



Chương 09 TẬP TIN

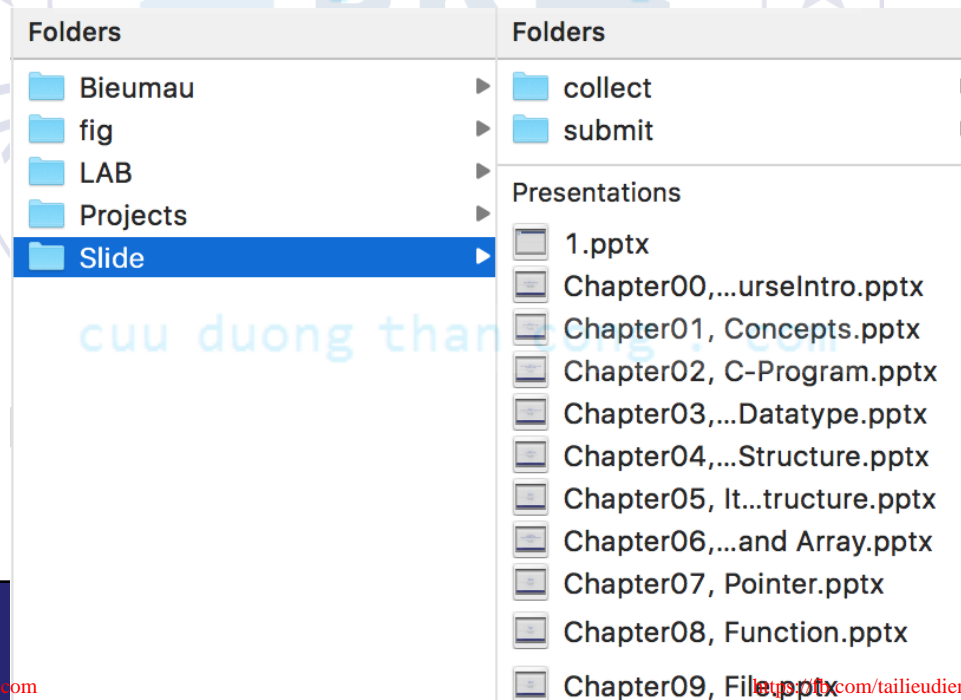
Nguyễn Thanh Tùng

Nội dung

- Tại sao phải dùng tập tin (file)?
- Mô hình tập tin
- Các loại tập tin
- Các thao tác bắt buộc
- Đọc và ghi dữ liệu vào tập tin
- Tập tin văn bản
 - Đọc, ghi, đọc và ghi
- Tập tin nhị phân
 - Đọc, ghi, đọc và ghi
- Các hàm xử lý tập tin
- Các ví dụ
- Tổng kết

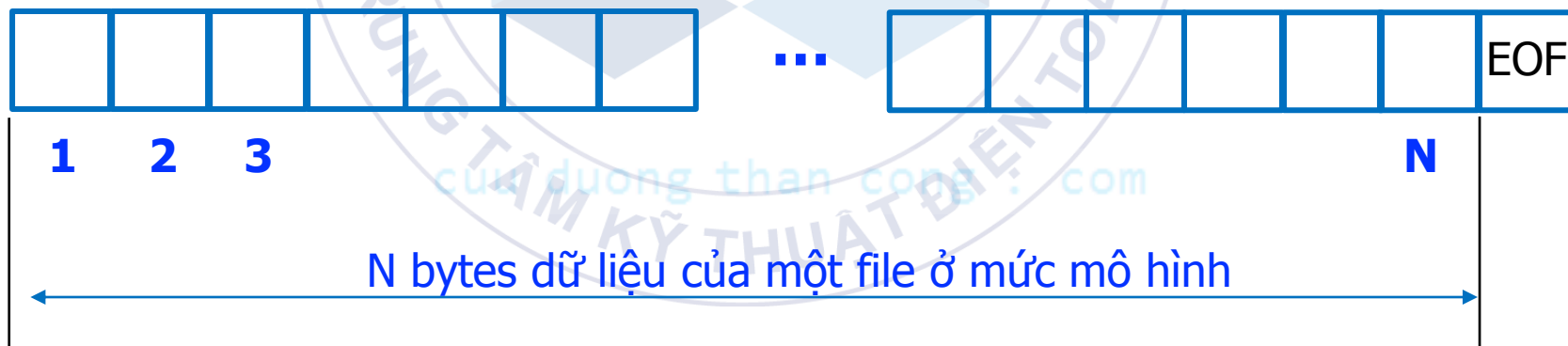
Tại sao phải dùng tập tin (file)?

- Khi một chương trình kết thúc thực thi, các biến dữ liệu liên quan sẽ bị dọn dẹp khỏi bộ nhớ chính (RAM) của máy tính
 - => Để dữ liệu không bị chương trình mất đi khi chương trình kết thúc, chương trình cần lưu chúng dưới dạng tập tin (file) vào các thiết bị lưu trữ như ổ cứng, CD, DVD, v.v.



Mô hình tập tin

- Tập tin là một dãy các bytes dữ liệu, như hình vẽ, kết thúc bằng ký hiệu đặc biệt EOF
 - **EOF** (End Of File): là giá trị đặt biệt, không trùng với bất cứ giá trị của byte dữ liệu nào.
 - **EOF**: Ký hiệu mà các hàm đọc dữ liệu trả về để cho biết kết thúc tập tin.
 - (Nhiều hệ thống EOF = -1)



Các loại tập tin

- Tập tin văn bản (text)
 - Các byte trong mô hình tập tin chứa các ký tự đọc được (có nghĩa) bởi con người
 - Tập tin có thể mở ra để đọc và thay đổi bởi chương trình soạn thảo văn bản như NOTEPAD.
- Tập tin nhị phân (binary)
 - Được tạo bởi chương trình nào đó, không dành cho con người đọc và hiểu trực tiếp bằng NOTEPAD
 - Tập tin dành riêng cho chương trình nào đó đọc và diễn dịch theo ý của chương trình đó.

Các thao tác bắt buộc

- (1) Khai báo sử dụng kiểu dữ liệu tập tin
- (2) Mở tập tin
 - Hàm: **fopen**, nói sau
- (3) Thao tác với tập tin
 - Đọc hay ghi dữ liệu
 - Mỗi lần đọc hay ghi dữ liệu, thẻ đánh dấu trong tập tin tự động tăng đến phần tử tiếp theo
- (4) Đóng tập tin
 - Hàm: **fclose**, nói sau

Các thao tác bắt buộc

Thẻ đánh dấu trong tập tin



Sau khi mở tập tin thành công, thẻ đánh dấu tự động chỉ đến byte đầu tiên của tập tin

Các thao tác bắt buộc

Thẻ đánh dấu trong tập tin



Sau khi đọc 1 byte dữ liệu, ví dụ sử dụng hàm `fgetc`



Các thao tác bắt buộc

Thẻ đánh dấu trong tập tin



Sau khi đọc 1 byte dữ liệu nữa

BK
TP.HCM

cuu duong than cong . com

TRUNG TÂM KỸ THUẬT ĐIỆN TOÁN

TRƯỜNG ĐẠI HỌC BÁCH KHOA

Các thao tác bắt buộc

Thẻ đánh dấu trong tập tin



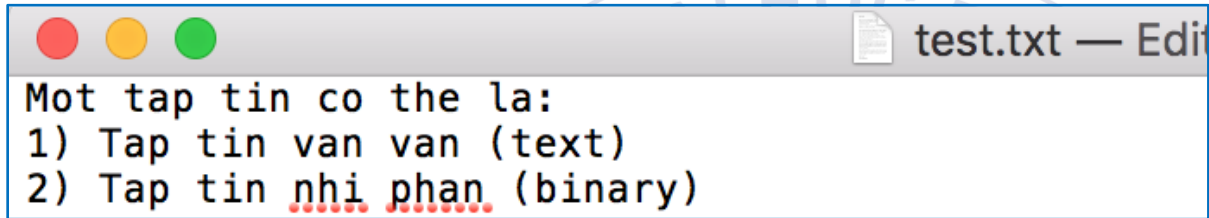
Sau khi đã đọc thành công N bytes
Thẻ đánh dấu chỉ đến EOF

Lần đọc dữ liệu kế tiếp hàm đọc trả
về giá trị EOF để nói rằng kết thúc
tập tin, và giá trị trả về là EOF (-1)

cuuduongthancong.com

Tập tin văn bản: Đọc tập tin

Yêu cầu

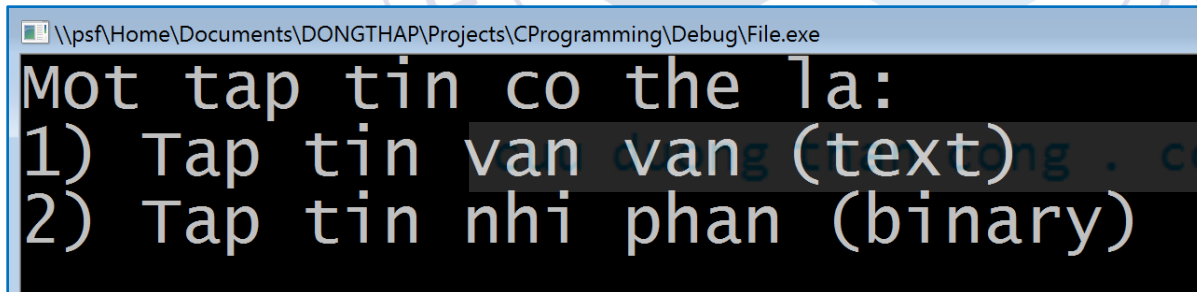


```
Mot tap tin co the la:  
1) Tap tin van van (text)  
2) Tap tin nhi phan (binary)
```

Vào: tập tin văn bản
(đọc được)

Chương trình

Chương trình: đọc các dòng
văn bản và đưa vào bộ đệm

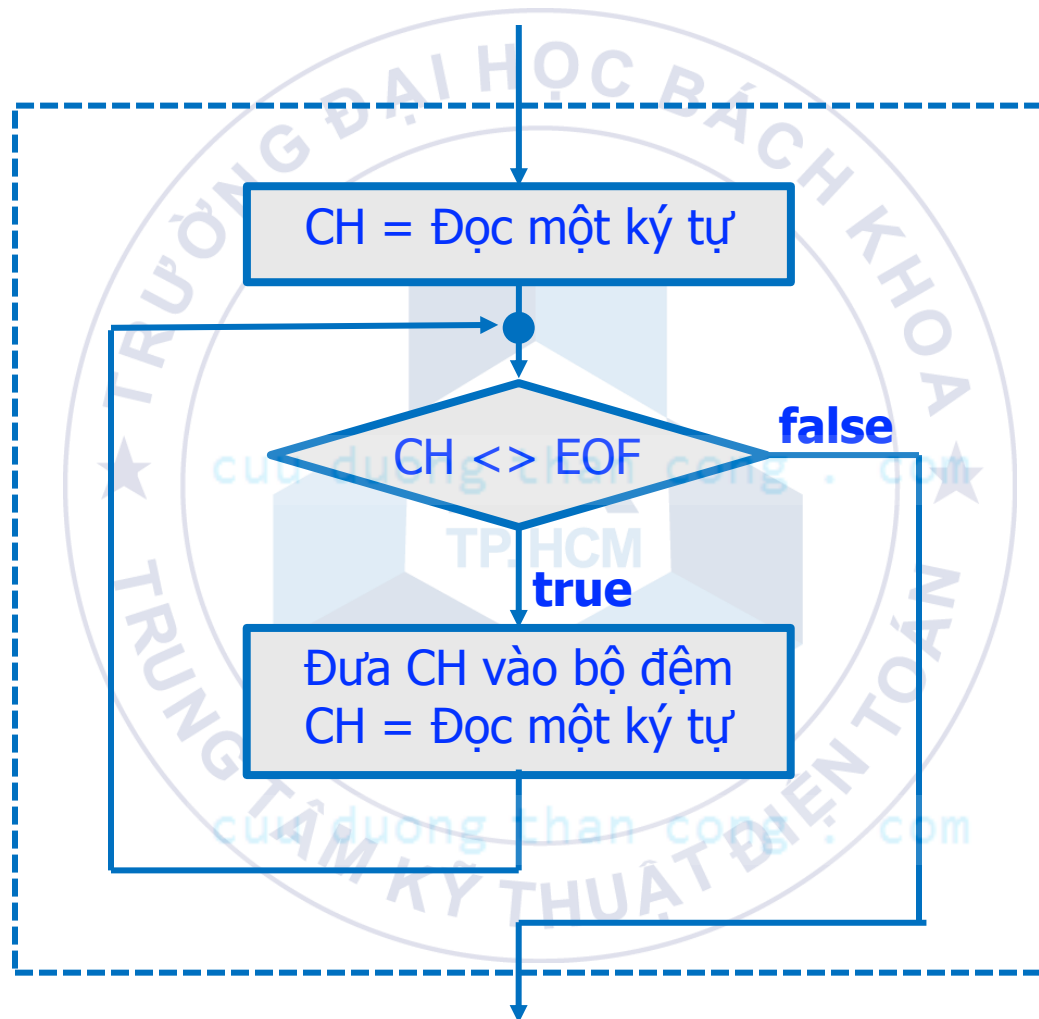


```
Mot tap tin co the la:  
1) Tap tin van van (text)  
2) Tap tin nhi phan (binary)
```

Ra: in lại các dòng
trên màn hình

Tập tin văn bản: Đọc tập tin

Ý tưởng



Tập tin văn bản: Đọc tập tin

(1) Khai báo con trỏ tập tin

```
FILE* file_ptr = NULL;
```

Con trỏ đến tập tin, một danh hiệu

Kiểu dữ liệu **FILE**
Định nghĩa trong `<stdio.h>`
→ Đặt `#include <stdio.h>` đầu tập tin

Tập tin văn bản: Đọc tập tin

(2) Mở tập tin

```
FILE* file_ptr = NULL;
file_ptr = fopen("test.txt", "r");
if(file_ptr == NULL){
    perror("Loi da xay ra: ");
    system("pause");
    exit(EXIT_FAILURE);
}
```

- Mở tập tin bằng hàm fopen
 - Tên tập tin: "test.txt"
 - Mục tiêu của việc mở là: "r" ⇔ cho việc đọc (read)
- Kiểm tra xem file_ptr có NULL không
 - = NULL nghĩa là không mở được tập tin
 - = NULL, có thể chấm dứt thực thi

Tập tin văn bản: Đọc tập tin

(2) Mở tập tin

```
FILE* file_ptr = NULL;
file_ptr = fopen("test.txt", "r");
if(file_ptr == NULL){
    perror("Loi da xay ra: ");
    system("pause");
    exit(EXIT_FAILURE);
}
```

"test.txt": Chương trình tìm tập tin này ở thư mục hiện tại (chứa tập tin thực thi), hoặc trong các thư mục chứa trong biến môi trường PATH

Nếu chỉ ra đường dẫn, dùng **HAI DẤU "\\"** thay cho chỉ 01 dấu; vì "\" là ký tự điều khiển trong chuỗi

Ví dụ: Dùng **C:\\DATA\\Test.txt** thay cho: C:\\DATA\\Test.txt

Tập tin văn bản : Đọc tập tin

(2) Mở tập tin – chế độ mở

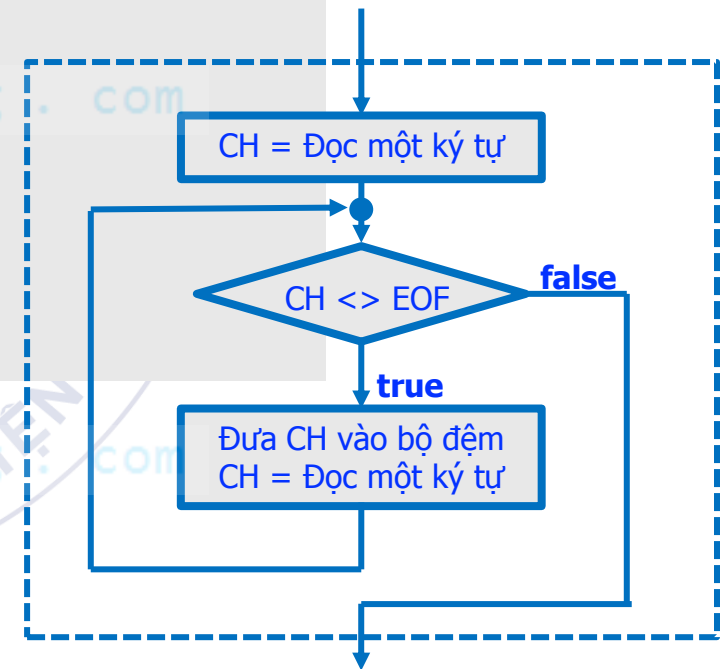
Chế độ	Mô tả
r	Mở tập tin để đọc.
w	Mở tập tin để ghi. Nếu tập tin đã tồn tại, xóa toàn bộ nội dung tập tin đó.
a	Nối tập tin. Mở tập tin đã có sẵn hoặc tạo mới tập tin, ghi vào đuôi của tập tin nếu nó tồn tại.
r+	Mở tập tin cho phép đọc lẫn ghi. Không tạo mới tập tin nếu tập tin chưa có sẵn.
w+	Mở tập tin cho phép đọc lẫn ghi. Tạo mới tập tin nếu tập tin chưa có sẵn.
a+	Nối tập tin, cho phép đọc tập tin. Mở tập tin đã có sẵn hoặc tạo mới tập tin, ghi vào đuôi của tập tin đó.

Tập tin văn bản : Đọc tập tin

(3) Đọc tất cả các ký tự trong tập tin vào một bộ đệm

```
void doc_tap_tin(FILE* file_ptr, char* buffer){  
    int index = 0;  
  
    int ch = fgetc(file_ptr);  
    while(ch != EOF){  
        buffer[index] = ch;  
        ch = fgetc(file_ptr);  
        index += 1;  
    }  
    buffer[index] = '\0';  
}
```

Giả sử buffer đủ lớn để chứa toàn bộ dữ liệu từ tập tin



Tập tin văn bản : Đọc tập tin

(3) Đọc tất cả các ký tự trong tập tin vào một bộ đệm

```
void doc_tap_tin(FILE* file_ptr, char* buffer){  
    int index = 0;  
  
    int ch = fgetc(file_ptr);  
    while(ch != EOF){  
        buffer[index] = ch;  
        ch = fgetc(file_ptr);  
        index += 1;  
    }  
    buffer[index] = '\\0';  
}
```

fgetc: hàm đọc từng ký tự

Tập tin văn bản : Đọc tập tin

(3) Đọc tất cả các ký tự trong tập tin vào một bộ đệm

```
void doc_tap_tin(FILE* file_ptr, char* buffer){
    int index = 0;

    int ch = fgetc(file_ptr);
    while(ch != EOF){
        if(index < MAX_LEN - 1){
            buffer[index] = ch;
            ch = fgetc(file_ptr);
            index += 1;
        }
        else break;
    }
    buffer[index] = '\0';
}
```

- Nếu chỉ số vượt qua chỉ số giới hạn thì thoát khỏi vòng lặp, không đọc nữa, dù còn ký tự trong tập tin

Trường hợp kiểm tra cả độ bộ đệm

- Gán ký tự kết thúc chuỗi cho bộ đệm

Tập tin văn bản : Đọc tập tin

(3) Đọc tất cả các ký tự trong tập tin vào một bộ đệm

■ Hàm **fgetc**:

- Nhận vào con trỏ đến tập tin
- Trả về một giá trị kiểu int
- Nếu giá trị trả về từ hàm **fgetc** là **EOF** thì chỉ ra là kết thúc tập tin
- Ngược lại:
 - **fgetc** trả về một ký tự có thể ép kiểu vào unsigned char (0, ..., 255)
 - **fgetc** đặt thẻ đánh dấu tại byte kế tiếp trên tập tin, để lần sau đó gọi **fgetc** sẽ đọc ký tự kế tiếp

Tập tin văn bản : Đọc tập tin

(4) Đóng tập tin

- Đóng tập tin bằng hàm `fclose`

```
fclose(file_ptr);  
printf("%s", buffer);
```

- In `buffer` ra màn hình

cuuduongthanhong.com

cuuduongthanhong.com

TRUNG TÂM KỸ THUẬT ĐIỆN TOÁN

BK

TRƯỜNG ĐẠI HỌC BÁCH KHOA

Tập tin văn bản : Đọc tập tin

Chương trình hoàn chỉnh

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_LEN 1024
int main(){
    //Mở tập tin
    FILE* file_ptr = NULL;
    file_ptr = fopen("test.txt", "r");
    if(file_ptr == NULL){
        perror("Lỗi đã xảy ra: ");
        system("pause");
        exit(EXIT_FAILURE);
    }
    //Đọc tập tin
    char buffer[MAX_LEN];
    doc_tap_tin(file_ptr, buffer);
    //Đóng tập tin
    fclose(file_ptr);
    printf("%s", buffer);
    system("pause");
    return EXIT_SUCCESS;
}
```

Hàm đọc ở
slide trước

Tập tin văn bản : Đọc file

Các hàm thao tác tập tin khác

Hàm	Mô tả
fopen	Mở luồng xử lý tập tin.
fclose	Đóng luồng xử lý tập tin.
fscanf	Nhận các dữ liệu phổ biến như char, int, float v.v. từ tập tin.
fprintf	In các dữ liệu phổ biến như char, int, float, v.v. lên tập tin.
rewind	Quay con trỏ byte trong con trỏ FILE về đầu tập tin. Như vậy, ta có thể đọc tập tin nhiều lần mà không cần đóng rồi mở lại tập tin.
fgets	Hàm fgets (char* str , int num , FILE* stream) Đọc tập tin theo từng cụm num các ký tự và sao chép cụm đó vào mảng char mà str trỏ đến. Nếu một hàng trong tập chứa ít hơn num các ký tự thì hàm sẽ sao toàn bộ hàng đó vào mảng mà str trỏ đến.
fputs	Hàm fputs (const char* str , FILE* stream) Ghi chuỗi ký tự mà str trỏ đến vào luồng xử lý tập tin đang mở.

Các hàm trên đều nằm trong thư viện stdio.h

Tập tin văn bản: Ghi tập tin

Các bước tương tự như đọc tập tin

Bước	Đọc tập tin	Ghi tập tin
(1) Khai báo con trỏ tập tin	<code>FILE *file_ptr</code>	<code>FILE *file_ptr</code>
(2) Mở tập tin	<code>file_ptr = fopen("test.txt", "r");</code>	<code>file_ptr = fopen("test.txt", "w");</code>
(4) Đóng tập tin	<code>fclose(file_ptr)</code>	<code>fclose(file_ptr)</code>

Chỉ có bước số (3) Dành cho việc đọc và ghi là thực sự khác nhau nhiều

Tập tin văn bản: Ghi tập tin

Bài toán

- Giả sử cần viết chương trình đọc và ghi tập tin có định dạng dạng như sau:

Nguyen Van A	:9.8 , 7.2, 9.5
Tran Van B	:4.0 , 5.3, 2.5
Phan Dinh C	:8.7 , 7.9, 8.1

- Phân tích bài toán:
 - Mỗi hàng dữ liệu gồm tên, 3 cột điểm (các con số thực). Độ rộng của từng cột là cố định.
 - Hàm **fprintf** để in các số liệu xuống tập tin có tính năng tương tự như printf → nên sử dụng.

Tập tin văn bản: Ghi tập tin

Bài toán

■ Phân tích bài toán:

- Cần định nghĩa kiểu dữ liệu Student gồm các trường thông tin như sau

```
typedef struct{  
    char name[20];  
    float math, physics, english;  
} Student;
```

Tập tin văn bản: Ghi tập tin

Bài toán

■ Phân tích bài toán:

- Cần khai báo danh sách sinh viên. Có thể khởi động bằng một số dữ liệu mẫu để kiểm tra như sau:

```
Student list[] = {  
    {"Nguyen Van A", 9.8f, 7.2f, 9.5f},  
    {"Tran Van B", 4.0f, 5.3f, 2.5f},  
    {"Phan Dinh C", 8.7f, 7.9f, 8.1f},  
};
```

Tập tin văn bản: Ghi tập tin

Bài toán

■ Phân tích bài toán:

- Ý tưởng chính của việc ghi là: duyệt qua từng phần tử trong mảng và ghi từng phần tử vào tập tin

```
for(int i=0; i< 3; i++){  
    fprintf(file_ptr, "%-15s:%-5.1f,%5.1f,%5.1f\n",  
        list[i].name,  
        list[i].math, list[i].physics, list[i].english);  
}
```

file_ptr: là con trỏ đến FILE, và đã mở tập tin cho ghi trước đó.

Tập tin văn bản: Ghi tập tin

Bài toán – chương trình hoàn chỉnh

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_LEN 1024

typedef struct{
    char name[20];
    float math, physics, english;
} Student;

int main(){
    Student list[] = {
        {"Nguyen Van A", 9.8f, 7.2f, 9.5f},
        {"Tran Van B", 4.0f, 5.3f, 2.5f},
        {"Phan Dinh C", 8.7f, 7.9f, 8.1f},
    };
};
```

cuu duong than cong . com

cuu duong than cong . com

Tập tin văn bản: Ghi tập tin

Bài toán – chương trình hoàn chỉnh

```
FILE* file_ptr = NULL;
file_ptr = fopen("data.txt", "a");
if(file_ptr == NULL){
    perror("Co loi: ");
    system("pause");
    exit(EXIT_FAILURE);
}
for(int i=0; i< 3; i++){
    fprintf(file_ptr, "%-15s:%-5.1f,%5.1f,%5.1f\n",
            list[i].name,
            list[i].math, list[i].physics, list[i].english);
}

fclose(file_ptr);
system("pause");
return EXIT_SUCCESS;
}
```

Tập tin văn bản có định dạng: đọc tập tin

Bài toán

Đọc tập tin có định dạng

Nguyen Van A	:9.8 , 7.2, 9.5
Tran Van B	:4.0 , 5.3, 2.5
Phan Dinh C	:8.7 , 7.9, 8.1

Tập tin văn bản có định dạng: đọc tập tin

Các thao tác

```
int main(){
    Student list[100];
    int size = 0;

    FILE* file_ptr = NULL;
    mo_tap_tin(&file_ptr, "data.txt");
    size = doc_du_lieu(file_ptr, list);
    dong_tap_tin(file_ptr);
    in_du_lieu(list, size);

    system("pause");
    return EXIT_SUCCESS;
}
```

1. Khai báo con trỏ tập tin
2. Mở tập tin cho đọc
3. Đọc dữ liệu có định dạng
4. Đóng tập tin
5. In dữ liệu

cuu duong than cong . com

Tập tin văn bản có định dạng: đọc tập tin

Các thao tác

Kiểu dữ liệu và các prototype hàm

```
typedef struct{
    char name[20];
    float math, physics, english;
} Student;

void mo_tap_tin(FILE** file_ptr, char tap_tin[]);
void dong_tap_tin(FILE* file_ptr);

bool doc_ten(FILE* file_ptr, char* name, char end_char);
bool doc_diem(FILE* file_ptr, float *math, float *physics, float *english);
bool xoa_xuong_hang(FILE* file_ptr);
int doc_du_lieu(FILE* file_ptr, Student list[]);
void in_du_lieu(Student list[], int size);
```

Tập tin văn bản có định dạng: đọc tập tin

Các thao tác

Mở tập tin cho đọc

```
void mo_tap_tin(FILE** file_ptr, char tap_tin[]){  
    *file_ptr = fopen(tap_tin, "r");  
    if(*file_ptr == NULL){  
        perror("Co loi: ");  
        system("pause");  
        exit(EXIT_FAILURE);  
    }  
}
```

Tập tin văn bản có định dạng: đọc tập tin

Các thao tác

Mở và đóng tập tin cho đọc

```
void mo_tap_tin(FILE** file_ptr, char tap_tin[]){
    *file_ptr = fopen(tap_tin, "r");
    if(*file_ptr == NULL){
        perror("Co loi: ");
        system("pause");
        exit(EXIT_FAILURE);
    }
}
```

```
void dong_tap_tin(FILE* file_ptr){
    fclose(file_ptr);
}
```

Tập tin văn bản có định dạng: đọc tập tin

Các thao tác

```
int doc_du_lieu(FILE* file_ptr, Student list[]){
    int size =0;
    bool doc_tiep = true;
    while(doc_tiep){
        doc_tiep = doc_ten(file_ptr, list[size].name, ':');
        if(!doc_tiep) break;
        doc_tiep = doc_diem(file_ptr,
            &list[size].math,
            &list[size].physics, &list[size].english);
        if(!doc_tiep) break;

        doc_tiep = xoa_xuong_hang(file_ptr);
        size += 1;

        if(!doc_tiep) break;
    }
    return size;
}
```

Các bước đọc dữ liệu

Tập tin văn bản có định dạng: đọc tập tin

Các thao tác

```
bool doc_ten(FILE* file_ptr, char* name, char end_char){
    int i = 0;
    int ch = fgetc(file_ptr);
    while((ch != ':' ) && (ch != EOF)){
        name[i] = ch;
        i += 1;
        ch = fgetc(file_ptr);
    }
    name[i] = '\0';

    if(ch == EOF) return false;
    else return true;
}
```

Tập tin văn bản có định dạng: đọc tập tin

Các thao tác

```
bool doc_diem(FILE* file_ptr,
    float *math, float *physics, float *english){
    int num = fscanf(file_ptr, "%f , %f, %f",
        math,
        physics,
        english);
    if(num != 3) return false;
    else return true;
}
```

Tập tin văn bản có định dạng: đọc tập tin

Các thao tác

```
bool xoa_xuong_hang(FILE* file_ptr){  
    int ch;  
    ch = fgetc(file_ptr);  
    while((ch != EOF) && (ch == '\n')){  
        ch = fgetc(file_ptr);  
    }  
    if(ch == EOF) return false;  
    else{  
        fseek(file_ptr, -1, SEEK_CUR);  
        return true;  
    }  
}
```

Tập tin văn bản có định dạng: đọc tập tin

Các thao tác

```
void in_du_lieu(Student list[], int size){  
    for(int i=0; i< size; i++){  
        printf("%-20s:%-5.1f,%5.1f,%5.1f\n",  
            list[i].name,  
            list[i].math,  
            list[i].physics,  
            list[i].english);  
    }  
}
```


Đọc và ghi với tập tin

- Hai thao tác phổ biến với tập tin là
 - Ghi vào tập tin
 - Đọc dữ liệu từ tập tin.
- Ghi dữ liệu
 - Sử dụng các hàm thư viện
 - Với tập tin văn bản: `fprintf`, `fputs`
 - Với tập tin nhị phân: `fwrite`
 - **Việc ghi thường dễ dàng hơn đọc.**
 - Với tập tin văn bản: **`fprintf`** tương tự như `printf` có các định dạng
 - `%s`: để ghi chuỗi, với độ rộng, canh lề mong muốn
 - `%f`: để ghi số thực với độ rộng, độ chính xác mong muốn
 - `V.v`

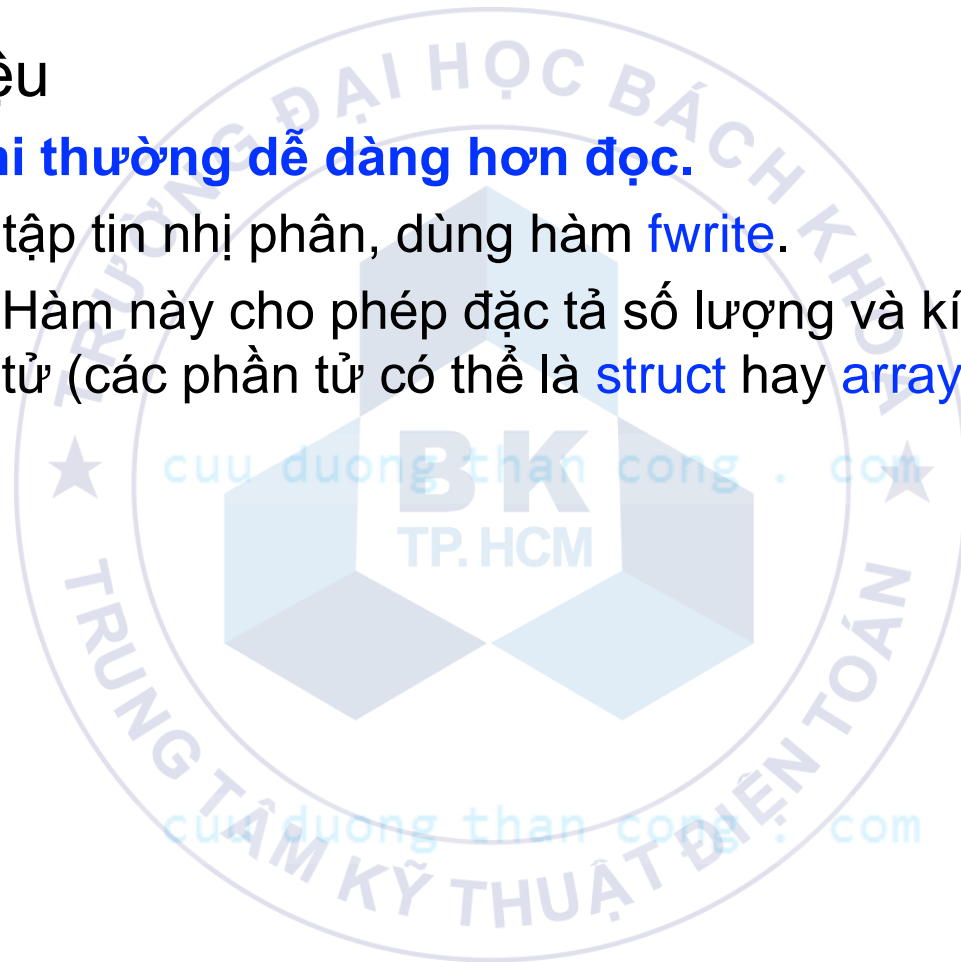
Đọc và ghi với tập tin

- Ghi dữ liệu

- **Việc ghi thường dễ dàng hơn đọc.**

- Với tập tin nhị phân, dùng hàm **fwrite**.

- Hàm này cho phép đặc tả số lượng và kích thước mỗi phần tử (các phần tử có thể là **struct** hay **array**)



Đọc và ghi với tập tin

■ Đọc dữ liệu

- Việc đọc dữ liệu từ tập tin thường phức tạp hơn ghi
- Giải thuật đọc tốt:
 - Phải làm việc được với cấu trúc tập tin bị thay đổi
- Với tập tin nhị phân:
 - Vì không biết trước bao nhiêu phần tử có trong tập tin
 - ➔ Giải thuật cần đọc từng phần tử, cho đến khi gặp cuối tập tin hoặc đến khi một cấu trúc bị lỗi nên chấm dứt việc đọc từ đó trong tập tin.
- Với tập tin văn bản:
 - Việc đọc khó hay dễ tùy vào định dạng dữ liệu

Tập tin nhị phân: Ghi tập tin

Các thao tác

```
int main(){
    Student list[MAX_SIZE];
    int size;

    size = sinh_du_lieu_mau(list);
    ghi_du_lieu(list, size, "stu_list.data");
    in_du_lieu(list, size);

    printf("\n\n");
    system("pause");
    return EXIT_SUCCESS;
}
```

Tập tin nhị phân : Ghi tập tin

Các thao tác

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define MAX_SIZE 100

typedef struct{
    char name[20];
    float math, physics, english;
} Student;

int sinh_du_lieu_mau(Student *list);
void ghi_du_lieu(Student *list, int size, char* file);
void in_du_lieu(Student *list, int size);
```

Tập tin nhị phân : Ghi tập tin

Các thao tác

```
int sinh_du_lieu_mau(Student *list){
    time_t t;
    srand((unsigned int) time(&t));
    for(char c='A'; c <= 'Z'; c++){
        list[c - 'A'].name[0] = c;
        list[c - 'A'].name[1] = '\\0';
        list[c - 'A'].math = ((float)rand()/RAND_MAX)*10;
        list[c - 'A'].physics = ((float)rand()/RAND_MAX)*10;
        list[c - 'A'].english = ((float)rand()/RAND_MAX)*10;
    }
    return ('Z' - 'A' + 1);
}
```

Tập tin nhị phân : Ghi tập tin

Các thao tác

```
void ghi_du_lieu(Student *list, int size, char* file){  
    FILE* file_ptr = NULL;  
    file_ptr = fopen(file, "ab+");  
    if(file_ptr == NULL)  
        exit(EXIT_FAILURE);  
  
    fwrite(list, sizeof(Student), size, file_ptr);  
    fclose(file_ptr);  
}
```

Tập tin nhị phân : Ghi tập tin

Các thao tác

```
void in_du_lieu(Student *list, int size){  
    for(int i=0; i< size; i++){  
        printf("%-20s:%-5.1f,%-5.1f,%-5.1f\n",  
            list[i].name,  
            list[i].math,  
            list[i].physics,  
            list[i].english);  
    }  
}
```


Tập tin nhị phân : Đọc tập tin

Các thao tác

Mở tập tin

```
FILE* file_ptr = NULL;  
file_ptr = fopen(file, "rb");  
if(file_ptr == NULL){  
    perror("Error in open file: ");  
    system("pause");  
    exit(EXIT_FAILURE);  
}
```

Tập tin nhị phân : Đọc tập tin

Các thao tác

Đọc từng hồ sơ

```
int c= 0;
int num = 0;
while(true){
    num = fread(&list[c], sizeof(Student), 1, file_ptr);
    if(num != 1){
        break;
    }
    c++;
}
*size = c;
```

Tập tin nhị phân : Đọc tập tin

Các thao tác

Đóng tập tin

```
fclose(file_ptr);
```

Tập tin nhị phân : Đọc tập tin

```
void doc_du_lieu(Student *list, int *size, char* file){
    FILE* file_ptr = NULL;
    file_ptr = fopen(file, "rb");
    if(file_ptr == NULL){
        perror("Error in open file: ");
        system("pause");
        exit(EXIT_FAILURE);
    }

    int c= 0;
    int num = 0;
    while(true){
        num = fread(&list[c], sizeof(Student), 1, file_ptr);
        if(num != 1){
            break;
        }
        c++;

        *size = c;
        fclose(file_ptr);
    }
}
```

cuu duong than cong . com

cuu duong than cong . com