' hay ***CR*** (***Carriage Return*** - về đầu dòng, mã ***ASCII***=*13*)
  - Ký tự '**\n**' hay ***LF*** (***Line Feed*** - xuống dòng, mã ***ASCII***=*10*).
- Tập tin được kết thúc bởi ký tự ***EOF*** (***End Of File***) có mã ***ASCII***=26 (tổ hợp phím ***Ctrl+Z***).
- Tập tin văn bản chỉ có thể truy xuất theo kiểu tuần tự.
- Ví dụ: Các tập tin: *.cpp, *.txt, ... là các tập tin văn bản

1.3. Phân loại kiểu tập tin

1.3.2. Tập tin nhị phân (Tập tin có kiểu -Typed File)

- Là loại tập tin bao gồm nhiều phần tử có cùng kiểu: char, int, long, struct, …
- Được lưu trữ trên đĩa dưới dạng một chuỗi các byte liên tục.
- Mỗi tập tin nhị phân do một phần mềm nào tạo ra thì sẽ có cấu trúc do phần mềm đó quy định.

Ví dụ: Các tập tin: *.exe, *.doc, *.xls, *.bmp, ... là các tập tin nhị phân.



1.3. Phân loại kiểu tập tin

1.3.3. Tập tin không có kiểu (Untyped File)

- Là loại tập tin mà dữ liệu của chúng gồm các cấu trúc dữ liệu mà người ta không quan tâm đến nội dung hoặc kiểu của nó.
- Chỉ lưu ý đến các yếu tố vật lý của tập tin như độ lớn và các yếu tố tác động lên tập tin mà thôi.

2. TỔNG QUAN KHI THAO TÁC VỚI TẬP TIN

2.1. Tên tập tin

- **Quy tắc đặt tên**

  <Tên tập tin> . <Phần mở rộng>

- **Phần tên tập tin**
  - Bắt buộc phải có
  - Chiều dài tối đa 128 ký tự
  - Gồm các ký tự từ A đến Z, a đến z, số 0 đến 9, khoảng trắng, các ký tự @ # $ % ^ ( ) !

- **Phần mở rộng tập tin**
  - Không bắt buộc
  - Thông thường 3 – 4 ký tự (chữ và số)

2.2. Đường dẫn *(Path)*

- Là địa chỉ chỉ đến một tập tin hiện hành trên ổ cứng.
- Ví dụ: *c:\data\list.txt* chỉ tập tin list.txt nằm trong ổ cứng C có thư mục con là data
- Trong chương trình C, đường dẫn trên được ghi dưới dạng như sau "*c:\\data\\list.txt*"

📖 **Tại sao phải viết đường dẫn dưới dạng 2 dấu *backslash* (\\)?**

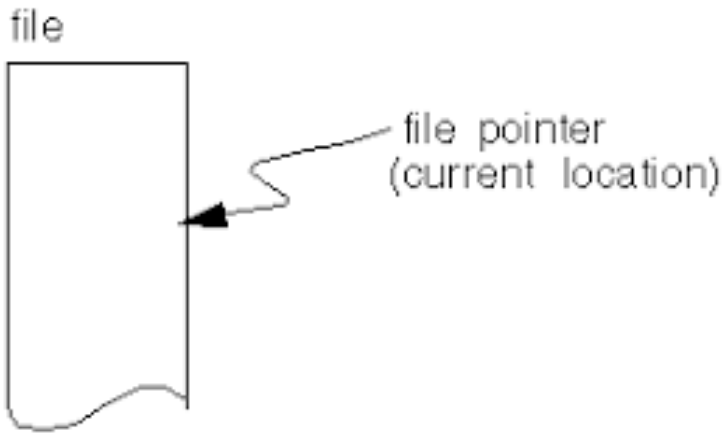
- Vì dấu '\' là một ký tự giúp biểu diễn cho một số ký tự đặc biệt trong C, nên để biểu diễn thì phải thêm một dấu '\' thành "\\" ở trước để ký hiệu.
- Nếu nhập đường dẫn từ bàn phím thì không cần thêm dấu '\'.

2.3. Biến kiểu tập tin

- Dùng để đại diện cho một tập tin.
- Dữ liệu chứa trong một tập tin được truy xuất qua các thao tác với thông số là biến tập tin đại diện cho tập tin đó.

2.4. Con trỏ tập tin *(file pointer)*

- Khi một tập tin được mở ra để làm việc, tại mỗi thời điểm, sẽ có một vị trí của tập tin mà tại đó việc đọc/ghi thông tin sẽ xảy ra. Người ta hình dung có một con trỏ đang chỉ đến vị trí đó và đặt tên nó là con trỏ tập tin.
- Sau khi đọc/ghi xong dữ liệu, con trỏ sẽ chuyển dịch thêm một phần tử (số byte đúng bằng số byte đọc/ghi trước đó) về phía cuối tập tin. Sau phần tử dữ liệu cuối cùng của tập tin là dấu kết thúc tập tin ***EOF*** (***End Of File***).



2.5. Vùng đệm *(buffer)*

- Trong C, tập tin được gắn liền với một vùng đệm. Mỗi thao tác đọc/ghi tập tin thường được tiến hành trên vùng đệm chứ không hẳn trên tập tin.
- Khi ghi, dữ liệu từ biến nhớ được đưa vào vùng đệm và khi nào vùng đệm đầy thì vùng đệm mới được đẩy lên đĩa.
- Khi đọc, dữ liệu được lấy từ vùng đệm đưa vào biến nhớ và chỉ khi nào vùng đệm đã trống rỗng thì máy mới lấy dữ liệu từ đĩa đưa vào vùng đệm.

3. CÁC THAO TÁC TRÊN TẬP TIN

Các bước thao tác trên tập tin.

B1: Khai báo biến tập tin

B2: Mở tập tin bằng hàm fopen()

B3: Thực hiện các thao tác xử lý dữ liệu của tập tin bằng các hàm đọc/ghi dữ liệu

B4: Đóng tập tin bằng hàm fclose()

3.1. 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*: `FILE *f1,*f2;`

3.2. 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.

Chế độ	*Ý nghĩa*
r+	Mở một tập tin văn bản để đọc/ghi. Tập tin cần tồn tại nếu không sẽ xuất hiện lỗi.
r+t	
wt	Tạo ra tập tin văn bản mới để ghi. Nếu tập tin đã tồn tại ⇒ nội dung cũ của tập tin sẽ bị xóa
w+	
w+t	
at	Mở một tập tin văn bản để ghi thêm. Nếu tập tin chưa tồn tại thì sẽ tạo tập tin mới.
a+t	Mở một tập tin văn bản để đọc/ghi thêm. Nếu tập tin chưa tồn tại thì sẽ tạo tập tin mới
rb	Mở ra tập tin nhị phân để đọc/ghi. Tập tin cần tồn tại nếu không sẽ xuất hiện lỗi.
r+b	
wb	Tạo ra tập tin nhị phân mới để đọc/ghi. Nếu tập tin đã tồn tại, nội dung cũ của tập tin sẽ bị xóa
w+b	
ab	Mở một tập tin nhị phân để đọc/ghi thêm. Nếu tập tin chưa tồn tại thì sẽ tạo tập tin mới
a+	
a+b	

3.2. Mở tập tin

- *Cú pháp*:

```
FILE *fopen(char *Path, const char *Mode)
```

  - Nếu mở file thành công, 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.

Trong đó:

- **`Path`**: chuỗi chỉ đường dẫn đến tập tin trên đĩa.
- **`Type`**: chuỗi xác định cách thức mà tập tin sẽ mở.

3.2. Mở tập tin

- *Ví dụ*: Mở một tập tin tên TEST.txt để ghi.

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

3.3. Đóng tập tin

- Đóng 1 tập tin
  - 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 tập tin lại.
  - *Cú pháp*: **`int fclose(FILE *f)`**
    - f là con trỏ tập tin được mở bởi hàm **`fopen()`**.
  - Giá trị trả về của hàm:
    - *Thành công*: **`0`**
    - *Có lỗi*: **`EOF`** (hay **`-1`**)
- Đóng tất cả các tập tin đang mở
  - *Cú pháp*: **`int fcloseall()`**
  - Giá trị trả về của hàm:
    - *Thành công*: số nguyên, cho biết tổng số các tập tin đã được đóng
    - *Có lỗi*: **`EOF`**

3.4. Kiểm tra đến cuối tập tin

- *Cú pháp*: `int feof(FILE *f)`
- *Ý nghĩa*: Kiểm tra xem đã chạm tới cuối tập tin hay chưa.
- *Kết quả trả về của hàm:*
  - EOF: nếu con trỏ tập tin **đã** chạm tới cuối tập tin
  - 0: nếu con trỏ tập tin **chưa** chạm tới cuối tập tin.

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

- *Cú pháp*: `void rewind(FILE *f)`
- *Ý nghĩa*: Khi đang thao tác một tập tin đang mở, con trỏ tập tin luôn di chuyển về phía cuối tập tin. Hàm `rewind()` giúp đưa con trỏ tập tin quay về đầu tập tin như khi mới mở.

3.6. Làm sạch vùng đệm của tập tin

- *Cú pháp*: `int fflush(FILE *f)`
- *Ý nghĩa*: Hàm dùng làm sạch vùng đệm của tập tin.
- *Kết quả trả về của hàm:*
  - Thành công: ***0***
  - Thất bại: ***EOF***

4. TRUY CẬP TẬP TIN VĂN BẢN

4.1. Hàm fputs()

- Dùng để ghi một chuỗi ký tự chứa trong vùng đệm lên tập tin văn bản. Ký tự '`\0`' sẽ không được ghi lên tập tin.
- ***Cú pháp***

```
int puts(const char *buffer, FILE *f)
```

- ***Giải thích***
  - **`buffer`** là con trỏ có kiểu char chỉ đến vị trí đầu tiên của chuỗi ký tự được ghi vào.
- ***Kết quả trả về của hàm***
  - Thành công: hàm trả về giá trị không âm
  - Thất bại: hàm trả về EOF.

4.2. Hàm fputs()

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

```
#include<stdio.h>
#include<conio.h>
void main()
{
    FILE *f;
    f=fopen("D:\\TinhTP.txt",“wt");
    if (f!=NULL)
    {
        fputs(“Sai gon.\n",f);
        fputs(“TP.Ho Chi Minh",f);
        fclose(f);
    }
    else
        printf(“Khong tao duoc file”);
    getch();
}
```

4.3. Hàm fgets()

- Đọc một dãy ký tự từ tập tin và lưu vào chuỗi.
- ***Cú pháp***

```
char *fgets(char *s, int n, FILE *fp);
```

- ***Giải thích:*** Việc đọc kết thúc khi
  - Đã được đọc n-1 ký tự
  - hoặc gặp dấu xuống dòng
  - hoặc gặp dấu kết thúc tập tin
- ***Kết quả trả về của hàm***
  - Thành công: hàm trả về con trỏ tới chuỗi s
  - Thất bại hoặc gặp dấu kết thúc tập tin: hàm trả về NULL.

4.4. Hàm fprintf()

- Dùng để ghi dữ liệu có định dạng lên tập tin văn bản theo khuôn dạng xác định trong chuỗi định dạng.
- *Cú pháp*

```
fprintf(FILE *f, const char *format, varexpr)
```

- *Giải thích*
  - `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 (,).
- *Kết quả trả về của hàm*
  - ***Thành công***: hàm trả về một giá trị nguyên bằng số byte ghi lên tập tin
  - ***Thất bại***: hàm trả về giá trị âm.

4.5. Hàm fscanf()

- Đọc dữ liệu từ biến kiểu con trỏ file, biến đổi theo khuôn dạng trong chuỗi định dạng và lưu kết quả vào danh sách các tham số.
- ***Cú pháp***

```
int fscanf(FILE *fp, format [,params]);
```

- ***Giải thích***
  - `format`: chuỗi định dạng (giống với các định dạng của hàm `printf()`),
  - `params`: danh sách các tham số.
- ***Kết quả trả về của hàm***
  - ***Thành công***: hàm trả về một giá trị nguyên bằng số byte bằng số trường đọc được.
  - ***Thất bại***: hàm trả về giá trị âm.

4.6. Ví dụ 1: thực hiện các yêu cầu sau

i. Tạo một tập tin văn bản có tên là "*Tho.txt*" nằm trong thư mực hiện hành có nội dung như sau:

Quê hương là chùm khế ngọt
Cho con trèo hái mỗi ngày

ii. Đọc từng dòng của tập tin văn bản "*Tho.txt*" mới vừa tạo ở trên và xuất ra màn hình.

iii. Ghi thêm vào tập tin văn bản "Tho.txt" mới vừa tạo ở trên hai câu thơ được nhập từ bàn phím có nội dung sau:

Quê hương là đường đi học
Con về rợp bướm vàng bay

4.6. Ví dụ 1: thực hiện các yêu cầu sau

i. Tạo một tập tin văn bản có tên là "*Tho.txt*" nằm trong thư mục hiện hành có nội dung như sau:

Quê hương là chùm khế ngọt
Cho con trèo hái mỗi ngày

```
int TaoTT(char *tentt)
{
    FILE *fp; int ret;
    fp = fopen(tentt, "wt");
    if(fp != NULL)
    {
        fprintf(fp, "Quê hương là chùm khế ngọt \n");
        fprintf(fp, "Cho con trèo hái mỗi ngày \n");
        fclose(fp);
        ret = 1; //Thành công
    }
    else
        ret = 0; //Thất bại
    return ret;
}
```

4.6. Ví dụ 1: thực hiện các yêu cầu sau

ii. Đọc từng dòng của tập tin văn bản "*Tho.txt*" mới vừa tạo ở trên và xuất ra màn hình.

```
int DocTT(char *tentt)
{
   FILE *fp; char s[MAX_LEN]; int ret;
   fp = fopen(tentt, "rt");
   if(fp != NULL)
   {
      while(!feof(fp))
      {     if(fgets(s, MAX_LEN, fp) != NULL)
               printf("%s", s);
      }
      fclose(fp);
      ret = 1;
   }
   else
      ret = 0;
   return ret;
}
```

4. Truy cập tập tin văn bản

4.6. Ví dụ 1: thực hiện các yêu cầu sau

iii. Ghi thêm vào tập tin văn bản "Tho.txt" mới vừa tạo ở trên hai câu thơ được nhập từ bàn phím có nội dung sau:

Quê hương là đường đi học
Con về rợp bướm vàng bay

```
int GhiThemTT(char *tentt)
{
    FILE *fp; int ret, i; char s[MAX_LEN];
    fp = fopen(tentt, "at");
    if(fp != NULL)
    {   printf("Nhập dữ liệu muốn thêm:\n");
        for(i = 0; i < 2; i++)
        {   gets(s);
            fprintf(fp, "%s", s);
        }
        fclose(fp);
        ret = 1;
    }
    else
      ret = 0;
    return ret;
}
```

4.6. Ví dụ 1: thực hiện các yêu cầu sau

```
#define MAX_LEN 256
int TaoTT(char *tentt);
int DocTT(char *tentt);
int GhiThemTT(char *tentt);
void main()
{
      char *S= “Tho.txt”;
      if (TaoTT(S) == 0)
            printf(“Không tạo được tập tin\n”);
      if (DocTT(S) == 0)
            printf(“Không tìm thấy tập tin cần đọc”);
      if(GhiThemTT(S) == 1)
      {     printf(“Noi dung tap tin sau khi ghi them:\n”);
            DocTT(“Tho.txt”);
      }
      else
            printf(“Tập tin không có trên đĩa\n”);
}
```

4.7. Ví dụ 2: Viết chương trình cho nhập vào tên hai tập tin văn bản. Hãy ghép nội dung tập tin thứ hai nối vào cuối tập tin thứ nhất.

```
void main()
{
    char tentt1[MAX_LEN], tentt2[MAX_LEN];
    printf("Nhap ten tap tin thu 1:");
    gets(tentt1);
    printf("Nhap ten tap tin thu 2:");
    gets(tentt2);
    int kq = GhepTT(tentt1, tentt2);
    if(kq == -2)
        printf("Tap tin %s và %s khong co tren dia\n", tentt1,
        tentt2);
    else
        if(kq == -1)
            printf("Tap tin %s khong co tren dia\n", tentt1);
        else
        if(kq == 0)
            printf("Tap tin %s khong co tren dia\n", tentt2);
        else
        {   printf("Tap tin %s sau khi ghep\n", tentt1);
            DocTT(tentt1);
        }
 }
```

4.7. Ví dụ 2: Viết chương trình cho nhập vào tên hai tập tin văn bản. Hãy ghép nội dung tập tin thứ hai nối vào cuối tập tin thứ nhất.

```
int GhepTT(char *tentt1, char *tentt2)
{       FILE* fp1, *fp2; char s[MAX_LEN]; int ret;
        fp1 = fopen(tentt1, “at”); fp2 = fopen(tentt2, “rt”);
        if(fp1 == NULL && fp2 == NULL)
                ret = -2;
        else
                if(fp1 == NULL)
                        fclose(fp2); ret = -1;
                else
                        if(fp2 == NULL)
                        {       fclose(fp1);
                                ret = 0;
                        }
                        else
                        {       while(!feof(fp2))
                                        if(fgets(s, MAX_LEN, fp2) != NULL)
                                                fprintf(fp1, “%s”, s);
                                fclose(fp1);
                                fclose(fp2);
                                ret = 1;
                        }
        return ret;
}
```

4.8. Ví dụ 3: Viết chương trình thực hiện các yêu cầu sau:

- Viết hàm tạo file có tên MaTran.txt tại thư mục gốc dĩa D. Trong hàm thực hiện cho người dùng nhập số dòng, số cột của ma trận, sau đó hàm ghi file với dòng 1 chứa số dòng (*row*), dòng 2 chứa số cột (*col*), từ dòng 3 trở đi mỗi dòng chứa các số của các dòng trong ma trận. Biết rằng, các số được tạo giá trị ngẫu nhiên <100 và được đặt cách nhau bởi ký tự khoảng trắng.
- Viết hàm đọc dữ liệu từ file MaTran.txt đưa vào ma trận A.
- In ra giá trị lớn nhất trên mỗi dòng

```
3
4
5  7  1  9
6  2  8  0
3  3  9  4
```

Minh họa
nội dung file
MaTran.txt

```
3
4
5  7  1  9        9
6  2  8  0        8
3  3  9  4        9
```

Minh họa kết quả
In giá trị lớn nhất của từng dòng
trong ma trận được đọc từ file
MaTran.txt

4.8. Ví dụ 3: (tt)

```
bool TaoFileMang2Chieu(char *FileName, int row, int col)
{
     FILE *f;
     f = fopen(FileName, "wt");
     if (f == NULL)
          return false;
     fprintf(f, "%d\n", row);
     fprintf(f, "%d\n", col);
     for (int i = 0; i < row; i++)
     {
          for (int j = 0; j < col; j++)
               fprintf(f, "%d\t", rand() % 100);
          fprintf(f, "\n");
     }
     fclose(f);
     return true;
}
```

4.8. Ví dụ 3: (tt)

```
bool DocFileMang2Chieu(char *TenFile, int A[][SIZE],
                                      int &row, int &col)
{
    FILE *f;
    int i, j, temp;
    f = fopen(TenFile, "rt");
    if (f == NULL)
        return false;
    // đọc số dòng
    fscanf(f, "%d", &row);
    // đọc số cột
    fscanf(f, "%d", &col);
    for (i = 0; i<row; i++)
        for (j = 0; j<col; j++)
            fscanf(f, "%d", &A[i][j]);
    fclose(f);
    return true;
}
```

4.8. Ví dụ 3: (tt)

```
void XuatMang2Chieu(int A[][MAX], int row, int col)
{
     for (int i = 0; i < row; i++)
     {
          for (int j = 0; j < col; j++)
               printf("%5d", A[i][j]);
          printf("\n");
     }
}
```

4.8. Ví dụ 3: (tt)

```
int TimSoLonNhatTrenDong(int A[][MAX], int row, int col)
{
     int max = A[row][0];
     for (int k = 1; k < col; k++)
          if (max < A[row][k])
               max = A[row][k];
     return max;
}
void InSoLonNhatTrenMoiDong(int A[][MAX], int row,
                                                  int col)
{
   for (int i = 0; i < row; i++)
   {
     for (int j = 0; j < col; j++)
          printf("%5d", A[i][j]);
     printf("\t\t%5d", TimSoLonNhatTrenDong(A, i, col));
     printf("\n");
   }
}
```

4. Truy cập tập tin văn bản

4.8. Ví dụ 3: (tt)

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <string.h>
#define SIZE 100
bool TaoFileMang2Chieu(char *FileName, int row, int col);
bool DocFileMang2Chieu(char *TenFile, int A[][SIZE], int &row, int &col);
void XuatMang2Chieu(int A[][SIZE], int row, int col);
int main()
{
	int row, col, A[SIZE][SIZE];
	char* TenFile = "D:\\MaTran.txt";
	printf("Nhap so dong: ");
	scanf("%d", &row);
	printf("Nhap so cot: ");
	scanf("%d", &col);
	if (TaoFileMang2Chieu(TenFile, row, col)==false)
		printf("Khong mo duoc file");
	else
	{
		DocFileMang2Chieu(TenFile,A,row,col);
		printf("Ma tran doc duoc tu file:\n");
		XuatMang2Chieu(A,row,col);
	}
	return 0;
}
```

5. TRUY CẬP TẬP TIN NHỊ PHÂN

5.1. Ghi dữ liệu lên tập tin nhị phân

- *Cú pháp*:

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

- *Giải thích*:
  - **`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.
  - Ghi **`n`** phần tử, mỗi phần tử có kích thước là **`size`** byte, từ vùng nhớ được trỏ bởi con trỏ **`ptr`** lên tập tin **`fp`**.
- *Kết quả trả về của hàm*
  - **`n`**: nếu thành công.
  - **`EOF`**: nếu gặp lỗi.

5.2. Đọc dữ liệu tập tin nhị phân

- *Cú pháp*:

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

- *Giải thích*:
  - `ptr`: con trỏ chỉ đến vùng nhớ sẽ nhận dữ liệu từ tập tin.
  - `n`: số phần tử sẽ đọc từ tập tin.
  - `size`: kích thước của mỗi phần tử.
  - `f`: con trỏ tập tin đã được mở.
  - Đọc `n` phần tử, mỗi phần tử có kích thước là `size` byte, từ tập tin `fp` và lưu vào vùng nhớ được trỏ bởi con trỏ `ptr`.
- *Kết quả trả về của hàm*
  - `=n`: nếu thành công.
  - `<n`: khi đọc đến cuối tập tin hoặc khi gặp lỗi.

5.3. Di chuyển con trỏ tập tin

- *Hoạt động của 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.
  - Khi 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.
  - Có thể điều khiển việc di chuyển con trỏ tập tin đến vị trí chỉ định bằng hàm `fseek()`.

5.3. Di chuyển con trỏ tập tin

5.3.1. Di chuyển bằng hàm rewind

- *Cú pháp*

```
void rewind(FILE *fp);
```

- *Giải thích*
  - **fp** : con trỏ tập tin đang thao tác.
  - Di chuyển con trỏ tập tin về đầu tập tin **fp**. Khi đó việc truy xuất trên tập tin được thực hiện từ đầu tập tin.

5.3. Di chuyển con trỏ tập tin

5.3.2. Di chuyển bằng hàm fseek

- *Cú pháp*:

```
int fseek (FILE *f, long offset, int whence)
```

- *Giải thích*:
  - **fseek** di chuyển con trỏ **f** đến vị trí **offset** theo mốc **whence**
  - **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
- *Kết quả trả về của hàm*
  - **=0**: nếu thành công.
  - **!=0**: khi di chuyển gặp lỗi.

5.3. Di chuyển con trỏ tập tin

5.3.3. Lấy vị trí hiện tại của con trỏ tập tin

- *Cú pháp*

```
long ftell(FILE *fp);
```

- *Giải thích*
  - Nếu thành công hàm trả về vị trí hiện tại của con trỏ tập tin (byte thứ mấy trên tập tin, số thứ tự byte được tính từ 0).
  - Nếu có lỗi hàm trả về giá trị -1L.

5.3. Di chuyển con trỏ tập tin

5.3.3. Lấy vị trí hiện tại của con trỏ tập tin

- Ví dụ 1: Tạo một tập tin chứa danh sách các sinh viên, danh sách này được lưu sẵn trong một mảng cấu trúc.

```
#define SIZE 20
struct SINHVIEN {
        char ma[10];
        char ten[30];
        int namsinh;
        float diem;
};
struct DSSV{
        int n;
        SINHVIEN arr[SIZE];
};
```

5.3. Di chuyển con trỏ tập tin

5.3.3. Lấy vị trí hiện tại của con trỏ tập tin

- Ví dụ 1: Tạo một tập tin chứa danh sách các sinh viên, danh sách này được lưu sẵn trong một mảng cấu trúc.

```
int TaoTTDSSV(char *tentt, DSSV u)
{    FILE *fp;
     int ret;
     fp = fopen(tentt, "wb");
     if(fp != NULL)
     {    /*Ghi u.n cấu trúc, mỗi cấu trúc kích thước
          sizeof(SINHVIEN ), chứa trong mảng u.arr
                                        vào tập tin fp*/
          fwrite(u.arr, sizeof(SINHVIEN), u.n, fp);
          fclose(fp);
          ret = 1;
     }
     else ret = 0;
          return ret;
}
```

5.3. Di chuyển con trỏ tập tin

5.3.3. Lấy vị trí hiện tại của con trỏ tập tin

- Ví dụ 1: Tạo một tập tin chứa danh sách các sinh viên, danh sách này được lưu sẵn trong một mảng cấu trúc.

```
void main()
{
    DSSV dssv;
    char tentt[MAX_LEN];
    printf("Nhap ten tap tin muon tạo:");
    gets(tentt);
    NhapDSSV(&dssv);
    int kq = TaoTTDSSV(tentt, dssv);
    if(kq == 0)
        printf("Khong the tao tap tin\n");
}
```

5.3. Di chuyển con trỏ tập tin

5.3.3. Lấy vị trí hiện tại của con trỏ tập tin

- Ví dụ 2: Đọc danh sách các sinh viên chứa trong tập tin vừa tạo ở trên và xuất ra màn hình.

```
int DocTTDSSV(char *tentt);
void main()
{
    char tentt[MAX_LEN];
    <Nhap ten tap tin muon doc>
    int kq = DocTTDSSV(tentt);
    if(kq == 0)
        printf("Tap tin %s khong ton tai tren
                                    dia\n", tentt);
}
```

5.3.3. Lấy vị trí hiện tại của con trỏ tập tin

- Ví dụ 2: Đọc danh sách các sinh viên chứa trong tập tin vừa tạo ở trên và xuất ra màn hình.

```
int DocTTDSSV(char *tentt)
{     FILE *fp; SINHVIEN sv; int ret;
      fp = fopen(tentt, "rb");
      if(fp != NULL) {
            while(!feof(fp))
            {     /*Đọc một cấu trúc có kích thước
      sizeof(SINHVIEN) từ tập tin fp vào biến sv*/
                  if(fread(&sv, sizeof(SINHVIEN), 1, fp) == 1)
                        Xuat1SV(sv);
            }
            fclose(fp);
            ret = 1;
      }
      else
            ret = 0;
      return ret;
}
```

5.3. Di chuyển con trỏ tập tin

5.3.3. Lấy vị trí hiện tại của con trỏ tập tin

- Ví dụ 3: Tương tự như ví dụ trên nhưng danh sách các sinh viên đọc từ tập tin được lưu vào một mảng cấu trúc (số sinh viên đọc được không được vượt quá kích thước mảng cấu trúc), sau đó xuất mảng cấu trúc này ra màn hình.

```
int DocTTDSSV2(char *tentt, DSSV *u);
void main()
{    char tentt[MAX_LEN];
     DSSV dssv;
     /*Nhap ten tap tin muon doc*/
     int kq = DocTTDSSV2(tentt, &dssv);
     if(kq == 1)
          XuatDSSV(dssv);
     else
          printf("Khong tim thay tap tin %s\n", tentt);
}
```

5.3. Di chuyển con trỏ tập tin

5.3.3. Lấy vị trí hiện tại của con trỏ tập tin

- Ví dụ 3:

```
int DocTTDSSV2(char *tentt, DSSV *u)
{    FILE *fp; int ret;
     fp = fopen(tentt, "rb");
     if(fp != NULL)
     {    u->n = 0;
          while(!feof(fp) && u->n < SIZE)
          {    if(fread(&u->arr[u->n], sizeof(SinhVien),
                                                  1, fp)== 1)
                    u->n++;
          }
          fclose(fp);
          ret = 1;
     }
     else
          ret = 0;
     return ret;
}
```

5.3. Di chuyển con trỏ tập tin

5.3.3. Lấy vị trí hiện tại của con trỏ tập tin

- Ví dụ 4: Thêm một sinh viên vào tập tin chứa danh sách sinh viên đã tạo ở trên.

```
int ThemSVVaoTTDSSV(char *tentt, SINHVIEN sv) {
{    FILE *fp; int ret;
     fp = fopen(tentt, "ab");
     if(fp != NULL)
     {    /*Ghi cấu trúc có kích thước sizeof(SINHVIEN)
                    chứa trong biến sv lên tập tin fp*/
          fwrite(&sv, sizeof(SINHVIEN), 1, fp);
          fclose(fp);
          ret = 1;
     }
     else ret = 0;
          return ret;
}
```

5.3. Di chuyển con trỏ tập tin

5.3.3. Lấy vị trí hiện tại của con trỏ tập tin

- Ví dụ 4:

```
void main()
{    SINHVIEN sv;
     char tentt[MAX_LEN];
     printf("Nhap ten tap tin muon them:");
     gets(tentt);
     printf("Nhap thong tin sinh vien muon them\n");
     Nhap1SV(&sv);
     int kq = ThemSVVaoTTDSSV(tentt, sv);
     if(kq == 1)
     {    printf("Danh sach sinh vien sau khi them\n");
          DocTTDSSV(tentt);
     }
     else
          printf("Tap tin %s khong ton tai tren dia\n",
                                                  tentt);
}
```

5.3. Di chuyển con trỏ tập tin

5.3.3. Lấy vị trí hiện tại của con trỏ tập tin

– Ví dụ 5: Nhập vào tên tập tin chứa danh sách sinh viên, mã sinh viên và năm sinh mới. Hãy tìm trong tập tin một sinh viên có mã trùng với mã nhập vào, nếu tìm thấy thì sửa năm sinh cũ thành năm sinh mới.

5.3. Di chuyển con trỏ tập tin

5.3.3. Lấy vị trí hiện tại của con trỏ tập tin

– Ví dụ 5:

```
void main()
{    char ma[10]; int namsinh; char tentt[MAX_LEN];
     /*Nhap ten tt muon sua*/
     /*Nhap ma sinh vien cua sinh vien can sua*/
     /*Nhap nam sinh moi*/
     int kq = SuaTTDSSV(tentt, ma, namsinh);
     if(kq == 0)
          printf("Khong tim thay tap tin %s\n", tentt);
     else if(kq == -1)
               printf("Khong tim thay sv co ma %s\n", ma);
          else
          {    printf("** Danh sach sinh vien moi **\n");
               DocTTDSSV(tentt);
          }
     }
```

5.3. Di chuyển con trỏ tập tin

5.3.3. Lấy vị trí hiện tại của con trỏ tập tin

```
int SuaTTDSSV(char *tentt, char *ma, int namsinh)
{     FILE *fp; SinhVien sv; int timthay = 0, vitri, ret;
      fp = fopen(tentt, "r+b");
      if(fp != NULL)
      {     while(!feof(fp))
            {     vitri = ftell(fp);
                  if(fread(&sv, sizeof(SinhVien), 1, fp) == 1)
                  {     if(strcmp(sv.ma, ma) == 0)
                        {     sv.namsinh = namsinh;
                              fseek(fp, vitri, SEEK_SET);
                              fwrite(&sv,sizeof(SinhVien),1,fp);
                              timthay = 1; break;
                        }
                  }
            }
            fclose(fp); ret = (timthay == 1 ? 1 : -1);
      }
      else
            ret = 0;
      return ret;
}
```

6. CÁC HÀM THAO TÁC CHUNG TRÊN TẬP TIN VĂN BẢN & TẬP TIN NHỊ PHÂN

6.1. Hàm putc()

- 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)`
- ***Giải thích***:
  - 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`.
- ***Kết quả trả về của hàm***
  - **`EOF`**: nếu gặp lỗi.

*int fgetc(FILE *fp)*

- Hàm đọc một ký tự từ tập tin fp. Nếu thành công hàm trả về mã đọc được (có giá trị từ 0 đến 255), nếu thất bại hoặc gặp cuối tập tin hàm trả về giá trị EOF.
- Ví dụ: Nhập vào tên một tập tin văn bản. Hãy đọc từng ký tự của tập tin, nếu ký tự đọc được là chữ thường thì đổi thành chữ hoa và ghi ký tự trở lại tập tin.

Ví dụ: cho nhập tên tập tin, mở tập tin và đổi tất cả các ký tự in thường thành ký tự in hoa

```
void main()
{
  char tentt[MAX_LEN];
  printf("Nhap ten tap tin muon sua");
  gets(tent);
  int kq = DoiHoaTT(tentt);
  if(kq == 1)
  {
        printf("** Tap tin sau khi doi hoa **\n");
        DocTT(tentt);
  }
  else
        printf("Khong tim thay tap tin %s \n", tentt);
}
```

6. Các hàm thao tác chung trên tập tin văn bản & Tập tin nhị phân

```
int DoiHoaTT(char *tentt)
{
  FILE *fp; char ch; int vitri, ret;
  fp = fopen(tentt, "r+b");
  if(fp != NULL)
  {   while(!feof(fp))
      {  if((ch = fgetc(fp)) != EOF)
         {    if(ch >= 'a' && ch <= 'z')
              {    ch -= 32;
                   vitri = ftell(fp);
                   fseek(fp, -1, SEEK_CUR);
                   fputc(ch, fp);
                   fseek(fp, vitri, SEEK_SET); }
              }
         }
         fclose(fp);
         ret = 1;
      }
  else
      ret = 0;
  return ret;
}
```

7. THỰC HÀNH

1. Nhập một ma trận vuông và ghi thành file TEXT trên đĩa với tên file là “matran.txt” theo yêu cầu sau :
   - Dòng đầu ghi cấp của ma trận
   - Các dòng tiếp theo là giá trị các phần tử trên mỗi dòng của ma trận. Hai số kề nhau được lưu bởi khoảng trắng.
   - Đọc file “matran.txt” để tìm:
     - phần tử lớn nhất
     - phần tử nhỏ nhất trong ma trận
     - xuất ra dòng có tổng số các phần tử là lớn nhất
     - …

7. THỰC HÀNH

2. Nhập một danh sách sinh viên (mỗi sinh viên có mã, họ tên, năm sinh, điểm trung bình) và ghi lên đĩa thành một tập tin.
   - Viết hàm đọc file và xuất ra danh sách
   - Nhập thêm dữ liệu cho mẫu tin của một sinh viên mới vào cuối tập tin
   - Tìm kiếm sinh viên theo mã
   - …


Q&A
How
Why
When
Where
What
Who