Nội dung

1. Khái niệm dòng (stream)
2. Khái niệm và phân loại tập tin
3. Các thao tác xử lý căn bản
4. Một số hàm quản lý tập tin

Nhập xuất

- Khái niệm:
  - C lưu dữ liệu (biến, mảng, cấu trúc, …) trong bộ nhớ RAM.
  - Dữ liệu được nạp vào RAM và gửi ra ngoài chương trình thông qua các thiết bị (device).
    - Thiết bị nhập (input device): bàn phím, con chuột
    - Thiết bị xuất (output device): màn hình, máy in
    - Thiết bị vừa nhập vừa xuất: tập tin
  - Các thiết bị đều thực hiện mọi xử lý thông qua các dòng (stream).

Stream (dòng)

- Khái niệm:
  - Là môi trường trung gian để giao tiếp (nhận/ gửi thông tin) giữa chương trình và thiết bị.
  - →Muốn nhận/gửi thông tin cho một thiết bị ta sẽ gửi thông tin cho stream nối với thiết bị đó (độc lập thiết bị).Các thiết bị đều thực hiện mọi xử lý thông qua các dòng (stream).
  - Stream là dãy byte dữ liệu:
    - "Chảy" vào chương trình gọi là stream nhập.
    - "Chảy" ra chương trình gọi là stream xuất.

Stream (dòng)

- Phân loại:
  - Stream văn bản (text)
    - Chỉ chứa các ký tự.
    - Tổ chức thành từng dòng, mỗi dòng tối đa 255 ký tự, kết thúc bởi ký tự cuối dòng '\0' hoặc ký tự sang dòng mới '\n'.
  - Stream nhị phân (binary)
    - Chứa các byte.
    - Được đọc và ghi chính xác từng byte.
    - Xử lý dữ liệu bất kỳ, kể cả dữ liệu văn bản.
    - Được sử dụng chủ yếu với các tập tin trên đĩa.

Stream (dòng)

- Các stream chuẩn định nghĩa sẵn:

Tên	Stream	Thiết bị tương ứng
stdin	Nhập chuẩn	Bài phím
stdout	Xuất chuẩn	Màn hình
stderr	Lỗi chuẩn	Màn hình
stdprn (MS-DOS)	In chuẩn	Máy in (LPT1:)
stdaux (MS-DOS)	Phụ chuẩn	Cổng nối tiếp COM 1:

- Ví dụ (hàm fprintf xuất ra stream xác định)
  - Xuất ra màn hình: fprintf(stdout, “Hello”);
  - Xuất ra máy in: fprintf(stdprn, “Hello”);
  - Xuất ra thiết bị báo lỗi: fprintf(stderr, “Hello”);
  - Xuất ra tập tin (stream fp): fprintf(fp, “Hello”);

Tập tin

- Nhu cầu:
  - Dữ liệu giới hạn và được lưu trữ tạm thời
    - Nhập: gõ từ bàn phím.
    - Xuất: hiển thị trên màn hình.
    - Lưu trữ dữ liệu: trong bộ nhớ RAM.

    →Mất thời gian, không giải quyết được bài toán với số dữ liệu lớn.
  - Cần một thiết bị lưu trữ sao cho dữ liệu vẫn còn khi kết thúc chương trình, có thể sử dụng nhiều lần và kích thước không hạn chế.

Tập tin

Khái niệm:

- Tập hợp thông tin (dữ liệu) được tổ chức theo một dạng nào đó với một tên xác định.
- Một dãy byte liên tục (ở góc độ lưu trữ).
- Được lưu trữ trong các thiết bị lưu trữ ngoài như đĩa mềm, đĩa cứng, USB…
  - Vẫn tồn tại khi chương trình kết thúc.
  - Kích thước không hạn chế (tùy vào thiết bị lưu trữ)
- Cho phép đọc dữ liệu (thiết bị nhập) và ghi dữ liệu (thiết bị xuất).

Tập tin

- Phân loại:
  - Theo người sử dụng: quan tâm đến nội dung tập tin nên sẽ phân loại theo phần mở rộng

    → .EXE, .COM, .CPP, .DOC, .PPT, …
  - Theo người 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 trong C

    → tập tin kiểu văn bản (ứng với stream văn bản) và tập tin kiểu nhị phân (ứng với stream nhị phân).

Phân loại tập tin

- Tập tin kiểu văn bản (stream văn bản):
  - Dãy các dòng kế tiếp nhau.
  - Mỗi dòng dài tối đa 255 ký tự và kết thúc bằng ký hiệu cuối dòng (end_of_line).
  - Dòng không phải là một chuỗi vì không được kết thúc bởi ký tự '\0'.
  - Khi ghi '\n' được chuyển thành cặp ký tự CR (về đầu dòng, mã ASCII 13) và LF (qua dòng, mã ASCII 10).
  - Khi đọc thì cặp CR-LF được chuyển thành '\n'.

Phân loại tập tin

- Tập tin kiểu nhị phân (stream nhị phân):
  - Dữ liệu được đọc và ghi một cách chính xác, không có sự chuyển đổi nào cả.
  - Ký tự kết thúc chuỗi '\0' và end_of_line không có ý nghĩa là cuối chuỗi và cuối dòng mà được xử lý như mọi ký tự khác.

Quy tắc đặt tên tập tin

- Tên (name):
  - Bắt buộc phải có.
  - Hệ điều hành MS-DOS: dài tối đa 8 ký tự.
  - Hệ điều hành Windows: dài tối đa 128 ký tự.
  - Gồm các ký tự A đến Z, số 0 đến 9, ký tự khác như #, $, %, ~, ^, @, (,), !, _, khoảng trắng.
- Mở rộng (extension):
  - Không bắt buộc.
  - Thường có 3 ký tự.
  - Thường do chương trình ứng dụng tạo tập tin tự đặt

Định vị tập tin

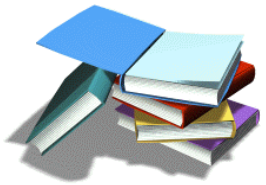
- Đường dẫn:
  - Chỉ đến một tập tin không nằm trong thư mục hiện hành. Ví dụ: c:\data\list.txt chỉ tập tin list.txt nằm trong thư mục data của ổ đĩa C.
  - Trong chương trình, đường dẫn này được ghi trong chuỗi như sau: "c:\\data\\list.txt"
  - Dấu "\" biểu thị ký tự điều khiển nên để thể hiện nó ta phải thêm một dấu '\' ở trước. Nhưng nếu chương trình yêu cầu nhập đường dẫn từ bàn phím thì chỉ nhập một dấu '\'.

Quy trình thao tác với tập tin

- 1. Mở tập tin: tạo một stream nối kết với tập tin cần mở, stream được quản lý bởi biến con trỏ đến cấu trúc FILE
  - Cấu trúc được định sẵn trong STDIO.H
  - Các thành phần của cấu trúc này được dùng trong các thao tác xử lý tập tin.
- 2. Sử dụng tập tin (sau khi đã mở được tập tin)
  - Đọc dữ liệu từ tập tin đưa vào chương trình.
  - Ghi dữ liệu từ chương trình lên tập tin.
- 3. Đóng tập tin (sau khi sử dụng xong).

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)

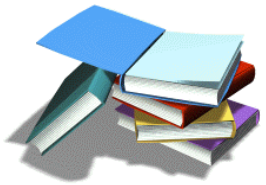
Đối số	Ý nghĩa
b	Mở tập tin kiểu nhị phân (binary)
t	Mở tập tin kiểu văn bản (text) (mặc định)
r	Mở tập tin chỉ để đọc dữ liệu từ tập tin. Trả về NULL nếu không tìm thấy tập tin.
w	Mở tập tin chỉ để ghi dữ liệu vào tập tin. Tập tin sẽ được tạo nếu chưa có, ngược lại dữ liệu trước đó sẽ bị xóa hết.
a	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.

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, ...)



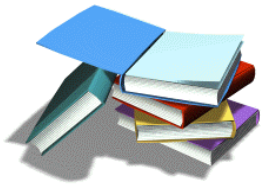
Ghi dữ liệu có chuỗi định dạng fnt (giống hàm printf) vào stream fp.
Nếu fp là stdout thì hàm giống printf

- 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; char 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 *fnt, ...)



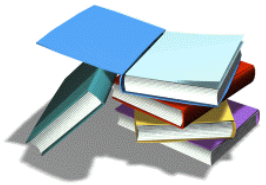
Đọc dữ liệu có chuỗi định dạng fnt (giống hàm scanf) từ stream fp.
Nếu fp là stdin thì hàm giống printf.

- 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);
```

Hàm nhập ký tự

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



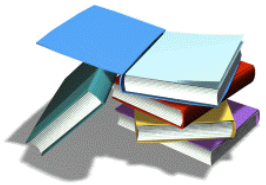
Đọc một ký tự từ stream fp.
getc là macro còn fgetc là phiên bản hàm của macro getc.

- 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)



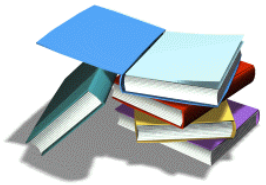
Đọc một dãy ký tự từ stream fp vào vùng nhớ str, kết thúc khi đủ n-1 ký tự hoặc gặp ký tự xuống dòng.

- 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)



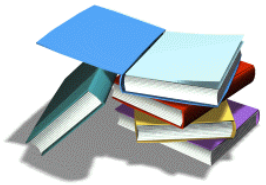
Ghi ký tự ch vào stream fp.
putc là macro còn fputc là phiên bản hàm của macro putc.

- 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))



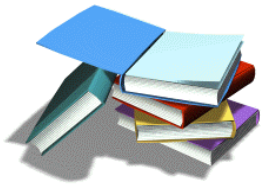
Ghi chuỗi ký tự str vào stream fp. Nếu fp là stdout thì fputs giống hàm puts, nhưng puts ghi ký tự xuống dòng.

- 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)



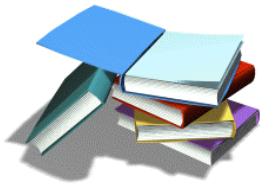
Ghi count mẫu tin có kích thước mỗi mẫu tin là size (byte) từ vùng nhớ buf vào stream fp (theo kiểu nhị phân).

- 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)



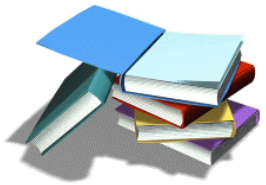
Đọc count mẫu tin có kích thước mỗi mẫu tin là size (byte) vào vùng nhớ buf từ stream fp (theo kiểu nhị phân).

- 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", "rb");
if (fp != NULL)
        fread(a, sizeof(int), 3, fp);
```

Hàm đóng tập tin xác định

int fclose(FILE *fp)



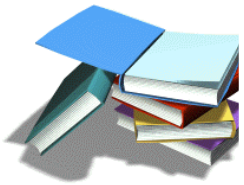
Đóng stream fp.
Dữ liệu trong stream fp sẽ được "vét" (ghi hết lên đĩa) trước khi đóng.

- 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, 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)

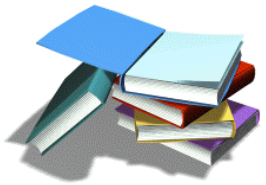
- Khái niệm:
  - Được tạo tự động khi mở tập tin.
  - Xác định nơi diễn ra việc đọc/ghi trong tập tin.
- Vị trí con trỏ chỉ vị
  - Khi tập tin chưa mở: ở đầu tập tin (giá trị 0).
  - Khi mở tập tin:
    - Ở cuối tập tin khi mở để chèn (mode a hay a+)
    - Ở đầu tập tin (hay giá trị 0) khi mở với các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



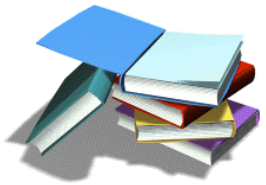
Đặt lại vị trí con trỏ chỉ vị về đầu (byte 0) tập tin fp.

- 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, ing origin)



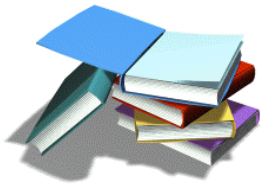
Đặt vị trí con trỏ chỉ vị trong stream fp với vị trí offset so với cột mốc origin (SEEK_SET hay 0: đầu tập tin; SEEK_CUR hay 1: vị trí hiện tại; SEEK_END hay 2: cuối tập tin)

- 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

long ftell(FILE *fp)



Hàm trả về vị trí hiện tại của con trỏ chị 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("Kich thuoc tap tin la %ld\n", size);
```

Dấu hiệu kết thúc tập tin

- Khi đã biết kích thước tập tin
  - Sử dụng fwrite để lưu n mẫu tin

    ➔kích thước = n * sizeof(1 mẫu tin);
  - Sử dụng hàm fseek kết hợp hàm ftell.
- Khi chưa biết kích thước tập tin
  - Hằng số EOF (=-1) (chỉ cho tập tin văn bản)

    ➔while ((c = fgetc(fp)) != EOF) …
  - Hàm int feof(FILE *fp) (cho cả 2 kiểu tập tin)

    ➔trả về số 0 nếu chưa đến cuối tập tin

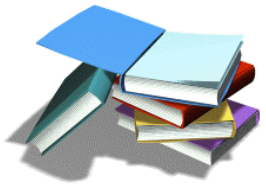
    ➔trả về số khác 0 nếu đã đến cuối tập tin.

Các hàm quản lý tập tin

- Hàm nhập xuất tập tin (File I/O function) là các đã đề cập phần trước:
  - Mở và đóng tập tin: fopen, fclose
  - Nhập/Xuất tập tin:
    - Theo định dạng: fprintf, fscanf
    - Từng ký tự hay chuỗi: fputc, fputs, fgetc, fgets
    - Trực tiếp từ bộ nhớ: fwrite, fread
- Hàm quản lý tập tin (File-Management function)
  - Xóa tập tin: remove
  - Đổi tên tập tin: rename

Hàm xóa tập tin

int remove(const char *filename)



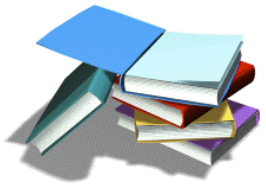
Xóa tập tin xác định bởi filename.

- Thành công: trả về 0.
- Thất bại: trả về -1.

```
if (remove("c:\\vc.txt") == 0)
        printf("Tap tin vc.txt da bi xoa!");
else
        printf("Ko xoa duoc tap tin vc.txt!");
```

Hàm đổi tên tập tin

int rename(const char *oldname, const char *newname)



Đổi tên tập tin oldname thành newname. Hai tập tin phải cùng ổ đĩa nhưng không cần thiết phải cùng thư mục (có thể sử dụng để di chuyển hay sao chép tập tin).

- Thành công: trả về 0.
- Thất bại: trả về -1.

```
if (rename("c:\\a.txt", "c:\\BT\b.cpp") == 0)
        printf("Doi ten tap tin thanh cong");
else
        printf("Doi ten tap tin that bai");
```

Bài tập lý thuyết

- Bài 1: Sự khác nhau giữa stream kiểu văn bản và stream kiểu nhị phân?
- Bài 2: Cần phải làm gì trước khi muốn truy xuất tập tin?
- Bài 3: Khi mở tập tin bằng fopen, ta cần phải xác định thông tin nào và hàm sẽ trả về cái gì?
- Bài 4: Cách xác định cuối tập tin trong kiểu văn bản và kiểu nhị phân?
- Bài 5: Con trỏ chỉ vị là gì và cách thay đổi nó?

Bài tập lý thuyết

- Bài 6: Trình bày hai cách khác nhau để chuyển con trỏ chỉ vị về đầu tập tin fp.

Bài tập thực hành

- Ví dụ 1: Minh họa việc đọc từ bàn phím và ghi chúng vào một tập tin cho đến khi người dùng nhấn nhập ký tự $.

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    FILE *fp;
    char ch;
    fp=fopen("test.txt", "w");
    if(fp==NULL)
    {
        printf("Cannot open file.\n");
        exit(1);
    }
    do
    {
        ch = getchar();
        putc(ch, fp);
    }while (ch != '$');
    fclose(fp);
}
```

Bài tập thực hành

- Ví dụ 2: Minh họa việc đọc từ một file văn bản và xuất chúng ra màn hình.

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    FILE *fp;
    char ch;
    fp=fopen("test.txt", "r");
    if(fp==NULL)
    {
        printf ("Cannot open file.\n");
        exit(1);
    }
    ch = getc(fp); // read one character
    while (ch!=EOF)
    {
        putchar(ch); // print on screen
        ch = getc(fp);
    }
    fclose(fp);
}
```

Bài tập thực hành

- Ví dụ 3: Chương trình sau minh họa hàm fputs(). Nó đọc các chuỗi từ bàn phím và viết chúng đến file tên teststr.txt. Để kết thúc chương trình, nhập một dòng trống

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main(void)
{
    char str[80];
    FILE *fp;
    if((fp = fopen("teststr.txt", "w"))==NULL)
    {
        printf ("Cannot open file.\n");
        exit(1);
    }
    do
    {
        printf("Enter a string (CR to quit):\n");
        gets(str);
        strcat(str, "\n"); /* add a newline */
        fputs(str, fp);
    } while(*str!='\n');
}
```

Bài tập thực hành

- Ví dụ 4: Nhập các mẫu tin nhân viên vào tập tin employee.dat

```c
#include <stdio.h>
typedef struct EMPLOYEE
{
    char name[30];
    int age;
};
int main()
{
    FILE *fp;

    EMPLOYEE e;
    char more = 'Y';
    fp = fopen("employee.dat", "wb");
    if(fp == NULL)
    {
        printf("Cannot open file");
        return 0;
    }
    while(more == 'Y'|| more == 'y')
    {
        printf("\nEnter name and age: ");
        fflush(stdin);
        gets(e.name);
        scanf("%d",&e.age);
        fwrite(&e, sizeof(e), 1, fp);
        printf("Input more (y/n)?: ");
        fflush(stdin);
        more = getchar();
    }
}
```

Bài tập thực hành

- Ví dụ 5: Xuất các mẫu tin nhân viên có trong tập tin employee.dat

```c
#include <stdio.h>
typedef struct EMPLOYEE
{
    char name[30];
    int age;
};
int main()
{
    FILE *fp;

    EMPLOYEE e;
    char more = 'Y';

    fp = fopen("employee.dat", "rb");
    if(fp == NULL)
    {
        printf("Cannot open file");
        return 0;
    }
    while(fread(&e, sizeof(e), 1, fp) == 1)
    {
        printf("\nName= %s", e.name);
        printf("\tAge= %d",e.age);
    }
    fclose(fp);
}
```