Thực tập KỸ THUẬT LẬP TRÌNH

Tuần 4-6: Xây dựng cấu trúc dữ liệu và các chức năng nhập/xuất dữ liệu

Yêu cầu:

- Xây dựng cấu trúc dữ liệu phù hợp để quản lý đối tượng sinh viên, gồm các thông tin:
    - Mã lớp
    - Mã sinh viên
    - Họ và tên
    - Ngày sinh
    - Điểm trung bình tích lũy
- Dữ liệu (hồ sơ sinh viên) được ghi trên file.
- Xây dựng các chức năng cho phép nhập hồ sơ, in danh sách đã nhập.
- Tự động chỉnh sửa chính tả khi nhập họ tên sinh viên.
- Kiểm tra tính hợp lệ của ngày sinh khi nhập. Ngày sinh có dạng dd/mm/yyyy, dd là ngày có giá trị trong khoảng từ 1 đến 31, mm là tháng có giá trị trọng khoảng 1 đến 12 và yyyy là năm có giá trị từ 1900 đến 2016; và dd/mm/yyyy phải là ngày hợp hệ (có trên lịch), ví dụ ngày 30/2/2016 là không hợp lệ.
- Kiểm tra tính hợp lệ của điểm trung bình tích lũy, điểm trung bình tích lũy >=0 và <=10.
- Các chức năng này được kết hợp trong chương trình đã xây dựng ở các tuần trước.

Kiến thức liên quan:

A. KIỂU BẢN GHI

1 Khái niệm

Vấn đề: Làm thế nào để mô tả một SINH VIÊN với các thông tin:

- Mã số sinh viên;
- Họ tên;
- Ngày tháng năm sinh;
- Giới tính;

- Địa chỉ thường trú

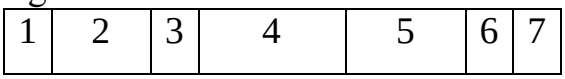
Hoặc làm thế nào để mô tả NGÀY THÁNG bao gồm các thông tin:

- Ngày;
- Tháng;
- Năm

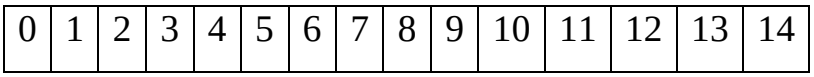
=> Hầu hết các ngôn ngữ lập trình trong đó có C/C++ cho phép người lập trình tự định nghĩa ra cấu trúc mới theo nhu cầu sử dụng từ những kiểu dữ liệu đã có hoặc đã định nghĩa trước đó.

Kiểu cấu trúc (Structure) là kiểu dữ liệu bao gồm nhiều thành phần có kiểu khác nhau, mỗi thành phần được gọi là một trường (field).

Sự khác biệt giữa kiểu cấu trúc và kiểu mảng là: các phần tử của mảng là cùng kiểu còn các phần tử của kiểu cấu trúc có thể có kiểu khác nhau. Hình ảnh sau là của kiểu cấu trúc với 7 trường



còn kiểu mảng có dạng:



Cú pháp 1:

```
struct <Tên cấu trúc>
{
    <Kiểu> <Tên trường 1> ;
    <Kiểu> <Tên trường 2> ;
    ……..
    <Kiểu> <Tên trường n> ;
};
```

Cú pháp 2:

```
typedef struct
{
    <Kiểu> <Tên trường 1> ;
    <Kiểu> <Tên trường 2> ;
    ……..
    <Kiểu> <Tên trường n> ;
} <Tên cấu trúc>;
```

Trong đó:

- <Tên cấu trúc>: là một tên được đặt theo quy tắc đặt tên của danh biểu; tên này mang ý nghĩa sẽ là tên kiểu cấu trúc;

- <Kiểu> <Trường i> (i=1..n): mỗi trường trong cấu trúc có dữ liệu thuộc kiểu gì (tên của trường phải là một tên được đặt theo quy tắc đặt tên của danh biểu).

Ví dụ 1: Để quản lý ngày, tháng, năm của một ngày trong năm ta có thể khai báo kiểu cấu trúc gồm 3 thông tin: ngày, tháng, năm.

```
struct KieuNgayThang
{
    unsigned char Ngay;
    unsigned char Thang;
    unsigned int Nam;
};
typedef struct
{
    unsigned char Ngay;
    unsigned char Thang;
    unsigned int Nam;
} KieuNgayThang;
```

Ví dụ 2: Mỗi sinh viên cần được quản lý bởi các thông tin: Mã số sinh viên, họ tên, ngày tháng năm sinh, giới tính, địa chỉ thường trú. Lúc này ta có thể khai báo một struct gồm các thông tin trên.

```
struct KieuSinhVien
{
char MSSV[10];
char HoTen[40];
struct KieuNgayThang NgaySinh;
int Phai;
char DiaChi[40];
};

typedef struct
{
char MSSV[10];
char HoTen[40];
KieuNgayThang NgaySinh;
int Phai;
char DiaChi[40];
```

```
} KieuSinhVien;
```

- Mỗi thành phần giống là một biến riêng thuộc cấu trúc, nó gồm kiểu và tên thành phần. Một thành phần cũng còn được gọi là trường.
- Phần tên của kiểu cấu trúc và phần danh sách biến có thể có hoặc không. Tuy nhiên trong khai báo kí tự kết thúc cuối cùng phải là dấu chấm phẩy (;).
- Các kiểu cấu trúc được phép khai báo lồng nhau, nghĩa là một thành phần của kiểu cấu trúc có thể lại là một trường có kiểu cấu trúc.
- Một biến có kiểu cấu trúc sẽ được cấp phát bộ nhớ sao cho các thực hiện của nó được sắp liên tục theo thứ tự xuất hiện trong khai báo.

2. Khai báo biến cấu trúc

Việc khai báo biến cấu trúc cũng tương tự như khai báo biến thuộc kiểu dữ liệu chuẩn.

Cú pháp:

- Đối với cấu trúc được định nghĩa theo cách 1:

  struct <Tên cấu trúc> <Biến 1> [, <Biến 2>…];

- Đối với các cấu trúc được định nghĩa theo cách 2:

  <Tên cấu trúc> <Biến 1> [, <Biến 2>…];

Ví dụ: Khai báo biến NgaySinh có kiểu cấu trúc KieuNgayThang; biến SV có kiểu cấu trúc KieuSinhVien.

```
struct KieuNgayThang NgaySinh;
struct KieuSinhVien SV;
KieuNgayThang NgaySinh;
KieuSinhVien SV;
```

3. Các thao tác trên biến kiểu cấu trúc

Truy xuất đến từng trường của biến cấu trúc

Cú pháp:

<Biến cấu trúc>.<Tên trường>;

Khi sử dụng cách truy xuất theo kiểu này, các thao tác trên <Biến cấu trúc>.<Tên trường> giống như các thao tác trên các biến của kiểu dữ liệu của <Tên trường>.

Ví dụ1: Viết chương trình cho phép đọc dữ liệu từ bàn phím cho biến mẩu tin SinhVien và in biến mẩu tin đó lên màn hình:

```
#include<conio.h>
```

```
#include<stdio.h>
#include<string.h>
typedef struct
{
   unsigned char Ngay;
   unsigned char Thang;
   unsigned int Nam;
} KieuNgayThang;
typedef struct
{
   char MSSV[10];
   char HoTen[40];
   KieuNgayThang NgaySinh;
   int Phai;
   char DiaChi[40];
} KieuSinhVien;
/* Hàm in lên màn hình 1 m?u tin SinhVien*/

void InSV(KieuSinhVien s)
{
   printf("MSSV: | Ho va ten | Ngay Sinh | Dia chi\n");
   printf("%s | %s | %d-%d-%d | %s\n",s.MSSV,s.HoTen,
s.NgaySinh.Ngay,s.NgaySinh.Thang,s.NgaySinh.Nam,s.DiaChi);
}
int main()
{
   KieuSinhVien SV, s;
   printf("Nhap MSSV: ");gets(SV.MSSV);
   printf("Nhap Ho va ten: ");gets(SV.HoTen);
   printf("Sinh ngay: ");scanf("%d",&SV.NgaySinh.Ngay);
   printf("Thang: ");scanf("%d",&SV.NgaySinh.Thang);
   printf("Nam: ");scanf("%d",&SV.NgaySinh.Nam);
   printf("Gioi tinh (0: Nu), (1: Nam): ");scanf("%d",&SV.Phai);
   fflush(stdin);
   printf("Dia chi: ");gets(SV.DiaChi);
   InSV(SV);
   s=SV;
   InSV(s);
```

```
    getch();
    return 0;
}
```

Lưu ý:

- Các biến cấu trúc có thể gán cho nhau. Thực chất đây là thao tác trên toàn bộ cấu trúc không phải trên một trường riêng rẽ nào. Chương trình trên dòng s=SV là một ví dụ;
- Với các biến kiểu cấu trúc ta không thể thực hiện được các thao tác sau đây:
    - Sử dụng các hàm xuất nhập trên biến cấu trúc;
    - Các phép toán quan hệ, các phép toán số học và logic.

Ví dụ 2: Nhập vào hai số phức và tính tổng của chúng. Ta biết rằng số phức là một cặp (a,b) trong đó a, b là các số thực, a gọi là phần thực, b là phần ảo. (Đôi khi người ta cũng viết số phức dưới dạng a + ib trong đó i là một đơn vị ảo có tính chất i^2=-1). Gọi số phức c1=(a1, b1) và c2=(a2,b2) khi đó tổng của hai số phức c1 và c2 là một số phức c3 mà c3=(a1+a2, b1+b2). Với hiểu biết như vậy ta có thể xem mỗi số phức là một cấu trúc có hai trường, một trường biểu diễn cho phần thực, một trường biểu diễn cho phần ảo. Việc tính tổng của hai số phức được tính bằng cách lấy phần thực cộng với phần thực và phần ảo cộng với phần ảo.

```
#include<conio.h>
#include<stdio.h>
#include<string.h>
typedef struct
{
    float Thuc;
    float Ao;
} SoPhuc;
/* Ham in so phuc len man hinh*/
void InSoPhuc(SoPhuc p)
{
    printf("%.2f + i%.2f\n",p.Thuc,p.Ao);
}
int main()
{
```

```
    SoPhuc p1,p2,p;
    printf("Nhap so phuc thu nhat:\n");
    printf("Phan thuc: ");scanf("%f",&p1.Thuc);
    printf("Phan ao: ");scanf("%f",&p1.Ao);
    printf("Nhap so phuc thu hai:\n");
    printf("Phan thuc: ");scanf("%f",&p2.Thuc);
    printf("Phan ao: ");scanf("%f",&p2.Ao);
    printf("So phuc thu nhat: ");
    InSoPhuc(p1);
    printf("So phuc thu hai: ");
    InSoPhuc(p2);
    p.Thuc = p1.Thuc+p2.Thuc;
    p.Ao = p1.Ao + p2.Ao;
    printf("Tong 2 so phuc: ");
    InSoPhuc(p);
    getch();
    return 0;
}
```

Khởi tạo cấu trúc

Việc khởi tạo cấu trúc có thể được thực hiện trong lúc khai báo biến cấu trúc. Các trường của cấu trúc được khởi tạo được đạt giữa 2 dấu { và }, chúng được phân cách nhau bởi dấu phẩy (,).

Ví dụ: Khởi tạo biến cấu trúc NgaySinh:

```
struct KieuNgayThang NgaySinh ={29, 8, 1986};
```

4. Con trỏ cấu trúc

Khai báo

Việc khai báo một biến con trỏ kiểu cấu trúc cũng tương tự như khi khai báo một biến con trỏ khác, nghĩa là đặt thêm dấu * vào phía trước tên biến.

Cú pháp:

struct <Tên cấu trúc> * <Tên biến con trỏ>;

Ví dụ: Ta có thể khai báo một con trỏ cấu trúc kiểu NgayThang như sau:

```
struct NgayThang *p;
/* NgayThang *p; // Nếu có định nghĩa kiểu */
```

Sử dụng các con trỏ kiểu cấu trúc

Khi khai báo biến con trỏ cấu trúc, biến con trỏ chưa có địa chỉ cụ thể. Lúc này nó chỉ mới được cấp phát 2 byte để lưu giữ địa chỉ và được ghi nhận là con trỏ chỉ đến 1 cấu trúc, nhưng chưa chỉ đến 1 đối tượng cụ thể. Muốn thao tác trên con trỏ cấu trúc hợp lệ, cũng tương tự như các con trỏ khác, ta phải:

- Cấp phát một vùng nhớ cho nó (sử dụng hàm malloc() hay calloc);
- Hoặc, cho nó quản lý địa chỉ của một biến cấu trúc nào đó.

Ví dụ: Sau khi khởi tạo giá trị của cấu trúc:

```
struct NgayThang Ngay = {29,8,1986};
p = &Ngay;
```

lúc này biến con trỏ p đã chứa địa chỉ của Ngay.

Truy cập các thành phần của cấu trúc đang được quản lý bởi con trỏ

Để truy cập đến từng trường của 1 cấu trúc thông qua con trỏ của nó, ta sử dụng toán tử dấu mũi tên (->: dấu - và dấu >). Ngoài ra, ta vẫn có thể sử dụng đến phép toán * để truy cập vùng dữ liệu đang được quản lý bởi con trỏ cấu trúc để lấy thông tin cần thiết.

Ví dụ: Sử dụng con trỏ cấu trúc.

```
#include<conio.h>
#include<stdio.h>
typedef struct
{
   unsigned char Ngay;
   unsigned char Thang;
   unsigned int Nam;
} NgayThang;
int main()
{
   NgayThang Ng={29,8,1986};
   NgayThang *p;
   p=&Ng;
   printf("Truy cap binh thuong %d-%d-%d\n",
   Ng.Ngay,Ng.Thang,Ng.Nam);
   printf("Truy cap qua con tro %d-%d-%d\n",
   p->Ngay,p->Thang,p->Nam);
   printf("Truy cap qua vung nho con tro %d-%d-%d\n",
```

```
    (*p).Ngay,(*p).Thang,(*p).Nam);
    getch();
    return 0;
}
```

- Cho phép sử dụng cấu trúc, con trỏ cấu trúc là đối số của hàm như các loại biến khác
- Cho phép hàm xây dựng trả về là kiểu cấu trúc

Ví dụ : Xử lý danh sách sinh viên, sử dụng hàm với đối số là cấu trúc

```
#include <stdio.h>
#include <conio.h>
//Ngay thang
struct Ngaythang {
int ng ;
int th ;
int nam ;
};
//Sinh vien
struct Sinhvien {
char hoten[25] ;
Ngaythang ns;
int gt;
float diem ;
} ;
//cac ham xu ly
int sua(Sinhvien &r) ;
int in(Sinhvien x) ;
int nhap(Sinhvien *p) ;
int nhapds(Sinhvien *a, int n) ;
int suads(Sinhvien *a, int n) ;
int inds(const Sinhvien *a, int n) ;
//
struct Sinhvien a[10];

int main()
{
    nhap(a);
//  in(a[1]);
//  sua(a[1]);
```

```
    nhapds(a,9);
    suads(a,9);
    inds(a,9);
    getch();
    return 0;
}
///trien khai cac ham
int sua(Sinhvien &r)
{
 int chon;
 do {
    printf("1: Sua ho ten\n2: Sua ngay sinh\n3:Sua gioi tinh\n4:Sua diem\n5:In\n0: Ket thuc\n");
    scanf("%d",&chon);
    switch (chon)
    {
        case 1:
            printf("Nhap ho ten:");
            scanf("%s",r.hoten);
            break;
        case 2:
            printf("Nhap ngay thang nam sinh:");
            scanf("%d%d%d",&r.ns.ng,&r.ns.th,&r.ns.nam);
            break;
        case 3:
            printf("Nhap gioi tinh 0:Nu 1:Nam:");
            scanf("%d",&r.gt) ;
            break;
        case 4:
            printf("Nhap diem:");
            scanf("%f",&r.diem);
            break;
        case 5:
            printf("Sinh vien:");
            in(r);
            break;
        case 0:
            break;
```

```
            default:
                        printf("Nhap gia tri khong dung\n");
                        break;
        }
     } while (chon) ;
     return 0;
    }
    ////////
    int in(Sinhvien x)
    {
    printf("Ho ten :%s\nNgay sinh %d/%d/%d\n",x.hoten,x.ns.ng, x.ns.th,
x.ns.nam) ;
    printf("Gioi tinh :%s\nDiem :%f\n",(x.gt==1) ?"Nam" :"Nu",x.diem) ;
    return 0;
    }

    ///////
    int nhap(Sinhvien *p)
    {
            printf("Nhap ho va ten: ");
            scanf("%s",p->hoten);
            printf("Nhap ngay sinh (ngay thang nam): ");
            scanf("%d%d%d", &p->ns.ng ,&p->ns.th,&p->ns.nam);
            printf("Gioi tinh 0: nu, 1: nam: ");
            scanf("%d",& p->gt);
            printf("Diem: ");
            scanf("%f",& p->diem);
            return 0;
    }
    //////
    int nhapds(Sinhvien *a, int n)
    {

     for (int i=1; i<=n; i++) nhap(&a[i]) ;
     return 0;
    }
    /////
    int suads(Sinhvien *a, int n)
```

```
{
    int chon;
    do
    {
       printf("\nNhap phan tu duoc su tu 1 den %d, gia tri khac thoat:",n);
       scanf("%d",&chon);
       if(chon>0 &&chon<=n)
       {
           sua(a[chon]) ;
       }
    }while(chon>0 && chon<=n);
    return 0;
}
//////////
int inds(const Sinhvien *a, int n)
{
   for (int i=1; i<=n; i++)
      in(a[i]) ;
   return 0;
}
```

5. Cấu trúc với thành phần kiểu bit

a. Trường bit

Để tiết kiệm trong lưu trữ, trong ngôn ngữ lập trình C cho phép khai báo các trường của cấu trúc với số lượng bit xác định không phụ thuộc vào số lượng bit các kiểu dữ liệu chuẩn.

Một trường bit là một khai báo trường int và thêm dấu: cùng số bit n theo sau, trong đó $0 \leq n < 15$. Ví dụ do độ lớn của ngày không vượt quá 31, tháng không vượt quá 12 nên 2 trường này trong cấu trúc ngày tháng có thể khai báo tiết kiệm hơn bằng 5 và 4 bit như sau:

```
struct Date {
int ng: 5;
int th: 4;
int nam:14;
} ;
```

b. Đặc điểm

Cần chú ý các đặc điểm sau của một cấu trúc có chứa trường bit:

− Các bit được bố trí liên tục trên dãy các byte.

− Kiểu trường bit phải là int (signed hoặc unsigned).

− Độ dài mỗi trường bit không quá 16 bit.

− Có thể bỏ qua một số bit nếu bỏ trống tên trường, ví dụ:

```
struct tu {
    int: 8;
    int x:8;
}
```

mỗi một biến cấu trúc theo khai báo trên gồm 2 byte, bỏ qua không sử dụng byte thấp và trường x chiếm byte (8 bit) cao.

− Không cho phép lấy địa chỉ của thành phần kiểu bit.

− Không thể xây dựng được mảng kiểu bit.

− Không được trả về từ hàm một thành phần kiểu bit. Ví dụ nếu b là một thành phần của biến cấu trúc x có kiểu bit thì câu lệnh sau là sai:

```
return x.b ;    // sai
```

tuy nhiên có thể thông qua biến phụ như sau:

```
int tam = x.b ; return tam ;
```

− Tiết kiệm bộ nhớ

− Dùng trong kiểu union để lấy các bit của một từ (xem ví dụ trong phần kiểu hợp).

6. Câu lệnh typedef

Để thuận tiện trong sử dụng, thông thường các kiểu được NSD tạo mới sẽ được gán cho một tên kiểu bằng câu lệnh typedef như sau:

```
typedef <kiểu> <tên_kiểu> ;
```

Ví dụ: Để tạo kiểu mới có tên Bool và chỉ chứa giá trị nguyên (thực chất chỉ cần 2 giá trị 0, 1), ta có thể khai báo:

```
typedef int Bool;
```

khai báo này cho phép xem Bool như kiểu số nguyên.

hoặc có thể đặt tên cho kiểu ngày tháng là Date với khai báo sau:

```
typedef struct Date {
 int ng;
 int th;
 int nam;
};
```

khi đó ta có thể sử dụng các tên kiểu này trong các khai báo (ví dụ tên kiểu của đối, của giá trị hàm trả lại …).

7. Hàm sizeof()

Hàm trả lại kích thước của một biến hoặc kiểu. Ví dụ:

```
Bool a, b;
Date x, y, z[50];
printf("%d,%d,%d",sizeof(a),sizeof(b),sizeof(Bool)) ;     // in 2,2,2
printf("Số phần tử của z =%d ",sizeof(z) / sizeof(Date));  // in 50
```

B. LÀM VIỆC VỚI FILE

I. Một số khái niệm

Dữ liệu trong chương trình được lưu trữ ở RAM máy tính, vì thế khi kết thúc chương trình, tắt máy dữ liệu sẽ bị giải phóng. Để xử lý dữ liệu cần phải lưu trữ trên bộ nhớ ngoài (đĩa cứng, USB, …) dưới dạng file. File là tập hợp các byte liên tục được lưu trữ và được gán tên gọi. Khi xử lý file chương trình có thể xem xét chuỗi byte với cách nhìn khác nhau, có những ứng xử khác nhau với dữ liệu. Trong C hỗ trợ việc thao tác với file với quan điểm: file văn bản, file nhị phân.

- File văn bản (Text File): là loại tập tin dùng để ghi các ký tự lên đĩa. Điểm đặc biệt là dữ liệu của tập 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). Mỗi tập tin được kết thúc bởi ký tự EOF (End Of File) có mã Ascii là 26 (xác định bởi tổ hợp phím Ctrl + Z).

Truy xuất tập tin theo kiểu văn bản chỉ có thể truy xuất theo kiểu tuần tự.

- Tập tin nhị phân: Quan điểm tập tin là dãy byte liên tục, việc xử lý với tập tin dựa trên việc đọc ghi dãy byte.

Biến tập tin: là một biến thuộc kiểu dữ liệu tập tin dùng để đại diện cho một tập tin. Dữ liệu chứa trong một tập tin được truy xuất qua các thao tác với thông số là biến tập tin đại diện cho tập tin đó.

Con trỏ tập tin: Khi một tập tin được mở ra để làm việc, tại mỗi thời điểm, sẽ có một vị trí của tập 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 tập tin.

Sau khi đọc/ghi xong dữ liệu, con trỏ sẽ chuyển dịch thêm một phần tử về phía cuối tập tin. Sau phần tử dữ liệu cuối cùng của tập tin là dấu kết thúc tập tin EOF (End Of File).

II. Các thao tác trên tập tin

Các thao tác với tập tin:

- Khai báo biến tập tin.
- Mở tập tin bằng hàm fopen().
- Thực hiện các thao tác xử lý dữ liệu của tập tin bằng các hàm đọc/ghi dữ liệu.
- Đóng tập tin bằng hàm fclose().

Ở đây, ta thao tác với tập tin nhờ các hàm được định nghĩa trong thư viện stdio.h.

1. Khai báo biến tập tin

Cú pháp

FILE <Danh sách các biến con trỏ, đại diện cho tập tin>

Các biến trong danh sách phải là các con trỏ và được phân cách bởi dấu phẩy(,).

Ví dụ: FILE *f1,*f2;

2. Mở tập tin

Cú pháp

*FILE *fopen(char *Path, const char *Mode)*

Trong đó:

- Path: chuỗi chỉ đường dẫn đến tập tin trên đĩa.

- Mode: chuỗi xác định cách thức mà tập tin sẽ mở. Các giá trị có thể của Mode:

Chế độ	Ý nghĩa
r	Mở tập tin văn bản để đọc
w	Tạo ra tập tin văn bản mới để ghi
a	Nối vào tập tin văn bản

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

Mặc định là mở dạng text nếu không có xác định là b, nếu rõ ràng hơn thì thêm chỉ định t để xác định là kiểu text.

- Hàm fopen trả về một con trỏ tập tin. Chương trình của ta không thể thay đổi giá trị của con trỏ này. Nếu có một lỗi xuất hiện trong khi mở tập tin thì hàm này trả về con trỏ NULL.

Ví dụ: Mở một tập tin tên TEST.txt để ghi.

```
FILE *f;
f = fopen("c:\\TEST.txt", "w");
if (f!=NULL)
{
/* Các câu lệnh để thao tác với tập tin*/
/* Đóng tập tin*/
}
```

Kiểm tra con trỏ f với giá trị NULL cho phép xác định được lệnh thực hiện thành công hay không?

Nếu mở tập tin để ghi, trường hợp tập tin đã tồn tại rồi thì tập tin sẽ bị xóa và một tập tin mới được tạo ra. Nếu ta muốn ghi nối dữ liệu, ta phải sử dụng chế độ "a". Khi mở với chế độ đọc, tập tin phải tồn tại rồi, nếu không một lỗi sẽ xuất hiện.

3. Đóng tập tin

Hàm fclose() được dùng để đóng tập tin được mở bởi hàm fopen(). Hàm này sẽ ghi dữ liệu còn lại trong vùng đệm vào tập tin và đóng lại tập tin.

Cú pháp: *int fclose(FILE *f)*

Trong đó f là con trỏ tập tin được mở bởi hàm fopen(). Giá trị trả về của hàm là 0 báo rằng việc đóng tập tin thành công. Hàm trả về EOF nếu có xuất hiện lỗi.

Ngoài ra, ta còn có thể sử dụng hàm fcloseall() để đóng tất cả các tập tin lại.

Cú pháp: *int fcloseall()*

Kết quả trả về của hàm là tổng số các tập tin được đóng lại. Nếu không thành công, kết quả trả về là EOF.

4. Kiểm tra đến cuối tập tin hay chưa?

Cú pháp: *int feof(FILE *f)*

Ý nghĩa: Kiểm tra xem đã chạm tới cuối tập tin hay chưa và trả về EOF nếu cuối tập tin được chạm tới, ngược lại trả về 0.

5. Di chuyển con trỏ tập tin về đầu tập tin - Hàm rewind()

- Chuyển về đầu tập tin, sử dụng hàm rewind().

Cú pháp: *void rewind(FILE *f);*

+ f: con trỏ tập tin đang thao tác.

- Chuyển đến vị trí bất kỳ sử dụng hàm fseek().

Cú pháp: *int fseek(FILE *f, long offset, int whence);*

+ f: con trỏ tập tin đang thao tác.

+ offset: số byte cần dịch chuyển con trỏ tập tin.

+ whence: vị trí bắt đầu để tính khoảng cách dịch chuyển cho offset:

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

- Lấy vị trị của con trỏ file hàm ftell();

Cú pháp: *long ftell(FILE *stream);*

+ stream: biến đại diện cho file

+ trả về vị trí của con trỏ file so với đầu file

III. Truy cập tập tin văn bản

1. Ghi dữ liệu lên tập tin văn bản

1.1 Hàm fputc()

Hàm này được dùng để ghi một ký tự lên một tập tin văn bản đang được mở để làm việc.

Cú pháp: *int fputc(int c, FILE *f)*

Trong đó, tham số c chứa mã Ascii của một ký tự nào đó. Mã này được ghi lên tập tin liên kết với con trỏ f. Hàm này trả về EOF nếu gặp lỗi.

1.2 Hàm fputs()

Hàm này dùng để ghi một chuỗi ký tự chứa trong vùng đệm lên tập tin văn bản.

Cú pháp: *int fputs(const char *buffer, FILE *f)*

Trong đó, 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.

1.3 Hàm fprintf()

Hàm này dùng để ghi dữ liệu có định dạng lên tập tin văn bản.

Cú pháp: *fprintf(FILE *f, const char *format, varexpr)*

Trong đó: 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	Ghi số thực dạng khoa học (nhân 10 mũ x)

hoặc %G	

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

```
#include<stdio.h>
#include<conio.h>
int main(){
FILE *f;
clrscr();
f=fopen("D:\\Baihat.txt","w+");
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;
}
```

2. Đọc dữ liệu từ tập tin văn bản

2.1 Hàm fgetc()

Hàm này dùng để đọc dữ liệu từ tập tin văn bản đang được mở để làm việc.

Cú pháp: *int fgetc(FILE *f)*

Hàm này trả về mã Ascii của một ký tự nào đó (kể cả EOF) trong tập tin liên kết với con trỏ f.

2.2 Hàm fgets()

Cú pháp: *char *fgets(char *buffer, int n, FILE *f)*

Hàm này được dùng để đọc một chuỗi ký tự từ tập 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’ (ký tự này cũng được đưa vào chuỗi kết quả) 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 tập tin nào đó.

- Ký tự NULL (‘\0’) tự động được thêm vào cuối chuỗi kết quả lưu trong vùng đêm.

- 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.

2.3 Hàm fscanf()

Hàm này dùng để đọc dữ liệu từ tập 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 tập tin D:\Baihat.txt ở trên sang tập 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;
```

```
}
```

3. Ví dụ

Đọc ma trận tính ma trận, ghi ma trận chuyển vị vào file mới. Xem xét file văn bản như đầu vào, đầu ra của chương trình, thay vì nhận từ bàn phím và xuất ra màn hình.

```
#include<stdio.h>
#include<conio.h>
int a[100][100];
int n,m;

int main()
{
    char *infile="vao.txt", *outfile="ra.txt";
    int i, j;
    FILE *f1,*f2;
    f1=fopen(infile,"rt");
    if(f1==NULL)
    {
            printf("Loi mo file %s",infile);
            getch();
            return -1;
    }
    f2=fopen(outfile,"wt");
    if(f2==NULL)
    {
            printf("Loi mo file %s",outfile);
            getch();
            fclose(f1);
            return -2;
    }
    fscanf(f1,"%d%d",&m,&n);
    for(i=0;i<m;i++)
    {
            for(j=0;j<n;j++)
            {
                    fscanf(f1,"%d",&a[i][j]);
            }
    }
```

```
        fprintf(f2,"%d %d\n",n,m);
        for(i=0;i<n;i++)
        {
                for(j=0;j<m;j++)
                {
                        fprintf(f2,"%4d",a[j][i]);
                }
                fprintf(f2,"\n");
        }
        fclose(f1);
        fclose(f2);
        printf("Ket thuc tinh chuyen vi");
        getch();
        return 0;
    }
```

IV. Truy cập tập tin nhị phân

1. Ghi dữ liệu lên tập tin nhị phân

Cú pháp: *size_t fwrite(const void *ptr, size_t size, size_t n, FILE *f)*

Trong đó:

- ptr: con trỏ chỉ đến vùng nhớ chứa thông tin cần ghi lên tập tin.
- n: số phần tử sẽ ghi lên tập tin.
- size: kích thước của mỗi phần tử.
- f: con trỏ tập tin đã được mở.
- Giá trị trả về của hàm này là số phần tử được ghi lên tập tin. Giá trị này bằng n trừ khi xuất hiện lỗi.

2. Đọc dữ liệu từ tập tin nhị phân

Cú pháp: *size_t fread(const void *ptr, size_t size, size_t n, FILE *f)*

Trong đó:

- ptr: con trỏ chỉ đến vùng nhớ sẽ nhận dữ liệu từ tập tin.
- n: số phần tử được đọc từ tập tin.
- size: kích thước của mỗi phần tử.
- f: con trỏ tập tin đã được mở.

- Giá trị trả về của hàm này là số phần tử đã đọc được từ tập tin. Giá trị này bằng n hay nhỏ hơn n nếu đã chạm đến cuối tập tin hoặc có lỗi xuất hiện..

3. Ví dụ

Ví dụ 1: Viết chương trình ghi lên tập tin CacSo.Dat 3 giá trị số (thực, nguyên, nguyên dài). Sau đó đọc các số từ tập tin vừa ghi và hiển thị lên màn hình.

```
#include<stdio.h>
#include<conio.h>
int main()
{
FILE *f;
f=fopen("number.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);
}
else
{
   printf("Khong mo duoc file");
}
getch();
return 0;
}
```

Ví dụ 2: 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 tập tin SinhVien.dat, đọc dữ liệu từ tập 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.

Ta nhận thấy rằng mỗi phần tử của tập tin SinhVien.Dat là một cấu trúc có 2 trường: mã và họ tên. Do đó, ta cần khai báo cấu trúc này và sử dụng các hàm đọc/ghi tập tin nhị phân với kích thước mỗi phần tử của tập tin là chính kích thước cấu trúc đó.

Trong trường hợp này có thể coi file nhị phân như là nơi lưu trữ dữ liệu lâu dài, cũng có thể coi như là nơi lưu trữ xử lý dữ liệu thay vì dùng bộ nhớ.

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
typedef struct
{
   char Ma[10];
   char HoTen[40];
} SinhVien;
///-----------
void WriteFile(char *FileName)
{
   FILE *f;
   int n,i;
   SinhVien sv;
   f=fopen(FileName,"ab");
   printf("Nhap bao nhieu sinh vien? ");
   scanf("%d",&n);
   for(i=1;i<=n;i++)
   {
         printf("Sinh vien thu %i\n",i);
         fflush(stdin);
         printf(" - MSSV: ");gets(sv.Ma);
         printf(" - Ho ten: ");gets(sv.HoTen);
         fwrite(&sv,sizeof(sv),1,f);
   }
   fclose(f);
```

```
    printf("Bam phim bat ky de tiep tuc");
    getch();
}
void ReadFile(char *FileName)
{
    FILE *f;
    SinhVien sv;
    f=fopen(FileName,"rb");
    printf(" MSSV | Ho va ten\n");
    fread(&sv,sizeof(sv),1,f);
    while (!feof(f))
    {
            printf(" %s | %s\n",sv.Ma,sv.HoTen);
            fread(&sv,sizeof(sv),1,f);
    }
    fclose(f);
    printf("Bam phim bat ky de tiep tuc!!!");
    getch();
}
void Search(char *FileName)
{
    char MSSV[10];
    FILE *f;
    int Found=0;
    SinhVien sv;
    fflush(stdin);
    printf("Ma so sinh vien can tim: ");gets(MSSV);
    f=fopen(FileName,"rb");
    while (!feof(f) && Found==0)
    {
            fread(&sv,sizeof(sv),1,f);
            if (strcmp(sv.Ma,MSSV)==0) Found=1;
    }
    fclose(f);
    if (Found == 1)
    {
            printf("Tim thay SV co ma %s. Ho ten la: %s",sv.Ma,sv.HoTen);
    }
```

```
    else
    {
            printf("Tim khong thay sinh vien co ma %s",MSSV);
    }
    printf("\nBam phim bat ky de tiep tuc!!!");
    getch();
}
int main()
{
int c;
for (;;)
{
printf("1. Nhap DSSV\n");
printf("2. In DSSV\n");
printf("3. Tim kiem\n");
printf("4. Thoat\n");
printf("Ban chon 1, 2, 3, 4: ");
scanf("%d",&c);
fflush(stdin);

if(c==1)
{
   WriteFile("SinhVien.dat");
}
else if (c==2)
{
   ReadFile("SinhVien.dat");
}
else if (c==3)
{
   Search("SinhVien.Dat");
}
else break;
}
return 0;
}
```

Ngoài ra thư viện stdio.h còn định nghĩa một số hàm khác cho phép thao tác với tập tin, sinh viên có thể tham khảo trong phần trợ giúp.

Kết quả: Chương trình **QLSV** chạy được theo các yêu cầu đặt ra.