Chương 8: Lập trình với tập tin văn bảng thô

TS. Nguyễn Sơn Hoàng Quốc

cuu duong than cong . com

Nhập môn lập trình

CÁC DẠNG TẬP TIN
THEO GÓC ĐỘ NGƯỜI LẬP TRÌNH

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.

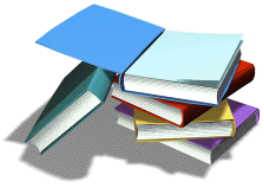
CÁC THAO TÁC TRÊN TẬP TIN

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!");
```

Đối số mở tập tin (mode)

Đối số	Ý nghĩa
b	Mở tập tin kiểu nhị phân (binary)
t	Mở tập tin kiểu văn bản (text) (mặc định)
r	Mở tập tin chỉ để đọc dữ liệu từ tập tin. Trả về NULL nếu không tìm thấy tập tin.
w	Mở tập tin chỉ để ghi 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 chỉ để thêm (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.

Bài tập ứng dụng

- Ví dụ

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

- Áp dụng
  1. Mở tập tin chỉ để đọc có sẵn tại đường dẫn "D:\\abc.txt". Nếu không có báo lỗi
  2. Mở tập tin "abc.txt" trong thư mục Data của chương trình để ghi. Nếu chưa có tạo tập tin tương ứng.
  3. Ghi log chương trình bằng cách mở để thêm dữ liệu tập tin "log.txt" trong thư mục Log của chương trình (tự tạo tập tin mới nếu chưa có).

Chương trình minh họa

- Mở tập tin chỉ để đọc có sẵn tại đường dẫn "D:\\abc.txt". Nếu không có báo lỗi
- Mã nguồn

```
1. FILE* fp = fopen("D:\\abc.txt", "r");
2. if (fp == NULL)
3.   printf("Khong mo duoc tap tin!");
```

Chương trình minh họa

- Mở tập tin "abc.txt" trong thư mục Data của chương trình để ghi. Nếu chưa có tạo tập tin tương ứng.
- Mã nguồn

```
1. FILE* fp = fopen("Data\\abc.txt", "w");
2. if (fp == NULL)
3.    printf("Khong mo duoc tap tin!");
```

Chương trình minh họa

- Ghi log chương trình bằng cách mở để thêm dữ liệu tập tin "log.txt" trong thư mục Log của chương trình (tự tạo tập tin mới nếu chưa có).
- Mã nguồn

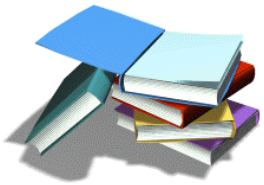
```
1. FILE* fp = fopen(“Log\\abc.txt", "a");
2. if (fp == NULL)
3.    printf("Khong mo duoc tap tin!");
```

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 *fnt, …)



Ghi dữ liệu có chuỗi định dạng fnt (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);
```

Ví dụ minh họa

- Ví dụ:

```
1. int i = 2912; int c = 'P'; float f = 17.06;
2. FILE* fp = fopen("taptin.txt", "wt");
3. if (fp != NULL)
4.    fprintf(fp, "%d %c %.2f\n", i, c, f);
```

- Áp dụng
  - Ghi log chương trình bằng cách thêm log "Ban da van tin tai khoan vao luc 15:30 ngay 17 thang 3 nam 2016" vào tập tin "log.txt" trong thư mục Log của chương trình.

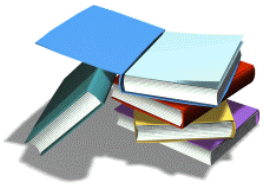
Chương trình minh họa

- Ghi log chương trình bằng cách thêm log "Ban da van tin tai khoan vao luc 15:30 ngay 17 thang 3 nam 2016" vào tập tin "log.txt" trong thư mục Log của chương trình.
- Mã nguồn


```
1. FILE* fp = fopen("Log\\log.txt", "at");
2. if (fp == NULL)
3. {
4.     printf("Khong mo duoc tap tin!");
5. }
6. else
7.     fprintf(fp, "Ban da van tin tai khoan vao luc 15:30
       ngay 17 thang 3 nam 2016\n");
```

Hàm nhập theo định dạng

int fscanf(FILE *fp, char *fnt, ...)



Đọc dữ liệu có chuỗi định dạng fnt (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);
```

Bài tập minh họa

- Ví dụ:

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

- Minh họa:
  - Đọc và xuất ra màn hình mảng một chiều các số nguyên được ghi trong tập tin input.txt được mô tả như sau:
    - Dòng thứ nhất là số phần tử n
    - Dòng thứ hai là n số, mỗi số cách nhau một khoảng trắng
  - Ví dụ

    5

    2 3 4 6 3

Chương trình minh họa

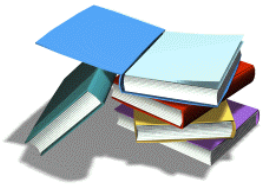
```
1.  int a[100], n;

2.  FILE* fp = fopen("input.txt", "r");
3.  if (fp == NULL)
4.  {
5.     printf("Khong mo duoc tap tin!");
6.     return -1;
7.  }

8.  \\ Nhap mang
9.  fscanf(fp, "%d", &n);
10. for (int i = 0; i < n; i++)
11.    fscanf(fp, "%d", &a[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);
```

Bài tập ứng dụng

- Ví dụ:

```
1. char ch;
2. FILE* fp = fopen("taptin.txt", "rt");
3. if (fp != NULL)
4.    ch = getc(fp); \\ ⇔ ch = fgetc(fp);
```

- Minh họa:
  - Đếm số từ trong tập tin "data.txt" biết rằng mỗi từ cách nhau bởi 01 khoảng trắng hoặc 01 dấu xuống dòng

Chương trình minh họa

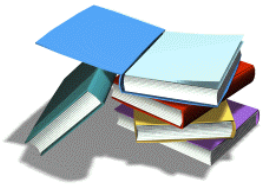
```
1.  int count = 0;
2.  char ch;

3.  FILE* fp = fopen("data.txt", "r");
4.  if (fp == NULL){
5.     printf("Khong mo duoc tap tin!");
6.     return -1;
7.  }

8.  ch = fgetc(fp);
9.  while (ch != EOF){
10.    if (ch == ' ' || ch == '\n')
11.           count++;
12.    ch = fgetc(fp);
13. }
14. printf("%d", count+1);
```

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", "wt");
if (fp != NULL)
        putc('a', fp); \\ hoặc fputc('a', fp);
```

Bài tập ứng dụng

- Ví dụ:

```
1. FILE* fp = fopen("taptin.txt", "wt");
2. if (fp != NULL)
3.    putc('a', fp); \\ ⇔ fputc('a', fp);
```

- Áp dụng:
  - Viết chương trình sao chép một tập tin cho trước sử dụng hàm putc.

Chương trình minh họa

```
1. char ch;

2. FILE* fi = fopen("input.txt", "r");
3. FILE* fo = fopen("output.txt", "w");

4. \\ chep file
5. while ((ch = getc(fi)) != EOF)
6.    putc(ch, fo);
```

Bài tập

- Bài 1: Viết chương trình ghi 3 số nguyên a, b, c được nhập từ bàn phím vào một tập tin.
- Bài 2: Viết chương trình đọc 3 số nguyên a, b, c từ một tập tin, sau đó giải phương trình $ax^2 + bx + c = 0$ rồi ghi kết quả vào một tập tin khác.
- Bài 3: Viết chương trình đọc n số nguyên từ một tập tin cho trước, sau đó sắp xếp tăng dần rồi ghi kết quả vào 1 tập tin khác. Ví dụ:

  4 → 4

  2 5 1 4 → 1 2 4 5

Bài tập thực hành

- Bài 4: Viết chương trình ghi các dòng văn bản được nhập từ bàn phím lên tập tin.
- Bài 5: Viết chương trình in nội dung một tập tin lên màn hình.
- Bài 6: Viết chương trình đếm số ký tự chữ cái của tập tin và xuất kết quả ra một tập tin khác.
- Bài 7: Viết chương trình đếm số từ của tập tin và xuất kết quả ra một tập tin khác.
- Bài 8: Viết chương trình đếm số lần lặp lại của một từ trong một tập tin.

Bài tập thực hành

- Bài 9: Viết chương trình mở tập tin văn bản đã có trên đĩa, sao chép nó thành một tập tin văn bản mới với điều kiện là các chữ thường đổi thành chữ hoa, tất cả các ký tự khác không đổi.
- Bài 10: Viết chương trình ghép 2 tập tin văn bản, nội dung tập tin thứ hai được ghép sau tập tin thứ nhất.

CÁC VẤN ĐỀ MỞ RỘNG KIẾN THỨC NGHỀ NGHIỆP

Tìm hiểu thêm

- Kiến trúc thư viện nhập xuất trong C++
- Cấu trúc của một vài tập tin cơ sở dữ liệu
- Cấu trúc của một số tập tin ảnh
- Tập tin XML và việc lập trình

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)***: mẩu 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

- ***string*:** chuỗi ký tự.
- ***string elements***: các phần tử (ký tự) nằm trong một chuỗi.
- ***string functions***: các hàm thao tác trên chuỗi ký tự.
- ***string operator***: phép toán thao tác trên chuỗi ký tự.
- ***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, ASNI 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 dạn 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.