



# Chuỗi ký tự và tập tin

**Kỹ thuật lập trình**

Trình bày: ...; Email: ...@fit.hcmus.edu.vn

# Nội dung

- Giới thiệu về các dạng tập tin
- Hệ thống nhập xuất trong lập trình
- Lập trình thao tác trên tập tin văn bản thô
- Sử dụng tập tin văn bản thô để lưu trữ dữ liệu của chương trình
- Các vấn đề tìm hiểu mở rộng kiến thức nghề nghiệp
- Thuật ngữ và bài đọc thêm tiếng Anh



# Giới thiệu về các dạng tập tin



# Giới thiệu về tập tin

- Việc lập trình với tập tin nhằm để lưu trữ dữ liệu của chương trình vào bộ nhớ phụ và truy xuất trở lại dữ liệu này khi cần thiết. Thông thường dữ liệu lưu trữ là các tập tin trên đĩa.
- Về mặt kỹ thuật lập trình, người ta xem có hai dạng tập tin chính là tập tin văn bản thô và tập tin tin nhị phân.



# Tập tin văn bản thô

- Đây là dạng tập tin văn bản có cấu trúc đơn giản và thông dụng nhất, có thể xem nội dung và sửa chữa bằng các lệnh của hệ điều hành hay những chương trình soạn thảo văn bản đơn giản.
- Thông thường được lưu trữ trên đĩa dưới dạng .txt.
- Hầu hết mã nguồn chương trình hiện nay đều lưu trữ trên đĩa dưới dạng tập tin văn bản thô.
- Nội dung gồm các ký tự 8-bit
  - Các ký tự thấy được có mã từ 0x20 trở lên.
  - Các ký tự điều khiển có mã nhỏ hơn 0x20.



# Tập tin văn bản thô mở rộng

- Có thể lưu các ký tự Unicode hay ký tự nhiều byte (multi-byte character).
- Hai cấu trúc văn bản thô mở rộng thông dụng nhất là:
  - Unicode text: lưu các ký tự UTF-16.
  - UTF-8: lưu các ký tự độ dài biến động từ 1 đến 4 byte.





# Tập tin nhị phân

- Là các tập tin không có cấu trúc như tập tin văn bản thô.
- Mỗi tập tin bao gồm một dãy các byte dữ liệu, gồm 2 dạng:
  - Các byte tuần tự không liên quan nhau về mặt cấu trúc tổ chức tập tin.
  - Được cấu trúc hóa tùy theo qui ước của phần mềm tạo ra tập tin.



# Hệ thống nhập xuất trong lập trình





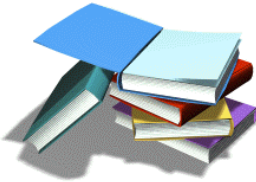
# Các bước để lập trình tập tin

- Bao gồm 3 bước chính:
  - **Bước 1.** Mở tập tin, người lập trình cần phải đưa vào đường dẫn và tên tập tin chính xác.
  - **Bước 2.** Sử dụng tập tin (sau khi đã mở tập tin thành công).
    - Đọc dữ liệu từ tập tin đưa vào biến bộ nhớ trong chương trình.
    - Ghi dữ liệu từ biến bộ nhớ trong chương trình lên tập tin.
  - **Bước 3.** Đóng tập tin (sau khi đã hoàn tất các công việc cần thiết).



# Hàm mở tập tin

**FILE \*fopen**(const char \***filename**, const char \***mode**)



Mở tập tin có tên (đường dẫn) là chứa trong **filename** với kiểu mở **mode** (xem bảng).



- ◆Thành công: con trỏ kiểu cấu trúc **FILE**
- ◆Thất bại: **NULL** (sai quy tắc đặt tên tập tin, không tìm thấy ổ đĩa, không tìm thấy thư mục, mở tập tin chưa có để đọc, ...)



```
FILE* fp = fopen("taptin.txt", "rt");  
if (fp == NULL)  
    printf("Khong mo duoc tap tin!");
```



# Đôi số mở tập tin (mode)

Đôi số	Ý nghĩa
b	Mở tập tin kiểu <b>nhị phân</b> (binary)
t	Mở tập tin kiểu <b>văn bản</b> (text) ( <b>mặc định</b> )
r	Mở tập tin <b>chỉ để đọc</b> dữ liệu từ tập tin. Trả về NULL nếu không tìm thấy tập tin.
w	Mở tập tin <b>chỉ để ghi</b> dữ liệu vào tập tin. Tập tin sẽ được tạo nếu chưa có, ngược lại dữ liệu trước đó sẽ bị xóa hết.
a	Mở tập tin <b>chỉ để thêm</b> (append) dữ liệu vào cuối tập tin. Tập tin sẽ được tạo nếu chưa có.
r+	Giống mode r và bổ sung thêm tính năng ghi dữ liệu và tập tin sẽ được tạo nếu chưa có.
w+	Giống mode w và bổ sung thêm tính năng đọc.
a+	Giống mode a và bổ sung thêm tính năng đọc.



# Đọc và ghi dữ liệu (stdio.h)

- Thực hiện đọc/ghi dữ liệu theo các cách sau:
  - Nhập/xuất theo định dạng
    - Hàm: `fscanf`, `fprintf`
    - Chỉ dùng với tập tin kiểu văn bản.
  - Nhập/xuất từng ký tự hay dòng lên tập tin
    - Hàm: `getc`, `fgetc`, `fgets`, `putc`, `fputs`
    - Chỉ nên dùng với kiểu văn bản.
  - Đọc/ghi trực tiếp dữ liệu từ bộ nhớ lên tập tin
    - Hàm: `fread`, `fwrite`
    - Chỉ dùng với tập tin kiểu nhị phân.



# Hàm xuất theo định dạng

**int fprintf(FILE \*fp, char \*fmt, ...)**



Ghi dữ liệu có chuỗi định dạng **fmt** (giống hàm printf) vào stream **fp**.

Nếu fp là **stdout** thì hàm giống printf.



◆ **Thành công**: trả về số byte ghi được.

◆ **Thất bại**: trả về **EOF** (có giá trị là -1, được định nghĩa trong **STDIO.H**, sử dụng trong tập tin có kiểu văn bản)



```
int i = 2912; int c = 'P'; float f = 17.06;
```

```
FILE* fp = fopen("taptin.txt", "wt");
```

```
if (fp != NULL)
```

```
    fprintf(fp, "%d %c %.2f\n", i, c, f);
```



# Hàm nhập theo định dạng

```
int fscanf(FILE *fp, char *fmt, ...)
```



Đọc dữ liệu có chuỗi định dạng **fmt** (giống hàm scanf) từ stream **fp**.  
Nếu **fp** là **stdin** thì hàm giống printf.



- ◆Thành công: trả về số thành phần đọc và lưu trữ được.
- ◆Thất bại: trả về **EOF**.



```
int i;  
FILE* fp = fopen("taptin.txt", "rt");  
if (fp != NULL)  
    fscanf(fp, "%d", &i);
```





# Hàm nhập ký tự

int **getc**(FILE \***fp**) và int **fgetc**(FILE \***fp**)



Đọc một ký tự từ stream **fp**.

**getc** là macro còn **fgetc** là phiên bản hàm của macro **getc**.

◆ **Thành công**: trả về ký tự đọc được sau khi chuyển sang số nguyên không dấu.

◆ **Thất bại**: trả về **EOF** khi kết thúc stream **fp** hoặc gặp lỗi.

char ch;

FILE\* fp = fopen("taptin.txt", "rt");

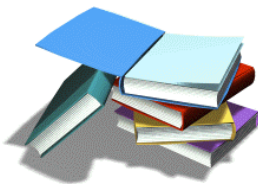
if (fp != NULL)

ch = **getc**(fp); // ⇔ ch = **fgetc**(fp);



# Hàm nhập chuỗi

```
int fgets(char *str, int n, FILE *fp)
```



Đọc một dãy ký tự từ stream **fp** vào vùng nhớ **str**, kết thúc khi đủ **n-1** ký tự hoặc gặp ký tự xuống dòng.



- ◆Thành công: trả về **str**.
- ◆Thất bại: trả về **NULL** khi gặp lỗi hoặc gặp ký tự **EOF**.



```
char s[20];  
FILE* fp = fopen("taptin.txt", "rt");  
if (fp != NULL)  
    fgets(s, 20, fp);
```



# Hàm xuất ký tự

`int putc(int ch, FILE *fp)` và `int fputc(in ch, FILE *fp)`



Ghi ký tự **ch** vào stream **fp**.  
**putc** là macro còn **fputc** là phiên bản hàm của macro **putc**.



- ◆Thành công: trả về ký tự **ch**.
- ◆Thất bại: trả về **EOF**.

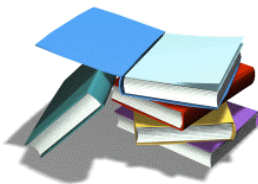


```
FILE* fp = fopen("taptin.txt", "rt");  
if (fp != NULL)  
    putc('a', fp); // hoặc fputc('a', fp);
```



# Hàm xuất chuỗi

```
int fputs(const char *str, FILE *fp)
```



Ghi chuỗi ký tự **str** vào stream **fp**. Nếu **fp** là **stdout** thì **fputs** giống hàm **puts**, nhưng **puts** ghi ký tự xuống dòng.



- ◆Thành công: trả về ký tự cuối cùng đã ghi.
- ◆Thất bại: trả về **EOF**.



```
char s[] = "Ky thuat lap trinh";  
FILE* fp = fopen("taptin.txt", "wt");  
if (fp != NULL)  
    fputs(s, fp);
```



# Hàm xuất trực tiếp

```
int fwrite(void *buf, int size, int count, FILE *fp)
```



Ghi **count** mẫu tin có kích thước mỗi mẫu tin là **size** (byte) từ vùng nhớ **buf** vào stream **fp** (theo kiểu nhị phân).



- ◆Thành công: trả về số lượng mẫu tin (không phải số lượng byte) đã ghi.
- ◆Thất bại: số lượng nhỏ hơn **count**.

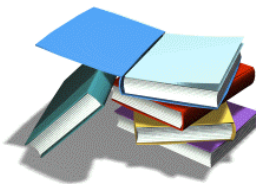


```
int a[] = {1, 2, 3};  
FILE* fp = fopen("taptin.dat", "wb");  
if (fp != NULL)  
    fwrite(a, sizeof(int), 3, fp);
```



# Hàm nhập trực tiếp

```
int fread(void *buf, int size, int count, FILE *fp)
```



Đọc **count** mẫu tin có kích thước mỗi mẫu tin là **size** (byte) vào vùng nhớ **buf** từ stream **fp** (theo kiểu nhị phân).



◆Thành công: trả về số lượng mẫu tin (không phải số lượng byte) thật sự đã đọc.

◆Thất bại: số lượng nhỏ hơn **count** khi kết thúc stream **fp** hoặc gặp lỗi.



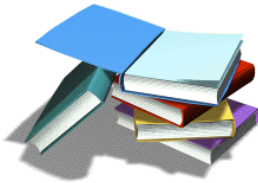
```
int a[5];  
FILE* fp = fopen("taptin.dat", "wb");  
if (fp != NULL)  
    fread(a, sizeof(int), 3, fp);
```





# Hàm đóng tập tin xác định

`int fclose(FILE *fp)`



Đóng stream **fp**.

Dữ liệu trong stream **fp** sẽ được “vét” (ghi hết lên đĩa) trước khi đóng.



◆Thành công: trả về 0.

◆Thất bại: trả về **EOF**.



```
FILE* fp = fopen("taptin.txt", "rt");
```

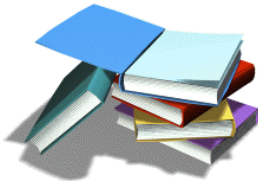
```
...
```

```
fclose(fp);
```



# Hàm đóng tất cả stream

int **fcloseall**()



Đóng tất cả stream đang được mở ngoại trừ các stream chuẩn **stdin**, **stdout**, **stderr**, **stdprn**, **stderr**, **stdaux**.

Nên đóng từng stream thay vì đóng tất cả.



◆Thành công: trả về số lượng stream được đóng.

◆Thất bại: trả về **EOF**.



```
FILE* fp1 = fopen("taptin1.txt", "rt");  
FILE* fp2 = fopen("taptin2.txt", "wt");
```

...

```
fcloseall();
```



# Con trỏ chỉ vị (position indicator)

- Khái niệm
  - Được tạo tự động khi mở tập tin.
  - Xác định nơi diễn ra việc đọc/ghi trong tập tin
- Vị trí con trỏ chỉ vị
  - Khi tập tin chưa mở: ở đầu tập tin (giá trị 0).
  - Khi mở tập tin:
    - Ở cuối tập tin khi mở để chèn (mode **a** hay **a+**)
    - Ở đầu tập tin (hay giá trị 0) khi mở với các mode khác (**w**, **w+**, **r**, **r+**).



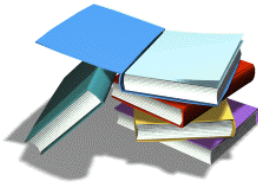
# Truy xuất tuần tự & ngẫu nhiên

- Truy xuất tuần tự (sequentially access)
  - Phải đọc/ghi dữ liệu từ vị trí con trỏ chỉ vị đến vị trí  $n-1$  trước khi đọc dữ liệu tại vị trí  $n$ .
  - Không cần quan tâm đến con trỏ chỉ vị do con trỏ chỉ vị tự động chuyển sang vị trí kế tiếp sau thao tác đọc/ghi dữ liệu.
- Truy xuất ngẫu nhiên (random access)
  - Có thể đọc/ghi tại vị trí bất kỳ trong tập tin mà không cần phải đọc/ghi toàn bộ dữ liệu trước đó => quan tâm đến con trỏ chỉ vị.



# Hàm đặt lại vị trí con trỏ chỉ vị

`void rewind(FILE *fp)`



Đặt lại vị trí con trỏ chỉ vị về đầu (byte 0) tập tin **fp**.



◆ Không



```
FILE* fp = fopen("taptin.txt", "w+");  
fprintf(fp, "0123456789");  
rewind(fp);  
fprintf(fp, "*****");
```



# Hàm tái định vị con trỏ chỉ vị

`int fseek(FILE *fp, long offset, int origin)`



Đặt vị trí con trỏ chỉ vị trong stream `fp` với vị trí `offset` so với cột mốc `origin` (`SEEK_SET` hay `0`: đầu tập tin; `SEEK_CUR` hay `1`: vị trí hiện tại; `SEEK_END` hay `2`: cuối tập tin)



- ◆Thành công: trả về `0`.
- ◆Thất bại: trả về giá trị khác `0`.



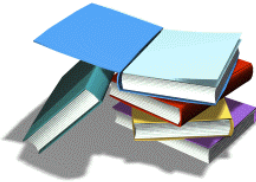
```
FILE* fp = fopen("taptin.txt", "w+");  
fseek(fp, 0L, SEEK_SET); // ⇔ rewind(fp);  
fseek(fp, 0L, SEEK_END); // cuối tập tin  
fseek(fp, -2L, SEEK_CUR); // lùi lại 2 vị trí
```





# Hàm xác định vị trí con trỏ chỉ vị

long **ftell**(FILE \***fp**)



Hàm trả về vị trí hiện tại của con trỏ chỉ vị (tính từ vị trí đầu tiên của tập tin, tức là 0) của stream **fp**.



◆Thành công: trả về vị trí hiện tại của con trỏ chỉ vị.

◆Thất bại: trả về **-1L**.



```
FILE* fp = fopen("taptin.txt", "rb");  
fseek(fp, 0L, SEEK_END);  
long size = ftell(fp);  
printf("Kích thước tập tin là %ld\n", size);
```



# Lập trình thao tác trên tập tin văn bản thô



# Xem giáo trình NMLT (323-331)

- Đọc nội dung tập tin có sẵn.
- Tạo tập tin để ghi dữ liệu.
- Chỉnh sửa nội dung tập tin có sẵn.
- Ghi thêm, ghép nội dung các tập tin.
- Những thao tác hỗ trợ
  - Kiểm tra tập tin đã có sẵn chưa.
  - Kiểm tra tập tin cho phép sửa hay không.
  - Xóa, đổi tên tập tin.



# Sử dụng tập tin văn bản thô để lưu dữ liệu chương trình



# Xem giáo trình NMLT (332-352)

- Văn bản thô thường được sử dụng để lưu dữ liệu của chương trình
  - Dữ liệu ký tự và chuỗi.
  - Dữ liệu số trong tập tin văn bản.
  - Dữ liệu phức hợp trong tập tin văn bản.





# Các vấn đề mở rộng kiến thức nghề nghiệp





# Tìm hiểu thêm

- Cấu trúc của tập tin HTML
- Các dạng khác về tập tin văn bản thô có cấu trúc (RTF, TEX, XML...)
- Cấu trúc tập tin văn bản thô mở rộng



# Thuật ngữ và bài đọc thêm tiếng Anh



# Thuật ngữ tiếng Anh

- **binary file**: tập tin nhị phân.
- **end of file, EOF character**: ký hiệu kết thúc tập tin.
- **file processing**: xử lý tập tin.
- **Hypertext Markup Language**: ngôn ngữ HTML dùng để lưu trữ tập tin văn bản thô có cấu trúc được dùng cho các trình duyệt web.
- **line**: dòng (văn bản).
- **multi-byte character**: ký tự được lưu trữ bằng nhiều byte.
- **random access**: truy xuất ngẫu nhiên.
- **read only**: chỉ được phép đọc.
- **record (danh từ)**: mẫu tin.
- **Rich Text Format**: định dạng RTF, lưu trên đĩa dưới dạng các văn bản ASCII có cấu trúc, dùng để lưu trữ các văn bản phức hợp có cả thông tin định dạng lẫn bản biểu, hình ảnh.
- **sequentially access**: truy xuất tuần tự.



# Thuật ngữ tiếng Anh

- ***stream***: khái niệm dùng trong lập trình bằng ngôn ngữ C/C++, chỉ dòng dữ liệu nhập xuất, được dùng khi đọc ghi dữ liệu tập tin hay thiết bị nhập xuất.
- ***tab***: ký tự tab (tương đương với một số khoảng trống khi hiển thị).
- ***text file, plain text, ASCII text (hay ASCII text)***: nói chung về định dạng văn bản đơn giản được soạn bằng các trình soạn thảo thông dụng của các hệ điều hành.
- ***Unicode text, UTF-8 text***: các định dạng văn bản thô dạng mở rộng, mỗi ký tự chiếm nhiều byte lưu trữ trong bộ nhớ hay trên đĩa.



# Bài đọc thêm tiếng Anh

- **Theory and Problems of Fundamentals of Computing with C++**, John R. Hubbard, Schaum's Outlines Series, McGraw-Hill, 1998.



