HỌC VIỆN KTQS
KHOA CÔNG NGHỆ THÔNG TIN

Chương 9
Kiểu dữ liệu cấu trúc, file

Học phần: LẬP TRÌNH CƠ BẢN

Tài liệu tham khảo

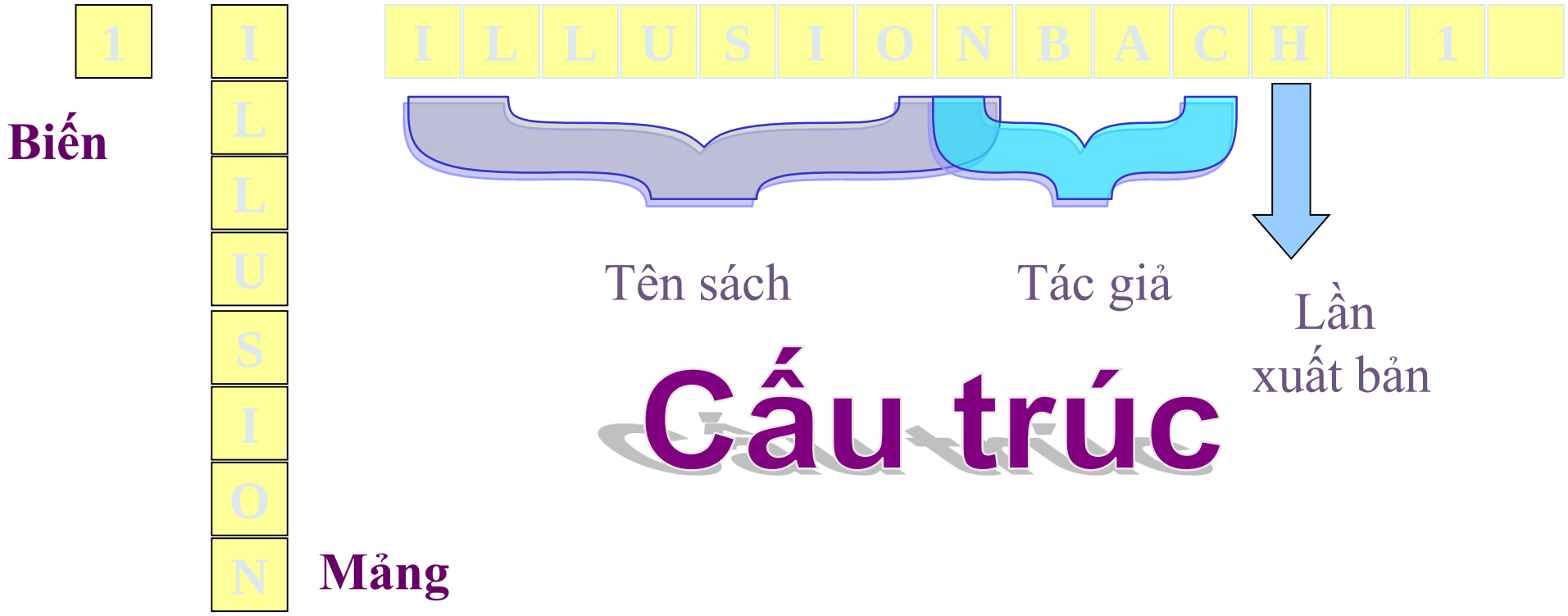
- Kỹ thuật lập trình C: cơ sở và nâng cao, Phạm Văn Ất, Nhà xuất bản KHKT – Chương 7
- Kỹ thuật lập trình C: cơ sở và nâng cao, Phạm Văn Ất, Nhà xuất bản KHKT – Chương 10.
- The C programming language 2nd Edition, Brian Kernighan and Dennis Ritchie, Prentice Hall Software Series – Chương 7.

Nội dung bài

- Khái niệm kiểu dữ liệu có cấu trúc
- Khai báo các biến kiểu cấu trúc
- Khởi tạo biến cấu trúc
- Sử dụng biến cấu trúc
- Sử dụng mảng các cấu trúc
- Khái niệm file
- Các thao tác với file

Khái niệm cấu trúc

- Một cấu trúc bao gồm các mẫu dữ liệu (không nhất thiết cùng kiểu) được nhóm lại với nhau.



Khái niệm cấu trúc (tiếp)

- Việc định nghĩa cấu trúc sẽ tạo ra kiểu dữ liệu mới cho phép người dùng sử dụng chúng để khai báo các biến kiểu cấu trúc .
- Các biến trong cấu trúc được gọi là các phần tử của cấu trúc hay thành phần của cấu trúc
- Ví dụ:

```
struct cat {
        char bk_name [25];
        char author [20];
        int edn;
        float price;
};
```

Khai báo biến cấu trúc

- Khi một cấu trúc đã được định nghĩa, chúng ta có thể khai báo một hoặc nhiều biến sử dụng kiểu này.
- Ví dụ: **struct cat books1;**
- Câu lệnh này sẽ dành đủ vùng nhớ để lưu trữ tất cả các mục trong một cấu trúc.


```
struct cat {
        char bk_name[25];
        char author[20];
        int edn;
        float price;
} books1, books2;
```

```
struct cat books1, books2;
```

hoặc

```
struct cat books1;
struct cat books2;
```

Khởi tạo cấu trúc

- Giống như các biến khác và mảng, các biến kiểu cấu trúc có thể được khởi tạo tại thời điểm khai báo

```
struct employee
    {    int no;
         char name [20];
    };
```

- Các biến **emp1** và **emp2** có kiểu **employee** có thể được khai báo và khởi tạo như sau:

```
struct employee emp1 = {346, “Abraham”};
struct employee emp2 = {347, “John”};
```

Truy cập phần tử của cấu trúc

- Các phần tử của cấu trúc được truy cập thông qua việc sử dụng **toán tử chấm** (.), toán tử này còn được gọi là **toán tử thành viên - membership**.
- Cú pháp:

  structure_name.element_name

- Ví dụ:

  scanf(“%s”, books1.bk_name);

Gán sử dụng cấu trúc

- Có thể sử dụng câu lệnh gán đơn giản để gán giá trị của một biến cấu trúc cho một biến khác có cùng kiểu
- Chẳng hạn, nếu **books1** và **books2** là các biến cấu trúc có cùng kiểu, thì câu lệnh sau là hợp lệ

books2 = books1;

Sao chép cấu trúc

- Do kiểu dữ liệu cấu trúc là phức tạp nên trong một số trường hợp không thể dùng câu lệnh gán trực tiếp, thì có thể sử dụng hàm **memcpy()**, đây là hàm có sẵn trong thư viện lập trình
- Cú pháp:

  memcpy (char * destn, char &source, int nbytes);
- Ví dụ:

  memcpy (&books2, &books1, sizeof(struct cat));

Cấu trúc lồng nhau

- Một cấu trúc có thể lồng trong một cấu trúc khác. Tuy nhiên, một cấu trúc không thể lồng trong chính nó.

```
struct issue     {
    char borrower [20];
    char dt_of_issue[8];
    struct cat books;
}issl;
```

- Việc truy cập vào các phần tử của cấu trúc này tương tự như với cấu trúc bình thường khác,

issl.borrower

- Để truy cập vào phần tử của cấu trúc cat là một phần của cấu trúc issl ,

issl.books.author

Mảng cấu trúc

- Một kiểu cấu trúc phải được định nghĩa trước, sau đó một biến mảng có kiểu cấu trúc tương ứng mới được khai báo
- Ví dụ: **struct cat books[50];**
- Để truy cập vào thành phần author của phần tử thứ tư của mảng **books** ta làm như sau:

  books[4].author

Khởi tạo các mảng cấu trúc

- Mảng cấu trúc được khởi tạo bằng cách liệt kê danh sách các giá trị phần tử của nó dưới dạng liệt kê
- Ví dụ:

```
struct unit {
        char ch;
        int i;
};
struct unit series [3] =
        {{‘a’, 100}{‘b’, 200}{‘c’, 300}};
```

Con trỏ đến cấu trúc

- Con trỏ cấu trúc được khai báo bằng cách đặt dấu * trước tên của biến cấu trúc.
- Toán tử -> được dùng để truy cập vào các phần tử của một cấu trúc sử dụng một con trỏ
- Ví dụ:

```
struct cat *ptr_bk;
ptr_bk = &books;
printf("%s",ptr_bk->author);
```

- Con trỏ cấu trúc được truyền vào hàm, cho phép hàm thay đổi trực tiếp các phần tử của cấu trúc.

Từ khóa typedef

- Một kiểu dữ liệu có thể được định nghĩa bằng cách sử dụng từ khóa typedef
- Nó không tạo ra một kiểu dữ liệu mới, mà định nghĩa một tên mới cho một kiểu đã có.
- Cú pháp: **typedef type name;**
- Ví dụ: **typedef float deci;**
- typedef không thể sử dụng với *storage classes*

Ví dụ 1:

- Xây dựng cấu trúc phân số gồm Tử số, mẫu số và các hàm nhập, in, tối giản]
- Hướng giải quyết
  - Tạo cấu trúc: struct phanso {int tuso; int maso;}
  - Viết hàm nhập từ bàn phím, hàm in ra màn hình;
  - Viết hàm tìm ước chung lớn nhất
  - Viết hàm in ra phân số tối giản

Khái niệm tệp tin

- Các biến khai báo khi lập trình như int, char, struct,.. sẽ được tổ chức trong bộ nhớ trong (RAM) của máy tính nên khi kết thúc chương trình thì dữ liệu cũng bị mất. Nếu muốn sử dụng lại các giá trị này, chúng ta bắt buộc phải nhập lại từ bàn phím.
- Kiểu tệp tin (file) cho phép lưu trữ dữ liệu ở bộ nhớ ngoài (Disk). Khi kết thúc chương trình thì dữ liệu vẫn còn do đó có thể sử dụng nhiều lần. Kiểu tệp tin có kích thước và các phần tử không hạn chế, chỉ phụ thuộc vào dung lượng của bộ nhớ ngoài.

Phân loại tệp tin

- Tệp tin định kiểu (Typed File): là loại tệp tin bao gồm nhiều phần tử có cùng kiểu: char, int, long, cấu trúc… và được lưu trữ trên đĩa dưới dạng một chuỗi các byte liên tục.
- Tệp tin không định kiểu (Untyped File): là loại tệp tin mà dữ liệu của chúng gồm các cấu trúc dữ liệu mà không quan tâm đến nội dung hoặc kiểu của nó, chỉ lưu ý đến các yếu tố vật lý của tệp tin như độ lớn và các yếu tố tác động lên tệp tin.

Ví dụ: tệp tin kiểu Text

- Tệp tin văn bản (Text File): là loại tệp tin dùng để ghi các ký tự lên bộ nhớ ngoài, các ký tự này được lưu trữ dưới dạng mã Ascii có 1 số đặc điểm sau:
  - Dữ liệu của tệp tin được lưu trữ thành các dòng, mỗi dòng được kết thúc bằng ký tự xuống dòng (new line), ký hiệu '\n'; ký tự này là sự kết hợp của 2 ký tự CR (Carriage Return - Về đầu dòng, mã Ascii là 13) và LF (Line Feed - Xuống dòng, mã Ascii là 10).
  - Tệp tin được kết thúc bởi ký tự EOF (End Of File) có mã Ascii là 26 (xác định bởi tổ hợp phím Ctrl +Z).
  - Tệp tin văn bản chỉ có thể truy xuất theo kiểu tuần tự.

Sử dụng biến loại tệp tin

- **Biến tệp tin:** là một biến thuộc kiểu dữ liệu tệp tin dùng để đại diện cho một tệp tin. Dữ liệu chứa trong một tệp tin được truy xuất qua các thao tác với thông số là biến tệp tin đại diện cho tệp tin đó.
- **Con trỏ tệp tin:** Khi một tệp tin được mở ra để làm việc, tại mỗi thời điểm, sẽ có một vị trí của tệp 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 tệp tin. Sau khi đọc/ghi xong dữ liệu, con trỏ sẽ chuyển dịch thêm một phần tử về phía cuối tệp tin. Sau phần tử dữ liệu cuối cùng của tệp tin là dấu kết thúc tệp tin EOF (End Of File).

Thao tác trên tệp tin

- **Khai báo biến tệp 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;
- **Mở tệp tin**
  - **Cú pháp**: FILE *fopen(char *Path, const char *Mode)
  - Trong đó:
    - Path: chuỗi chỉ đường dẫn đến tệp tin trên đĩa.
    - Type: chuỗi xác định cách thức mà tệp tin sẽ mở.

Tham số

- Các giá trị có thể của *Mode*:

Chế độ	Ý nghĩa
r	Mở tệp tin văn bản để đọc
w	Tạo ra tệp tin văn bản mới để ghi
a	Nối vào tệp tin văn bản
rb	Mở tệp tin nhị phân để đọc
wb	Tạo ra tệp tin nhị phân để ghi
ab	Nối vào tệp tin nhị phân
r+	Mở một tệp tin văn bản để đọc/ghi
w+	Tạo ra tệp tin văn bản để đọc ghi
a+	Nối vào hay tạo mới tệp tin văn bản để đọc/ghi
r+b	Mở ra tệp tin nhị phân để đọc/ghi
w+b	Tạo ra tệp tin nhị phân để đọc/ghi
a+b	Nối vào hay tạo mới tệp tin nhị phân

Một số thao tác khác tệp tin

- **Đóng tệp tin**
  - Hàm fclose() được dùng để đóng tệp tin được mở bởi hàm fopen(). Hàm này sẽ ghi dữ liệu còn lại trong vùng đệm vào tệp tin và đóng lại tệp tin.
  - **Cú pháp**: int fclose(FILE *f)
  - f là con trỏ tệp tin được mở bởi hàm fopen(). Giá trị trả về của hàm là 0 báo rằng việc đóng tệp tin thành công. Hàm trả về EOF nếu có xuất hiện lỗi. Có thể sử dụng hàm fcloseall() để đóng tất cả các tệp tin lại.
  - Cú pháp: int fcloseall()
  - Kết quả trả về của hàm là tổng số các tệp tin được đóng lại. Nếu không thành công, kết quả trả về là EOF.
- **Kiểm tra đến cuối tệp tin**
  - **Cú pháp**: int feof(FILE *f)
  - Ý nghĩa: Kiểm tra xem đã chạm tới cuối tệp tin hay chưa và trả về EOF nếu cuối tệp tin được chạm tới, ngược lại trả về 0.
- **Di chuyển con trỏ tệp tin về đầu tệp tin - Hàm rewind()**
  - Khi ta đang thao tác một tệp tin đang mở, con trỏ tệp tin luôn di chuyển về phía cuối tệp tin. Muốn cho con trỏ quay về đầu tệp tin như khi mở nó, ta sử dụng hàm *rewind()*.
  - Cú pháp: void rewind(FILE *f)

Truy cập tệp văn bản

- **Ghi dữ liệu lên tệp tin văn bản**
  - **Hàm putc()**
    - Hàm này được dùng để ghi một ký tự lên một tệp tin văn bản đang được mở để làm việc.
    - **Cú pháp: int putc(int c, FILE *f)**
      - tham số c chứa mã Ascii của một ký tự nào đó. Mã này được ghi lên tệp tin liên kết với con trỏ f. Hàm này trả về EOF nếu gặp lỗi.
  - **Hàm fputs()**
    - Hàm này dùng để ghi một chuỗi ký tự chứa trong vùng đệm lên tệp tin văn bản.
    - **Cú pháp: int puts(const char *buffer, FILE *f)**
      - buffer là con trỏ có kiểu char chỉ đến vị trí đầu tiên của chuỗi ký tự được ghi vào. Hàm này trả về giá trị 0 nếu buffer chứa chuỗi rỗng và trả về EOF nếu gặp lỗi.

Ghi dữ liệu vào tệp văn bản

- **Hàm fprintf()**
  - Hàm này dùng để ghi dữ liệu có định dạng lên tệp tin văn bản.
  - Cú pháp: fprintf(FILE *f, const char *format, varexpr)
    - 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 (,).

Định dạng	**Ý nghĩa**
%d	Ghi số nguyên
%[.số chữ số thập phân] f	Ghi số thực có <số chữ số thập phân> theo quy tắc làm tròn số.
%o	Ghi số nguyên hệ bát phân
%x	Ghi số nguyên hệ thập lục phân
%c	Ghi một ký tự
%s	Ghi chuỗi ký tự
%e hoặc %E hoặc %g hoặc %G	Ghi số thực dạng khoa học (nhân 10 mũ x)

Đọc dữ liệu từ tệp văn bản

- **Đọc dữ liệu từ tệp tin văn bản**
  - **Hàm getc()**
    - Hàm này dùng để đọc dữ liệu từ tệp tin văn bản đang được mở để làm việc.
    - Cú pháp: int getc(FILE *f)
    - Hàm này trả về mã Ascii của một ký tự nào đó (kể cả EOF) trong tệp tin liên kết với con trỏ f.
  - **Hàm fgets()**
    - Cú pháp: char *fgets(char *buffer, int n, FILE *f)
    - Hàm này được dùng để đọc một chuỗi ký tự từ tệp tin văn bản đang được mở ra và liên kết với con trỏ f cho đến khi đọc đủ n ký tự hoặc gặp ký tự xuống dòng '\n' hay gặp ký tự kết thúc EOF (ký tự này không được đưa vào chuỗi kết quả).
    - Trong đó:
      - buffer (vùng đệm): con trỏ có kiểu char chỉ đến cùng nhớ đủ lớn chứa các ký tự nhận được.
      - n: giá trị nguyên chỉ độ dài lớn nhất của chuỗi ký tự nhận được.
      - f: con trỏ liên kết với một tệp tin nào đó.
      - Ký tự NULL ('\0') tự động được thêm vào cuối chuỗi kết quả lưu trong vùng đệm.
      - Hàm trả về địa chỉ đầu tiên của vùng đệm khi không gặp lỗi và chưa gặp ký tự kết thúc EOF. Ngược lại, hàm trả về giá trị NULL.

Đọc dữ liệu từ tệp văn bản (tiếp)

Hàm fscanf()

Hàm này dùng để đọc dữ liệu từ tệp tin văn bản vào danh sách các biến theo định dạng.

Cú pháp: fscanf(FILE *f, const char *format, varlist)

Trong đó: format: chuỗi định dạng (giống hàm scanf()); varlist: danh sách các biến mỗi biến cách nhau dấu phẩy (,).

Ví dụ: Viết chương trình chép tệp tin D:\Baihat.txt ở trên sang tệp tin D:\Baica.txt.

```
#include<stdio.h>
#include<conio.h>
int main()
{
FILE *f1,*f2;
clrscr();
f1=fopen("D:\\Baihat.txt","rt");
f2=fopen("D:\\Baica.txt","wt");
if (f1!=NULL && f2!=NULL)
{
int ch=fgetc(f1);
while (! feof(f1))
{
fputc(ch,f2);
ch=fgetc(f1);
}
fcloseall();
}
getch();
return 0;
}
```

Ví dụ: Viết chương trình ghi chuỗi ký tự lên tệp tin văn bản D:\\Baihat.txt

```
#include<stdio.h>
#include<conio.h>
int main()
{
	FILE *f;
	clrscr();
	f=fopen("D:\\Baihat.txt","r+");
	if (f!=NULL)
	{
	fputs("Em oi Ha Noi pho.\n",f);
	fputs("Ta con em, mui hoang lan; ta con em, mui hoa sua.",f);
	fclose(f);
	}
	getch();
	return 0;
}
```

Làm việc với tệp nhị phân

- **Ghi dữ liệu lên tệp tin nhị phân - Hàm fwrite()**
  - **Cú pháp: size_t fwrite(const void *ptr, size_t size, size_t n, FILE *f)**
    - ptr: con trỏ chỉ đến vùng nhớ chứa thông tin cần ghi lên tệp tin.
    - n: số phần tử sẽ ghi lên tệp tin.
    - size: kích thước của mỗi phần tử.
    - f: con trỏ tệp tin đã được mở.
    - Giá trị trả về của hàm này là số phần tử được ghi lên tệp tin. Giá trị này bằng n trừ khi xuất hiện lỗi.

Đọc dữ liệu từ tệp nhị phân

- **Hàm fread()**
  - **Cú pháp: size_t fread(const void *ptr, size_t size, size_t n, FILE *f)**
    - ptr: con trỏ chỉ đến vùng nhớ sẽ nhận dữ liệu từ tệp tin.
    - n: số phần tử được đọc từ tệp tin.
    - size: kích thước của mỗi phần tử.
    - f: con trỏ tệp tin đã được mở.
    - Giá trị trả về của hàm này là số phần tử đã đọc được từ tệp tin. Giá trị này bằng n hay nhỏ hơn n nếu đã chạm đến cuối tệp tin hoặc có lỗi xuất hiện.

Con trỏ trong tệp nhị phân

- **Di chuyển con trỏ tệp tin - Hàm fseek()**
  - Việc ghi hay đọc dữ liệu từ tệp tin sẽ làm cho con trỏ tệp tin dịch chuyển một số byte, đây chính là kích thước của kiểu dữ liệu của mỗi phần tử của tệp tin.
  - Khi đóng tệp tin rồi mở lại nó, con trỏ luôn ở vị trí ngay đầu tệp tin. Nhưng nếu ta sử dụng kiểu mở tệp tin là "a" để ghi nối dữ liệu, con trỏ tệp tin sẽ di chuyển đến vị trí cuối cùng của tệp tin này.
  - Ta cũng có thể điều khiển việc di chuyển con trỏ tệp tin đến vị trí chỉ định bằng hàm fseek().
  - **Cú pháp: int fseek(FILE *f, long offset, int whence)**
    - f: con trỏ tệp tin đang thao tác.
    - offset: số byte cần dịch chuyển con trỏ tệp 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:

0	**SEEK_SET**	Vị trí đầu tệp tin
1	**SEEK_CUR**	Vị trí hiện tại của con trỏ tệp tin
2	**SEEK_END**	Vị trí cuối tệp tin

    - Kết quả trả về của hàm là 0 nếu việc di chuyển thành công. Nếu không thành công, 1 giá trị khác 0 (đó là 1 mã lỗi) được trả về.

Ví dụ: Viết chương trình ghi lên tệp tin CacSo.Dat 3 giá trị số (thực, nguyên, nguyên dài). Sau đó đọc các số từ tệp tin vừa ghi và hiển thị lên màn hình.

```
#include<stdio.h>
#include<conio.h>
int main()
{
FILE *f;
clrscr();
f=fopen("D:\\CacSo.txt","wb");
if (f!=NULL)
{
double d=3.14;
int i=101;
long l=54321;
fwrite(&d,sizeof(double),1,f);
fwrite(&i,sizeof(int),1,f);
fwrite(&l,sizeof(long),1,f);
/* Doc tu tap tin*/
rewind(f);
fread(&d,sizeof(double),1,f);
fread(&i,sizeof(int),1,f);
fread(&l,sizeof(long),1,f);
printf("Cac ket qua la: %f %d %ld",d,i,l);
fclose(f);
}
getch();
return 0;
}
```

Tóm tắt nội dung

- Khái niệm về cấu trúc dữ liệu
- Định nghĩa cấu trúc
- Sử dụng cấu trúc
- Khái niệm tệp tin
- Khai báo, phân loại tệp
- Truy cập tệp tin

Thảo luận

- Sử dụng mảng cấu trúc
- Tìm hiểu cách truyền tham số kiểu cấu trúc
- Các file văn bản và các file nhị phân
- Con trỏ đến vị trí hiện hành của file
- Vấn đề cần đóng file sau khi làm việc xong

CÂU HỎI VÀ BÀI TẬP

Bài 1: Viết chương trình quản lý một tệp tin văn bản theo các yêu cầu:

A. Nhập từ bàn phím nội dung một văn bản sau đó ghi vào đĩa.

B. Đọc từ đĩa nội dung văn bản vừa nhập và in lên màn hình.

C. Đọc từ đĩa nội dung văn bản vừa nhập, in nội dung đó lên màn hình và cho phép nối thêm thông tin vào cuối tệp tin đó.

Bài 2: Viết chương trình cho phép thống kê số lần xuất hiện của các ký tự là chữ (‘A’..’Z’,’a’..’z’) trong một tệp tin văn bản.

Bài 3: Viết chương trình đếm số từ và số dòng trong một tệp tin văn bản.

Bài 4: Mỗi sinh viên cần quản lý ít nhất 2 thông tin: mã sinh viên và họ tên. Viết chương trình cho phép lựa chọn các chức năng: nhập danh sách sinh viên từ bàn phím rồi ghi lên tệp tin SinhVien.dat, đọc dữ liệu từ tệp tin SinhVien.dat rồi hiển thị danh sách lên màn hình, tìm kiếm họ tên của một sinh viên nào đó dựa vào mã sinh viên nhập từ bàn phím

HỎI VÀ ĐÁP