BỘ GIÁO DỤC VÀ ĐÀO TẠO

ĐẠI HỌC CÔNG NGHỆ TP.HCM

THỰC HÀNH KỸ THUẬT LẬP TRÌNH

Biên Soạn:

ThS. Trương Thị Minh Châu

www.hutech.edu.vn

THỰC HÀNH KỸ THUẬT LẬP TRÌNH

Các ý kiến đóng góp về tài liệu học tập này, xin gửi về e-mail của ban biên tập: tailieuhoctap@hutech.edu.vn

MỤC LỤC

HƯỚNG DẪN

MÔ TẢ MÔN HỌC

Môn Thực hành Kỹ thuật Lập trình cung cấp cho sinh viên những kỹ năng thực hành cơ bản về ngôn ngữ lập trình C. Môn học này là nền tảng để tiếp thu hầu hết các môn học khác trong chương trình đào tạo. Mặt khác, nắm vững môn này là cơ sở để phát triển tư duy và kỹ năng lập trình để giải các bài toán và các ứng dụng trong thực tế.

Học xong môn này, sinh viên phải nắm được các vấn đề sau:

- Thao tác trên mảng 1 chiều, 2 chiều.
- Thao tác trên kiểu dữ liệu có cấu trúc
- Thao tác trên kiểu dữ liệu con trỏ với mảng 1 chiều, 2 chiều và mảng cấu trúc.
- Cách viết hàm đệ quy.
- Thao tác trên tập tin.
- Kỹ thuật sắp xếp, tìm kiếm cơ bản.

NỘI DUNG MÔN HỌC

- Bài 1 MẢNG MỘT CHIỀU-HAI CHIỀU.
- Bài 2 KIỂU DỮ LIỆU CÓ CẤU TRÚC
- Bài 3 KIỂU CON TRỎ
- Bài 4 ĐỆ QUY
- Bài 5 TẬP TIN (FILE)- TÌM KIẾM
- Bài 6 SẮP XẾP

YÊU CẦU MÔN HỌC

Có tư duy tốt về logic và kiến thức toán học cơ bản.

CÁCH TIẾP NHẬN NỘI DUNG MÔN HỌC

Thực hành Kỹ thuật lập trình là môn học đầu tiên giúp sinh viên làm quen với phương pháp lập trình trên máy tính, giúp sinh viên có khái niệm cơ bản về cách tiếp cận và giải quyết các bài toán tin học, giúp sinh viên có khả năng tiếp cận với cách tư duy của người lập trình, và là tiền đề để tiếp cận với các học phần quan trọng còn lại của ngành Công nghệ Thông tin. Vì vậy, yêu cầu người học phải dự học đầy đủ các buổi thực hành tại lớp, thực hành lại các bài tập ở nhà, nghiên cứu tài liệu trước khi đến lớp và gạch chân những vấn đề không hiểu khi đọc tài liệu để đến lớp trao đổi.

PHƯƠNG PHÁP ĐÁNH GIÁ MÔN HỌC

Để học tốt môn này, người học cần ôn tập các bài đã học, trả lời các câu hỏi và làm đầy đủ bài tập; đọc trước bài mới và tìm thêm các thông tin liên quan đến bài học.

Đối với mỗi bài học, người học đọc trước phần tóm tắt kiến thức lý thuyết, sau đó thực hành các bài từ cơ bản đến nâng cao. Lưu ý xem kỹ hướng dẫn trong mỗi bài thực hành và làm theo. Kết thúc mỗi bài học, học viên cần làm lại các bài thực hành ở nhà và luyện tập thêm.

Điểm đánh giá : gồm 2 phần

1. Điểm quá trình: (chuyên cần) + (điểm bài thực hành hàng tuần).
2. Điểm kiểm tra: trung bình cộng điểm các bài kiểm tra thực hành trên máy (ít nhất 2 lần kiểm tra).

BÀI 1: MẢNG 1 CHIỀU - 2 CHIỀU

Sau khi thực hành xong bài này, sinh viên có thể nắm được:

- *Cách khai báo dữ liệu kiểu mảng 1 chiều, 2 chiều*
- *Các thao tác nhập xuất, các kỹ thuật thao tác trên mảng 1 chiều, 2 chiều, chuỗi ký tự.*

1.1 TÓM TẮT LÝ THUYẾT MẢNG 1 CHIỀU

Mảng một chiều thực chất là một biến được cấp phát bộ nhớ liên tục và bao gồm nhiều biến thành phần.

Các thành phần của mảng là tập hợp các biến có cùng kiểu dữ liệu và cùng tên. Do đó để truy xuất các biến thành phần, ta dùng cơ chế chỉ mục.

Khai báo:

```
Kiểu_dữ_ liệu  Tên_mảng[Kích thước];
```

Ví dụ:

int a[100]; // Khai báo mảng số nguyên a gồm 100 phần tử

float b[50]; // Khai báo mảng số thực b gồm 50 phần tử

Sử dụng:

```
Tên_biến_mảng[chỉ số];
```

Ví dụ : int A[5] // Khai báo mảng A gồm tối đa 5 phần tử nguyên.

Chỉ số (số thứ tự)	0	1	2	3	4
	A[0]	A[1]	A[2]	A[3]	A[4]

1.2 MỘT SỐ KỸ THUẬT NÂNG CAO TRÊN MẢNG 1 CHIỀU

1.2.1 Tìm kiếm trên mảng một chiều

Ví dụ: Viết hàm tìm phần tử có giá trị x xuất hiện đầu tiên trong mảng một chiều.

(Nếu tìm thấy trả về vị trí xuất hiện x, ngược lại trả về -1)

```
int Tim(int a[], int n, int x)
{
	for (int i = 0; i < n ; i ++)
		if ( x==a[i] )    //nếu a[i] thoả điều kiện tìm: if (a[i] thoả đk)
			return i;
	return -1;
}
```

1.2.2 Xoá phần tử tại vị trí cho trước

Duyệt mảng từ trái sang phải. Xuất phát từ vị trí cần xoá tiến hành dời lần lượt các phần tử về phía trước cho đến khi kết thúc mảng, sau đó giảm kích thước mảng.

Vấn đề đặt ra là tìm vị trí cần xóa theo điều kiện bài toán rồi thực hiện xóa. Để tìm vị trí cần xoá, áp dụng hàm tìm kiếm ở trên.

```
//vitri là vị trí phần tử cần xoá
void XoaTaiViTri (int a[], int &n, int vitri)
{
	for (int i = vitri; i < n-1 ; i++)
		a[i] = a[i+1];
	n--;
}
```

1.2.3 Thêm một phần tử tại vị trí cho trước

Duyệt mảng từ phải sang trái. Xuất phát từ cuối mảng tiến hành đẩy lần lượt các phần tử về phía sau cho đến vị trí cần thêm, thêm phần tử mới vào vị trí cần thêm và tăng kích thước mảng. Trước khi thêm ta phải xác định vị trí cần thêm theo điều kiện bài toán.

```
//X là phần tử cần thêm, vitri là vị trí sẽ thêm
void ChenX (int a[], int &n, int X, int vitri)
{
	for (int i = n; i >vitri ; i--)
		a[i] = a[i-1] ;
	a[vitri] = X;
	n++;
}
```

1.2.4 Tách mảng

Kĩ thuật tách cơ bản: Cho mảng a kích thước n (n chẵn). Tách mảng a thành 2 mảng b và c sao cho: b có ½ phần tử đầu của mảng a, ½ phần tử còn lại đưa vào mảng c.

```
//mảng b có m phần tử, mảng c có l phần tử
void TachMang(int a[], int n, int b[], int &m, int c[], int &l)
{
	int k=n/2;
	m=l=0;
	for(int i=0; i<k; i++)
	{
		b[m++]=a[i];
		c[l++]=a[k+i];
	}
}
```

1.2.5 Kĩ thuật đặt cờ hiệu

Kĩ thuật này thường được áp dụng cho những bài toán "kiểm tra" hay "đánh dấu".

Ví dụ: Viết hàm kiểm tra xem mảng các số nguyên có thứ tự tăng dần không?

(Trả về 1: Nếu mảng tăng dần, ngược lại trả về 0).

```
int KiemTraTang (int a[ ], int n)
{
	int flag = 1;  //Giả sử mảng thoả điều kiện tăng dần
```

```
        for (int i = 0; i < n-1; i ++ )
            if ( a[i] > a[i+1] ) // Vi phạm điều kiện tăng dần
            {
                flag = 0;
                break;
            }
    return flag;
    }
```

Sử dụng:

```
if (KiemTraTang(a,n)==1) //hoặc viết gọn if (KiemTraTang(a,n))
    printf("Mang tang dan");
else printf("Mang khong tang dan");
```

1.3 THỰC HÀNH MẢNG 1 CHIỀU

Câu 1: Viết chương trình gồm các hàm thực hiện:

a. Nhập mảng số nguyên gồm n phần tử (0 < n <= 100).

b. Xuất mảng vừa nhập.

c. Tìm vị trí phần tử dương đầu tiên. Xuất ra vị trí và giá trị phần tử dương đầu tiên tìm được nếu có.

d. Tìm vị trí phần tử dương cuối cùng. Xuất ra vị trí và giá trị phần tử dương cuối cùng tìm được nếu có.

e. Tìm giá trị phần tử lớn nhất.

f. Đếm số phần tử lớn nhất.

g. Xuất ra vị trí của các phần tử lớn nhất.

h. Thêm 1 phần tử mới vào đầu mảng.

i. Thêm 1 phần tử mới vào vị trí k trong mảng (k do người dùng nhập, 0<k<=MAX, với MAX là kích thước cấp phát mảng).

j. Xóa phần tử đầu mảng

k. Xóa phần tử tại vị trí k (k do người dùng nhập, 0<k<MAX, với MAX là kích thước cấp phát mảng).

l. Kiểm tra mảng có chứa số lẻ không?

m. Tạo mảng mới chứa các phần tử chẵn của mảng đã nhập.

Hướng dẫn:

- Chương trình minh hoạ nhập xuất mảng số nguyên

```
#include <stdio.h>
#define MAX 100
//-- Hàm nhập sô lượng phần tử cho mảng
void NhapSL (int &n) {
    do{
        printf ("Nhap so phan tu 0<sl<100: ");
        scanf ("%d", &n);
    }while (n<=0 || n>100);
}
//--Hàm nhập giá trị cho từng phần tử trong mảng
void NhapMang (int a[], int n) {
    for (int i = 0; i < n; i ++) {
        printf ("a[%d] =  ", i);
        scanf ("%d", &a[i]);
    }
}
//-- Hàm xuất các phần tử của mảng ra màn hình
void XuatMang (int a[], int n)
{
    printf ("\nMang gom cac phan tu:\n ");
    for (int i = 0; i < n; i ++)
        printf ("%5d", a[i]);
```

```
}
//-- Câu c. Hàm tìm vị trí phần tử dương đầu tiên
int TimDuongDau(int a[], int n){…}
//================================
int main() {
    int a[MAX], n;
      NhapSL(n);
    NhapMang (a,n);
    XuatMang (a,n);
    int vitriD=TimDuongDau(a, n);
    if (vitriD==-1)
        printf("Mang khong co phan tu duong\n");
    else
        printf("Phan tu duong dau tien la %d, nam tai vi tri %d\n", a[vitriD],
vitriD);
    return 0;
}
```

Câu 2: Viết chương trình gồm các hàm sau:

a. Nhập vào mảng A gồm n phần tử, trong quá trình nhập kiểm tra các phần tử nhập vào không được trùng, nếu trùng thông báo và yêu cầu nhập lại.

b. Xuất mảng.

c. Xuất ra màn hình các phần tử là số chính phương nằm tại những vị trí lẻ trong mảng.

d. Xuất ra vị trí của các phần tử có giá trị lớn nhất.

e. Tìm phần tử âm lớn nhất / phần tử dương nhỏ nhất

f. Tính tổng các phần tử nằm ở vị trí chẵn trong mảng.

g. Viết hàm sắp xếp mảng theo thứ tự tăng dần.

1.4 TÓM TẮT LÝ THUYẾT MẢNG 2 CHIỀU

1.4.1 Khai báo và truy cập

Mảng hai chiều thực chất là mảng một chiều trong đó mỗi phần tử của mảng là một mảng một chiều, và được truy xuất bởi hai chỉ số dòng và cột.

Khai báo

```
Kiểu_dữ_liệu  Tên_mảng [Số dòng tối đa ][Số cột tối đa];
```

Ví dụ:

int A[10][10]; // Khai báo mảng 2 chiều kiểu int gồm 10 dòng, 10 cột

float b[10][10]; // Khai báo mảng 2 chiều kiểu float gồm 10 dòng, 10 cột

Truy cập

```
Tên_mảng [chỉ số dòng][chỉ số cột];
```

1.4.2 Ma trận vuông và các khái niệm liên quan

Khái niệm: Là ma trận có số dòng và số cột bằng nhau.Trong đó:

Đường chéo chính là đường chéo có ***chỉ số dòng = chỉ số cột.***

Đường chéo phụ là đường chéo có: ***chỉ số cột + chỉ số dòng = số dòng (hoặc số cột)***

1.5 THỰC HÀNH MẢNG 2 CHIỀU

Câu 1: Viết chương trình gồm các hàm sau:

a. Nhập ma trận gồm *d* dòng và *c* cột (*d, c* nhập từ bàn phím)

b. Xuất ma trận

c. Tính tổng các phần tử của ma trận

d. Tính trung bình cộng các phần tử trong ma trận

e. Tính trung bình cộng các phần tử dương trong ma trận

f. Xuất các phần tử nằm trên dòng k (k do người dùng nhập) trong ma trận

g. Tính tổng các phần tử nằm trên cột k (k do người dùng nhập) trong ma trận

h. Tìm phần tử lớn nhất trong ma trận

Hướng dẫn:

```
#define MAX 10
typedef int MATRAN[MAX][MAX];
//nhập số dòng và số cột của ma trận----------------------------
void NhapDC(int &d, int &c)
{
   //bạn tự code
}
//nhập giá trị cho từng phần tử của ma trận----------------------
void Nhap (MATRAN a, int d, int c )
{
      for ( int i = 0; i < d; i ++ )
            for (int j = 0; j < c; j ++)
            {
                  printf ("a[%d][%d] = ", i, j );
                  scanf ("%d", &a[i][j]);
            }
}
//xuất ma trận---------------------------------------------------
void Xuat (MATRAN a, int d, int c){
      printf ("\nNoi dung ma tran:\n");
      for (int i = 0; i < d; i++)
      {
            for (int j = 0; j < c; j++)
                  printf ("\t%d ", a[i][j] );
            printf ("\n");
      }
}
```

Câu 2: Ma trận vuông cấp *n* là mảng 2 chiều có (số dòng) = (số cột) = *n*.

a. Sinh ngẫu nhiên 1 ma trận vuông cấp *n* chứa số nguyên (n nhập từ bàn phím).

b. Xuất ma trận.

c. Liệt kê các phần tử trên đường chéo chính.

d. Liệt kê các phần tử trên đường chéo phụ.

e. Tính tổng các phần tử nằm trên dòng thứ *k* (*k* do người dùng nhập).

f. Tính tổng các phần tử trên mỗi dòng.

g. Xuất ra các dòng có tổng lớn nhất.

Câu 3: Viết chương trình nhập vào ma trận vuông kích thước n x n ($2 \leq n \leq 100$). Hãy viết hàm thực hiện những công việc sau :

a. In ra các phần tử trên 4 đường biên của ma trận.

b. Tính tổng các phần tử trên biên.

c. Kiểm tra xem ma trận vuông có đối xứng qua đường chéo chính hay không.

Câu 4: Viết chương trình tính tổng, tích của hai ma trận các số nguyên.

BÀI 2: KIỂU DỮ LIỆU CÓ CẤU TRÚC

Sau khi học xong bài này, sinh viên có thể:

- *Cách khai báo các kiểu dữ liệu mới để giải quyết theo yêu cầu của bài toán dựa vào những kiểu dữ liệu cơ bản được cài đặt sẵn trong ngôn ngữ lập trình.*
- *Biết nhập, xuất dữ liệu có cấu trúc và lưu trên mảng một chiều.*
- *Đi sâu vào các giải thuật trên mảng một chiều như tìm kiếm, thêm phần tử, xóa phần tử theo từng thành phần của kiểu dữ liệu có cấu trúc.*

2.1 TÓM TẮT LÝ THUYẾT

2.1.1 Định nghĩa cấu trúc

Cấu trúc (struct) thực chất là một kiểu dữ liệu do người dùng định nghĩa bằng cách gom nhóm các kiểu dữ liệu cơ bản có sẵn trong C thành một kiểu dữ liệu phức hợp nhiều thành phần.

Cú pháp định nghĩa kiểu cấu trúc

```
struct <tên cấu trúc>
{
        <Kiểu> <Trường 1>;
        <Kiểu> <Trường 2>;
        …
        <Kiểu> <Trường n>;
};
//dùng từ khoá typedef để định nghĩa một tên mới cho kiểu dữ liệu đã có.
typedef struct <tên cấu trúc> <tên mới>;
```

Khai báo biến kiểu cấu trúc

```
<tên mới> <tên biến>;
```

Truy xuất

```
<Tên biến cấu trúc>.<tên thành phần>;
```

2.1.2 Mảng cấu trúc

Cách khai báo tương tự như mảng một chiều (Kiểu dữ liệu bây giờ là kiểu dữ liệu có cấu trúc).

Cách truy cập phần tử trong mảng cũng như truy cập trên mảng một chiều. Nhưng do từng phần tử có kiểu cấu trúc nên phải chỉ định rõ cần lấy thành phần nào.

Nguyên tắc viết chương trình có mảng cấu trúc:

Do kiểu dữ liệu có cấu trúc thường chứa rất nhiều thành phần nên khi viết chương trình loại này ta cần lưu ý:

- Xây dựng hàm xử lý cho một cấu trúc.
- Muốn xử lý cho mảng cấu trúc, ta gọi lại hàm xử lý cho một cấu trúc đã được xây dựng bằng cách dùng vòng lặp.

2.2 THỰC HÀNH CƠ BẢN

Câu 5: Định nghĩa kiểu dữ liệu sinh viên, thông tin của mỗi sinh viên gồm:

- mã sinh viên
- họ tên
- giới tính (x: nữ, y: nam)
- ngày sinh (gồm ngày, tháng và năm)
- lớp (chuỗi 7 kí tự với 2 ký tự đầu là năm học, 1 kí tự tiếp theo là bậc học (D: đại học, C: cao đẳng), 2 kí tự tiếp theo là ngành học
- điểm trung bình.

Viết chương trình gồm các hàm thực hiện:

a. Nhập danh sách sinh viên

b. Xuất danh sách sinh viên

c. Xuất các sinh viên có điểm trung bình >5.

d. Xuất danh sách sinh viên thuộc ngành công nghệ thông tin

e. Đếm số lượng sinh viên nữ.

f. Xuất các sinh viên có điểm trung bình cao nhất.

g. Thêm một sinh viên mới vào cuối danh sách.

h. Tìm sinh viên có mã là X. Nếu tìm thấy hãy xoá sinh viên đó khỏi danh sách.

i. Sắp xếp danh sách tăng theo điểm trung bình.

Hướng dẫn:

- Trước hết ta phải định nghĩa kiểu dữ liệu ngày tháng năm (đặt tên kiểu là DATE), định nghĩa kiểu dữ liệu sinh viên (đặt tên kiểu là SV).
- Xây dựng hàm nhập và xuất ngày tháng năm.
- Muốn nhập được danh sách sinh viên ta phải xây dựng hàm nhập cho 1 sinh viên.
- Sau đó mới xây dựng hàm nhập cho danh sách sinh viên.
- Tham khảo đoạn code sau minh hoạ cho câu a và b, các câu còn lại học viên tự làm tiếp.

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#define MAX 100
//định nghĩa kiểu dữ liệu
struct DATE
{
    unsigned char ngay;
    unsigned char thang;
    int nam;
};
typedef struct SinhVien
{
    char ma[11];
    char hoten[31];
```

```
        struct DATE ns; //ngày sinh
        char gt;  //giới tính lưu đại diện bởi 1 kí tự x: nữ, y: nam
        char lop[8];
        float dtb;
}SV;
//khai báo các nguyên mẫu hàm sẽ dùng trong chương trình
void NhapNgaySinh(DATE &d);
void XuatNgaySinh(DATE d);
void Nhap1sv (SV &x);
void Xuat1sv (SV x);
void Nhapsl(int &n);
void Nhapds(SV a[], int n);
void Xuatds(SV a[], int n);
//=========================================
//hàm chính của chương trình
int main()
{
        SV  a[MAX]; //Khai báo mảng a gồm có tối đa 100 sinh viên
        int n; //n lưu số lượng sinh viên của danh sách
        Nhapsl(n); //nhập số lượng sinh viên
        Nhapds(a, n); //nhập danh sách
        Xuatds(a, n);
        return 0;
}
//---------------------------------------------------------
//Code chi tiết các hàm con
void NhapNgaySinh(DATE &d)
{
        printf("\nNhap vao ngay: ");
        scanf("%u", &d.ngay);
        printf("\nNhap vao thang: ");
        scanf("%u", &d.thang);
        printf("\nNhap vao nam: ");
        scanf("%d", &d.nam);
}
void XuatNgaySinh(DATE d)
{
```

```
        printf("%02u / %02u / %4d", d.ngay, d.thang, d.nam);
    }
    //---------------------------------------------------
    //nhập thông tin cho 1 sinh viên
    void Nhap1sv (SV &x)
    {

        printf("Nhap ma sinh vien: "); scanf("%s", &x.ma);

        printf("Nhap ho ten: ");
        fflush(stdin); //Xoa vung dem
        gets(x.hoten);

        printf("Nhap ngay thang nam sinh: "); NhapNgaySinh(x.ns);
        printf("Nhap lop: "); scanf("%s", &x.lop);

        do{
            printf("Nhap gioi tinh x-nu, y-nam: ");
            x.gt=getche();
        }while (x.gt!='x' && x.gt!='y');

        printf("\nNhap vao diem trung binh: "); scanf("%f", &x.dtb);
    }
    //---------------------------------------------------
    //xuất thông tin của 1 sinh viên ra màn hình
    void Xuat1sv (SV x)
    {
        printf("\n%-11s\t-30s\t", x.ma, x.hoten);
        if (x.gt=='x') printf("nu\t"); else printf("nam\t");
        XuatNgaySinh(x.ns);
        printf("\t%-8s\t%.1f\n", x.lop, x.dtb);
    }//nếu xuất ra chưa đẹp, học viên hãy chỉnh sửa giá trị thông số trong chuỗi định
dạng ☺
    //---------------------------------------------------
    //Nhập số lượng sinh viên của danh sách
    void Nhapsl(int &n)
    {
```

```
    //nhập số lượng n>0, nếu nhập sai bắt nhập lại
    // học viên tự code
}
//Nhập thông tin của từng sinh viên trong danh sách
void Nhapds(SV a[], int n)
{
    printf("\NHAP DANH SACH SINH VIEN---------- \n);
      for(int i=0; i<n; i++)
      {
            printf("\nNhap sinh vien thu %d:\n", i+1);
            Nhap1sv(a[i]); //Gọi hàm nhập thông tin cho 1 sinh viên thứ i trong ds.
      }
}
//---------------------------------------------------------
//xuất danh sách sinh viên ra màn hình
void Xuatds(SV a[], int n)
{
    printf("\NDANH SACH SINH VIEN----------------\n);
      for(int i=0; i<n; i++)
            Xuat1sv(a[i]); // Gọi hàm xuất thông tin cho 1 sinh viên thứ i trong ds.
}
```

Câu 6: Viết chương trình quản lý các bưu kiện của bưu điện, sử dụng mảng một chiều để lưu danh sách các bưu kiện. Thông tin mỗi bưu kiện gồm: mã bưu kiện, tên người gửi, tên người nhận, trọng lượng, ngày gửi (ngày, tháng, năm), nội dung bưu kiện, đơn giá gửi.

Chương trình có các chức năng sau:

a. Nhập thông tin của các bưu kiện

b. Xuất thông tin của các bưu kiện

c. Thêm một bưu kiện vào danh sách

d. Sắp xếp danh sách các bưu kiện theo mã bưu kiện

e. Tính giá trị của mỗi bưu kiện biết giá trị=trọng lượng x đơn giá gửi.

f. Đếm số lượng bưu kiện có trọng lượng lớn nhất

g. Tìm bưu kiện theo tên người gửi

h. Xuất ra các bưu kiện gửi vào tháng 04/2014.

2.3 THỰC HÀNH NÂNG CAO

Câu 1: Viết chương trình quản lý các thuê bao điện thoại, sử dụng mảng 1 chiều để lưu danh sách các thuê bao. Thông tin mỗi thuê bao gồm: mã thuê bao, họ tên chủ thuê bao, ngày đăng ký (ngày tháng năm), số điện thoại, loại thuê bao (TT: thuê bao trả trước, TS: thuê bao trả sau), thời gian gọi nội mạng, thời gian gọi ngoại mạng (đơn vị là phút).

Chương trình có các chức năng sau:

a. Nhập thông tin của các thuê bao

b. Xuất thông tin của các thuê bao

c. Thêm một thuê bao vào danh sách

d. Sắp xếp danh sách các thuê bao theo mã thuê bao

e. Tìm thuê bao theo họ tên chủ thuê bao

f. Xuất các thuê bao theo loại (loại nào là do người dùng chọn)

g. Xuất các thuê bao đăng kí sau năm 2010

h. Tính cước phí của mỗi thuê bao biết giá cước gọi nội mạng là 1500đ, ngoại mạng là 3000đ

i. Đếm số lượng thuê bao trả trước.

Câu 2: Viết chương trình quản lý sách cho một cửa hàng sách, sử dụng mảng 1 chiều để lưu các cuốn sách; thông tin của mỗi cuốn sách gồm: mã sách, tên sách, tên tác giả, loại sách (gồm 2 loại Tự nhiên và Xã hội), năm xuất bản, giá tiền, số lượng.

Chương trình có các chức năng sau:

a. Nhập thông tin các cuốn sách.

b. Xuất thông tin các cuốn sách.

c. Thêm 1 cuốn sách.

d. Tính tổng thành tiền tất cả cả các cuốn sách.

e. Sắp xếp các cuốn sách theo mã sách.

f. Tìm sách theo tên sách.

g. Xuất các cuốn sách có năm xuất bản trước năm 2000.

h. Đếm số lượng sách có giá lớn hơn 50000.

i. Xuất các cuốn sách theo loại (xuất loại nào là do người dùng chọn).

BÀI 3: KIỂU CON TRỎ

Sau khi thực hành xong bài này, sinh viên có thể nắm được:

- *Biết cách khai báo và sử dụng biến kiểu con trỏ;*
- *Biết xử lý các thao tác trên mảng một chiều theo kiểu con trỏ;*
- *Biết xử lý các thao tác trên chuỗi theo kiểu con trỏ;*
- *Biết xử lý các phép toán trên mảng hai chiều theo kiểu con trỏ;*
- *Đi sâu vào các giải thuật trên mảng 1 chiều , 2 chiều như tìm kiếm, sắp xếp, thêm phần tử, xóa phần tử...theo kiểu con trỏ;*
- *Biết cách dùng con trỏ với kiểu dữ liệu có cấu trúc.*

3.1 TÓM TẮT LÝ THUYẾT

3.1.1 Khai báo biến con trỏ

Cú pháp: <Kiểu> *<Tên con trỏ>

Ý nghĩa: Khai báo một biến có tên là ***Tên con trỏ*** dùng để chứa địa chỉ của các biến có kiểu ***Kiểu.***

Ví dụ 1: Khai báo 2 biến a,b có kiểu int và 2 biến pa, pb là 2 biến con trỏ

```
int a, b, *pa, *pb;
```

Ví dụ 2: Khai báo biến f kiểu float và biến pf là con trỏ float

```
float  f, *pf;
```

3.1.2 Các thao tác trên con trỏ

3.1.2.1 Gán địa chỉ của biến cho biến con trỏ

Toán tử **&** dùng để định vị con trỏ đến địa chỉ của một biến đang làm việc.

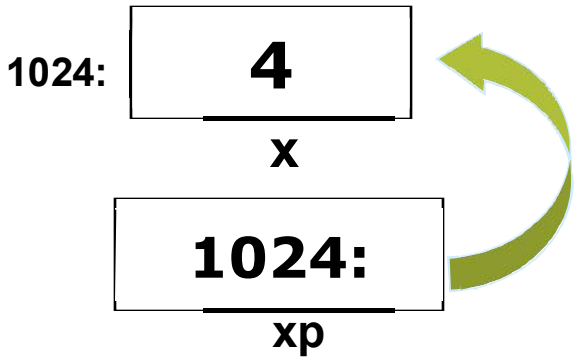
Cú pháp: **<Tên biến con trỏ> = &<Tên biến>**

Giải thích: Ta gán địa chỉ của biến ***Tên biến*** cho con trỏ ***Tên biến con trỏ***.

Ví dụ: khai báo biến int *xp , x = 4 ;

Gán địa chỉ của biến x cho con trỏ xp. xp = &x

Lúc này, hình ảnh của các biến trong bộ nhớ được mô tả:



3.1.2.2 Lấy giá trị của biến con trỏ chỉ tới

Để truy cập đến nội dung của ô nhớ mà con trỏ chỉ tới, ta sử dụng cú pháp:

***<Tên biến con trỏ>**

Với cách truy cập này thì ***<Tên biến con trỏ>** có thể coi là một biến có kiểu được mô tả trong phần khai báo biến con trỏ.

Ví dụ sau đây cho phép khai báo, gán địa chỉ cũng như lấy nội dung vùng nhớ của biến con trỏ:

```
int x=100; *ptr;

ptr= &x;

int y= *ptr;
```

Lưu ý: Khi gán địa chỉ của một biến cho một biến con trỏ, mọi sự thay đổi trên nội dung ô nhớ con trỏ chỉ tới sẽ làm giá trị của biến thay đổi theo (thực chất nội dung ô nhớ và biến chỉ là một).

3.1.3 Con trỏ và mảng 1 chiều

Truy cập từng phần tử đang được quản lý bởi con trỏ theo dạng mảng:

<Tên biến>[<Vị trí>] *tương đương với* ***(<Tên biến> + <Vị trí>)**

&<Tên biến>[<Vị trí>] *tương đương với* **(<Tên biến> + <Vị trí>)**

Trong đó:

<Tên biến> là biến con trỏ,

<Vị trí> là 1 biểu thức số nguyên.

3.1.4 Con trỏ và Chuỗi ký tự

Chuỗi ký tự là trường hợp đặc biệt của mảng một chiều. Chuỗi ký tự là một dãy các phần tử, mỗi phần tử có kiểu ký tự.

Chuỗi ký tự được kết thúc bằng ký tự '\0'. Do đó khi khai báo độ dài của chuỗi luôn luôn khai báo dư 1 phần tử để chứa ký tự '\0'.

Khai báo chuỗi

Cách 1: char <Tên chuỗi> [<Số ký tự tối đa của chuỗi>] ; Ví dụ: char s[30];
Cách 2: char *<Tên chuỗi>; Ví dụ: char *s;

3.1.4.1 Nhập, xuất chuỗi

❖ **Nhập chuỗi:**

Cách 1: **gets(biến_chuỗi);**

- Nhận các ký tự nhập từ phím cho đến khi nhấn phím Enter và đưa vào biến chuỗi.

Ví dụ:

```
char s[30];
fflush(stdin); printf("Nhap chuoi: "); gets(s);
```

Cách 2: **scanf("%s", biến_chuỗi);**

Nhận các ký tự nhập từ phím cho đến khi gặp phím **space**, tab, new line, Enter thì dừng, cho nên chỉ dùng **hàm scanf để nhập chuỗi không có khoảng trắng**.

❖ **Xuất chuỗi:**

Cách 1: **puts (biến_chuỗi);** //Xuất chuỗi xong tự động xuống dòng

Cách 2: **printf ("%s", biến_chuỗi);**

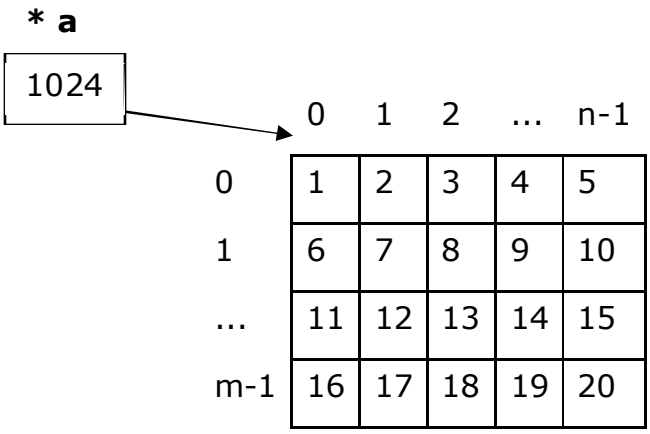
3.1.4.2 Một số hàm thư viện thường dùng (thư viện <string.h>)

SỐ TT	TÊN HÀM	CHỨC NĂNG
1	int strlen(char s[]);	Trả về độ dài của chuỗi s.
2	strcpy(char des[], char src[]);	Sao chép nội dung chuỗi src vào chuỗi des.
3	strncpy(char dest[], char src[], int n);	Chép n ký tự từ chuỗi src sang chuỗi dest. Nếu chiều dài src <n thì hàm sẽ điền khoảng trắng cho đủ n ký tự vào dest.
4	strcat(char s1[],char s2[]);	Nối chuỗi s2 vào chuỗi s1.
5	strncat(char s1[],char s2[],int n)	Nối n ký tự đầu tiên của chuỗi s2 vào chuỗi s1.
6	Int strcmp(chars1[],char s2[])	So sánh 2 chuỗi s1 và s2 theo nguyên tắc thứ tự từ điển. Phân biệt chữ hoa và thường. Trả về: • 0 : nếu s1 bằng s2. • >0: nếu s1 lớn hơn s2. • <0: nếu s1 nhỏ hơn s2.
7	int stricmp(char s1[],char s2[])	Tương tự như strcmp(), nhưng không phân biệt hoa thường.
8	char *strstr(char s1[], char s2[]);	Tìm sự xuất hiện đầu tiên của chuỗi s2 trong chuỗi s1. Trả về: • NULL: nếu không có. • Ngược lại: Địa chỉ bắt đầu chuỗi s2 trong s1.

3.1.5 Con trỏ và mảng 2 chiều

3.1.5.1 Cách 1

int *a // khai báo con trỏ a để quản lý ma trận m dòng, n cột



a = (int *) malloc (m*n)) // xin cấp phát bộ nhớ để quản lý ma trận

Khi đó

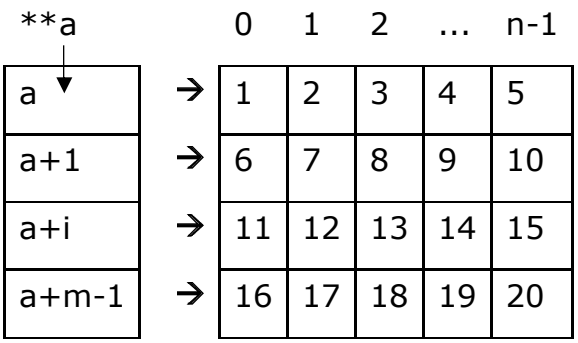
Hình ảnh ma trận mxn số nguyên lưu trong bộ nhớ do con trỏ a quản lý

A	**a +1**	**a +2**	**a +3**	**a +4**	**a +5**	**a +6**	**....**	**a+mxn-1**
1	2	3	4	5	6	7		mxn
a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]	a[0][5]	a[0][6]		a[m-1][n-1]

Để truy cập đến phần tử a [i][j] của mảng a, ta dùng công thức sau :

***(a + i*n + j);**

3.1.5.2 Cách 2



1. Con trỏ *a quản lý các phần tử dòng 0 : a[0][0], a[0][1], a[0][2],....a[0][n-1]

*a →	a[0][0]	a[0][1]	...	a[0][n-1]

2. Con trỏ *(a+1) quản lý các phần tử dòng 1 : a[1][0], a[1][1],a[1][2],....a[1][n-1]

*(a+1) →	a[1][0]	a[1][1]	...	a[1][n-1]

3. Con trỏ *(a+i) quản lý các phần tử dòng i : a[i][0], a[i][1], a[i][2],....a[i][n-1]

*(a+i) →	a[i][0]	a[i][1]	...	a[i][n-1]

4. Con trỏ *(a+m-1) quản lý các phần tử dòng m-1 : a[m-1][0], a[m-1][1], a[m-1][2],....a[m-1][n-1]

*(a+m-1) →	a[m-1][0]	a[m-1][1]	...	a[m-1][n-1]

Xin cấp phát cho con trỏ **a để quản lý mảng con trỏ: *a, *(a+1), *(a+2)..., *(a+m-1).

** a→	* a	* (a+1)	* (a+2)	...	*(a+m-1)

3.1.6 Con trỏ với kiểu dữ liệu có cấu trúc (struct)

Để truy cập đến các thành phần của struct thông qua con trỏ ta dùng một trong hai cách sau:

Cách 1:

<tên con trỏ> -> <tên thành phần>

Cách 2:

(*<tên con trỏ>).<tên thành phần>

3.2 THỰC HÀNH CƠ BẢN

Câu 1: Dùng con trỏ, viết chương trình thực hiện các chức năng sau (dùng hàm):

a. Nhập vào một số nguyên *n* (*0 < n < 100*) .

b. Xuất ra mảng số nguyên vừa nhập.

c. Tính tổng các phần tử có trong mảng.

Hướng dẫn:

a. Nhập vào một số nguyên *n* (*0 < n < 100*).

```
void NhapMang (int * a , int n)
{
      for (int  i=0 ; i<n ; i++)
     {          printf(" nhap a[%d]:", i );
                scanf (" %d ", (a+ i));
      }
}
```

b. Xuất ra mảng số nguyên vừa nhập

```
void XuatMang ( int * a, int n)
{
    for ( int i=0 ; i<n ; i++ )
          printf ("%4d", *( a+i ) );
}
```

c. Tính tổng các phần tử có trong mảng.

int TongMang(int *a, int n) {…}

Câu 2: Viết chương trình theo hàm cho phép thực hiện:

a. Nhập vào một ma trận số nguyên gồm m dòng, n cột có mxn phần tử (m, n<=100).

b. Xuất ra ma trận số nguyên vừa nhập.

c. Tính tổng các phần tử có trong ma trận.

Hướng dẫn:

Cách 1:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
// hàm nhập ma trận
```

```
void NhapMaTran (int * a , int m , int n)
{
	for (int  i=0 ; i<m ; i++)
		for ( int j=0 ; j<n ; j++)
		 {
			printf(" nhap a[%d][%d]:", i , j);
			scanf (" %d ", (a+ i*n + j ) );
		 }
}
// hàm xuất ma trận
void XuatMaTran ( int * a, int m , int n)
{
 for ( int i=0 ; i<m ; i++ )
 {
	for ( int j=0 ; j<n ; j++)

		printf ("%4d", *( a+i *n +j ) );
    printf("\n");
 }
 }
// hàm tính tổng các phần tử trong  ma trận
long TinhTong ( int * a, int m , int n)
{
 for ( int i=0 ; i<m ; i++ )
	for ( int j=0 ; j<n ; j++)
		s = s+ *( a+i *n +j ) ;
}
```

Cách 2: Tham khảo tài liệu và tự làm nhé!

Câu 3: Viết chương trình thực hiện:

a. Nhập vào 2 chuỗi dữ liệu s1 và s2.

b. Xuất ra màn hình hai chuỗi vừa nhập

c. Xuất ra màn hình độ dài của các chuỗi vừa nhập.

d. So sánh hai chuỗi s1 và s2

e. Nối chuỗi s2 vào chuỗi s1

f. Kiểm tra chuỗi s1 có chứa chuỗi s2 hay không?

g. Kiểm tra chuỗi s2 có chứa chuỗi s1 hay không?

Câu 4: Đổi các từ ở đầu câu sang chữ hoa và những từ không phải đầu câu sang chữ thường.

Ví dụ: nGuYen vAN an đổi thành: Nguyễn Văn An

3.3 THỰC HÀNH NÂNG CAO

Câu 1: Viết các hàm sau:

a. Viết hàm nhập vào một phân số

b. Viết hàm nhập vào một dãy gồm n phân số

c. Viết hàm xuất một phân số

d. Viết hàm xuất một dãy gồm n phân số

e. Viết hàm tìm phân số lớn nhất trong dãy phân số

f. Viết hàm tính tổng các phân số có trong dãy

Câu 2: Thông tin về một sinh viên gồm có: Họ tên, mã số, ngày tháng năm sinh, giới tính, lớp, điểm toán, điểm lý, điểm tin.

a. Viết hàm nhập dữ liệu cho một sinh viên.

b. Viết hàm xuất dữ liệu một sinh viên.

c. Viết hàm nhập danh sách sinh viên.

d. Viết hàm xuất danh sách sinh viên.

e. Xuất thông tin của sinh viên có mã sinh viên nhập từ bàn phím.

f. Xuất danh sách sinh viên thuộc ngành công nghệ thông tin

g. Xuất danh sách sinh viên Nữ thuộc ngành công nghệ thông tin

h. Tìm sinh viên theo tên

i. Tìm một sinh viên theo MSSV

j. Xuất danh sách sinh viên theo lớp

BÀI 4: ĐỆ QUY

Sau khi thực hành xong bài này, sinh viên có thể nắm được:

- *Phương pháp lập trình theo kỹ thuật đệ quy*
- *Cách hoạt động và cách cài đặt các hàm đệ quy.*

4.1 TÓM TẮT LÝ THUYẾT

Một hàm được gọi có tính đệ qui nếu trong thân của hàm đó có lệnh gọi lại chính nó.

Kỹ thuật giải bài toán bằng đệ quy:

1. Tham số hóa bài toán.
2. Tìm các điều kiện biên (chặn, dừng), tìm giải thuật cho các tình huống này.
3. Tìm giải thuật tổng quát theo hướng đệ quy lui dần về tình huống bị chặn.

Lưu ý

Đệ qui cung cấp cho ta cơ chế giải quyết các bài toán phức tạp một cách đơn giản hơn.

- Xây dựng hàm đệ qui thông qua việc xác định điều kiện dừng và bước thực hiện tiếp theo.
- Chỉ nên cài đặt bằng phương pháp đệ qui khi không còn cách giải quyết bằng cách lặp thông thường.

4.2 THỰC HÀNH CƠ BẢN

Câu 1: Viết chương trình thực hiện (dùng đệ quy):

a. Nhập 1 mảng số nguyên.

b. Xuất mảng số nguyên

c. Tính tổng các phần tử của mảng

d. Tính tổng các phần tử chẵn của mảng

e. Đếm số lượng phần tử dương

f. Tìm phần tử lớn nhất (nhỏ nhất) của mảng

g. Tìm phần tử chẵn cuối cùng có trong mảng

h. Nhập 1 trị x, tìm vị trí có x cuối cùng trong mảng.

Hướng dẫn:

```
void Nhap(int a[], int n)
{
    if(n==0)
        return;
    Nhap(a, n-1); //nhập cho n-1 phần tử đầu
    printf("\nNhap phan tu thu %d: ", n-1);
    scanf("%d", &a[n-1]);  //nhập cho phần tử cuối trong mảng
}
void Xuat(int a[], int n)
{
    if(n==0)
        return;
    Xuat(a, n-1);
    printf("%d\t", a[n-1]);
}
```

Câu 2: Tính dãy Fibonaci sử dụng đệ quy.

Câu 3: Viết chương trình thực hiện (dùng đệ quy):

a. Tìm USCLN của 2 số nguyên dương.

b. Tìm BSCNN của 2 số nguyên dương.

4.3 THỰC HÀNH NÂNG CAO

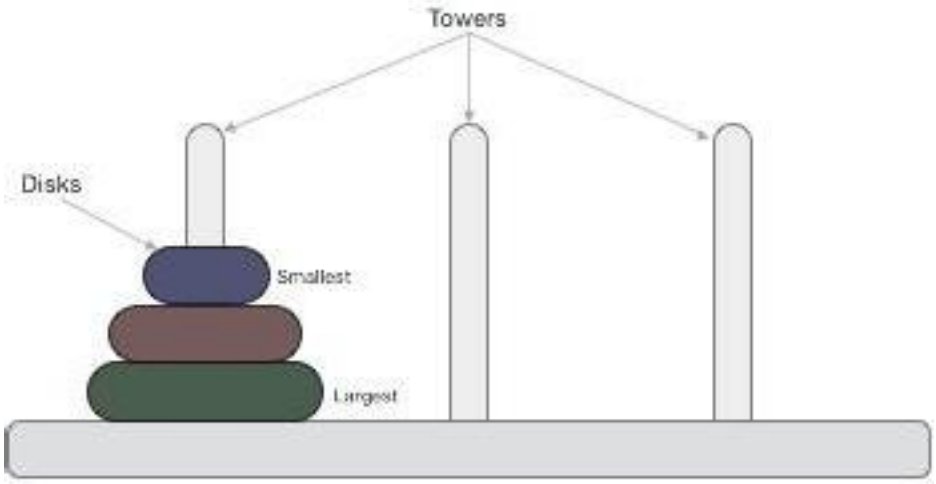
Bài toán Tháp Hà Nội (Tower of Hanoi) sử dụng đệ quy

Trước khi tìm hiểu lời giải cho bài toán Tháp Hà Nội (Tower of Hanoi), chúng ta cần biết một số qui tắc của trò chơi toán Tháp Hà Nội này:

1. Tháp Hà Nội (Tower of Hanoi) là gì ?

Bài toán Tháp Hà Nội (Tower of Hanoi) là một trò chơi toán học bao gồm 3 cột và số đĩa nhiều hơn 1.

Dưới đây là hình minh họa bài toán Tháp Hà Nội (Tower of Hanoi) với trường hợp có 3 đĩa.



Các đĩa có kích cỡ khác nhau và xếp theo tự tự tăng dần về kích cỡ từ trên xuống: đĩa nhỏ hơn ở trên đĩa lớn hơn. Với số đĩa khác nhau thì ta có các bài toán Tháp Hà Nội (Tower of Hanoi) khác nhau, tuy nhiên lời giải cho các bài toán này là tương tự nhau. Lời giải tối ưu cho bài toán Tháp Hà Nội (Tower of Hanoi) là khi trò chơi chỉ có 3 cọc. Với số cọc lớn hơn thì lời giải bài toán vẫn chưa được khẳng định.

2. Qui tắc trò chơi toán học Tháp Hà Nội (Tower of Hanoi)

Nhiệm vụ của trò chơi là di chuyển các đĩa có kích cỡ khác nhau sang cột khác sao cho vẫn đảm bảo thứ tự ban đầu của các đĩa: đĩa nhỏ nằm trên đĩa lớn. Dưới đây là một số qui tắc cho trò chơi toán học Tháp Hà Nội (Tower of Hanoi):

- Mỗi lần chỉ có thể di chuyển một đĩa từ cột này sang cột khác.

- Chỉ được di chuyển đĩa nằm trên cùng (không được di chuyển các đĩa nằm giữa).
- Đĩa có kích thước lớn hơn không thể được đặt trên đĩa có kích thước nhỏ hơn.

Bài toán Tháp Hà Nội (Tower of Hanoi) với số đĩa là n có thể được giải với số bước tối thiểu là $\mathbf{2^n-1}$. Do đó, với trường hợp 3 đĩa, bài toán Tháp Hà Nội (Tower of Hanoi) có thể được giải sau $\mathbf{2^3-1} = 7$ bước.

BÀI 5: TẬP TIN (FILE) – TÌM KIẾM

Sau khi học xong bài này, học viên có thể:

- *Cấu trúc tập tin, cài đặt các thao tác, một số hàm thư viện và ứng dụng trong việc tổ chức dữ liệu trên tập tin.*
- *Kỹ thuật tìm kiếm tuyến tính và tìm kiếm nhị phân.*

5.1 TÓM TẮT LÝ THUYẾT TẬP TIN (FILE)

Trong các chương trình trước thì các dữ liệu đưa vào chương trình chỉ được tồn tại trong RAM, khi thoát chương trình thì tất cả dữ liệu đều bị mất. Để khắc phục tình trạng này Dev-C cung cấp cho ta các hàm để lưu trữ và truy xuất tập tin, đó là kiểu FILE. Và ở đây ta chỉ đề cập đến 2 loại tập tin:

- Tập tin văn bản: là tập tin dùng để ghi các ký tự lên đĩa theo các dòng.
- Tập tin nhị phân: là tập tin dùng để ghi các cấu trúc dạng nhị phân (được mã hoá).

Quá trình thao tác trên tập tin thông qua 4 bước:

- Bước 1: Khai báo con trỏ trỏ đến tập tin.
- Bước 2: Mở tập tin.
- Bước 3: Các xử lý trên tập tin.
- Bước 4: Đóng tập tin.

Khai báo

```
FILE *  tên biến;
```

Ví dụ : FILE *f; *// Khai bao bien con tro file f*

Mở tập tin

```
fopen (đường dẫn tên tập tin, "kiểu truy cập");
```

Ví dụ : FILE *f = fopen ("C:\\VD1.txt" , "rt") ;

Các kiểu truy nhập tập tin thông dụng:

t là kiểu truy nhập tập tin đối với dạng tập tin văn bản (text).

b là kiểu truy nhập tập tin đối với dạng tập tin nhị phân (binary).

r mở ra để đọc (ready only).

w mở ra để ghi (create / write).

a mở ra để thêm vào (append).

r+ mở ra để đọc và ghi (modify).

Đóng tập tin

```
fclose ( <biến con trỏ tập tin> ) ;
hoặc fcloseall () ;
```

Các hàm đọc ghi tập tin văn bản

TÊN HÀM	Ý NGHĨA	VÍ DỤ
Đọc dữ liệu từ tập tin ra biến (nhập dữ liệu cho biến từ file)		
fscanf(biến file, "định dạng", các tham biến);	Đọc dữ liệu từ một tập tin theo định dạng.	int x; float y; fscanf(f, "%d%f", &x, &y);
fgets(biến chuỗi, kích thước tối đa, biến file);	Đọc một chuỗi ký tự từ một tập tin với kích thước tối đa cho phép, hoặc gặp ký tự xuống dòng.	char s[80]; fgets(s, 79, f);
biến kí tự = getc(biến file);	Đọc một ký tự từ tập tin đang mở.	char ch=getc(f);
Ghi tập tin (ghi dữ liệu (từ biến) ra tập tin		
fprintf(biến file, "chuỗi định dạng" [, <các tham biến nếu có>]);	Ghi dữ liệu theo một định dạng nào đó vào tập tin.	fprintf(f,"Ket qua la %d\n",x);
fputs(chuỗi ký tự, biến file);	Ghi một chuỗi ký tự vào tập tin đang mở.	fputs("Chuong trinh minh hoa", f);

Các hàm đọc ghi tập tin nhị phân

TÊN HÀM	Ý NGHĨA	VÍ DỤ
Đọc dữ liệu từ tập tin ra biến (nhập dữ liệu cho biến từ file)		
fread(&ptr, size, len, biến file);	• ptr: vùng nhớ để lưu dữ liệu đọc. • size: kích thước mỗi ô nhớ (tính bằng byte). • len: độ dài dữ liệu cần đọc.	int a[30], n; fread(a, sizeof(int), n , f); SV b; Fread(&b, sizeof(SV), 1 , f);
Ghi tập tin (ghi dữ liệu (từ biến) ra tập tin		
fwrite(&prt, size, len, biến file);	Tham số tương tự như hàm fread.	fwrite(a, sizeof(int), n , f);

5.2 THỰC HÀNH TẬP TIN (FILE)

Câu 1: Cho file TXT có cấu trúc sau:

- Dòng đầu lưu giá trị một số nguyên dương n
- Dòng còn lại lưu giá trị của một dãy gồm n các số nguyên.

a. Đọc file trên, lưu dữ liệu đọc được vào mảng một chiều.

b. Xuất mảng ra màn hình

c. Ghi các số nguyên tố có trong mảng vào file (ghi tiếp theo vào file đã có).

Câu 2: Viết chương trình phát sinh ngẫu nhiên ma trận a kích thước 5x6, lưu ma trận này vào file test.inp. Đọc lại file test.inp đưa dữ liệu vào ma trận b và xuất ra màn hình xem kết quả lưu đúng không? Cấu trúc của file test.inp như sau:

- Dòng đầu lưu 2 số nguyên: d, c thể hiện số dòng và số cột của ma trận.
- d dòng tiếp theo, mỗi dòng gồm c phần tử là giá trị các phần tử trên một dòng của ma trận.

Hướng dẫn:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#define MAX 100
```

```
#define duongdan "D:\\test.inp"
//sinh ma trận ngẫu nhiên--------------------------------------
void SinhMT(int a[MAX][MAX], int d, int c) {
    //bạn tự code
}
//ghi ma trận vào file-----------------------------------------
void LuuFile(int a[MAX][MAX], int d, int c) {
    FILE *f;
    f=fopen(duongdan, "wt");
    if(f==NULL) {
            printf("\nKhong tao duoc file.");
            getch();
            exit(0);
    }
fprintf(f, "%d %d\n", d, c);
for(int i=0; i<d; i++){
      for(int j=0; j<c; j++)
          fprintf(f, "%d\t", a[i][j]);
      fprintf(f, "\n");
}
fclose(f);
}
//Đọc dữ liệu từ file ra biến mảng lưu ma trận-----------------------------
void DocFile(int a[MAX][MAX], int &d, int &c)
{
    /*mở file như trên
      đọc số dòng, số cột
      đọc giá trị các phần tử của ma trận
      đóng file
    */
}
//xuất ma trận ------------------------------------------
void XuatMT(int a[MAX][MAX], int d, int c)
{
    //bạn tự code
}
//============================================
int main()
{
      int a[MAX][MAX], d=5, c=6;
      int b[MAX][MAX], x, y;
```

```
        SinhMT(a, d, c);
        LuuFile(a, d, c); //ghi dữ liệu từ mảng a xuống file

        DocFile(b, x, y); //đọc dữ liệu từ file ra mảng b
        XuatMT(b, x, y);
        return 0;
    }
```

Câu 3: Viết chương trình đọc một chuỗi tối đa 100 kí tự từ bàn phím. Lưu các ký tự là nguyên âm vào tập tin "NguyenAm.txt". Đọc các kí tự từ tập tin này và hiển thị lên màn hình.

Câu 4: Viết chương trình cho phép nhập từ bàn phím và ghi vào 1 tập tin tên DSHH.TXT với mỗi phần tử của tập tin là một cấu trúc bao gồm các trường:

- 1.mh (mã hàng: char[5]).
- sl (số lượng: int).
- dg (đơn giá: float).
- st (Số tiền: float) = số lượng * đơn giá.

Sau khi nhập xong yêu cầu in toàn bộ danh sách hàng hóa ra màn hình.

5.3 TÓM TẮT LÝ THUYẾT KỸ THUẬT TÌM KIẾM

5.3.1 Định nghĩa

Tìm kiếm là quá trình xác định một đối tượng nào đó trong một tập các đối tượng. Kết quả trả về là:

- Đối tượng tìm được (nếu có).
- Chỉ số (nếu có) xác định vị trí của đối tượng trong tập đó.

Việc tìm kiếm dựa theo một trường nào đó của đối tượng, trường này là khóa (key) của việc tìm kiếm.

VD: Tìm sinh viên có mã số là X bất kỳ trong DSSV.

5.3.2 Phương pháp tìm kiếm

Hai phương phương pháp tìm kiếm:

- Tìm kiếm tuần tự (tuyến tính) trên tập dữ liệu bất kỳ.
- Tìm kiếm nhị phân trên tập dữ liệu đã sắp xếp theo trường khóa.

5.4 THỰC HÀNH KỸ THUẬT TÌM KIẾM

5.4.1 Tìm kiếm trên dãy số nguyên

Câu 1: Viết chương trình thực hiện:

a. Nhập mảng gồm n số nguyên (n > 0).

b. Xuất mảng ra màn hình.

c. Tìm phần tử có giá trị X trong mảng, nếu có cho biết vị trí xuất hiện của X trong mảng (làm theo 2 cách tìm kiếm tuyến tính và tìm kiếm nhị phân).

Hướng dẫn:

- Chương trình minh hoạ nhập xuất mảng số nguyên

```
#include <stdio.h>
#define MAX 100
/*---------------------------------
*Hàm nhập số lượng phần tử cho mảng
*/
void NhapSL(int &n)
{
   do{
         printf ("Nhap so phan tu 0<sl<%d: ", MAX);
         scanf (" %d ", &n);
   }while (n<=0 || n>MAX);
}
/*---------------------------------
*Hàm nhập giá trị cho từng phần tử trong mảng
*/
void NhapMang (int a[], int n)
{
     for (int i = 0; i < n; i ++)
```

```
        {
            printf (" a[%d] =  ", i);
            scanf (" %d ", &a[i]);
        }
    }
    /*---------------------------------
    *Hàm xuất các phần tử của mảng ra màn hình
    */
    void XuatMang (int a[], int n)
    {
        printf ("\nMang gom cac phan tu:\n ");
        for (int i = 0; i < n; i ++)
            printf (" %5d", a[i]);
    }
    // ---------------------------------
    int TimTuyenTinh(int a[], int n, int X){...}
    void SapXep(int a[], int n){...}
    int TimNhiPhan(int a[], int n, int X){..}
    //================================
    int main()
    {
    int a[MAX], n;
    NhapMang (a,n);
    XuatMang (a,n);
    int X;
    printf("Nhap gia tri can tim: "); scanf("%d", &X);
    //gọi thực hiện tìm kiếm X bằng phương pháp tìm nhị phân
    int vt=TimTuyenTinh(a,n,X);
    if (vt==-1) printf("Khong tim thay");
    else printf("Tim thay %d tai vi tri %d",X, vt);
    //gọi thực hiện tìm kiếm X bằng phương pháp tìm nhị phân tương tự
    sapxep(a,n);
    printf("Mang sau khi sap tang dan:"); XuatMang(a,n);
    printf("Nhap gia tri can tim: "); scanf("%d", &X);
    vt=TimNhiPhan(a,n,X);
    //tương tự...
    return 0;
}
```

Mở rộng: Sau khi chạy thử hết các chức năng, chương trình chạy ra kết quả đúng, bạn hãy viết lại hàm main() theo menu chức năng.

5.4.2 Tìm kiếm trên danh sách phần tử kiểu struct

Câu 2: Viết chương trình quản lý thư viện, thông tin mỗi cuốn sách gồm: mã sách (int), tên sách (char[40]), giá (float).

a. Nhập danh sách gồm N cuốn sách.

b. Xuất danh sách các cuốn sách ra màn hình.

c. Tìm cuốn sách có mã là X (Làm theo 2 cách: tìm tuyến tính và tìm nhị phân).

d. Xuất ra các cuốn sách có tên là Y.

e. Xuất các cuốn sách có giá cao nhất (nếu có nhiều sách có giá cao nhất trùng nhau thì xuất hết ra màn hình).

Hướng dẫn:

- Khai báo cấu trúc sách

```
typedef struct Tên_cấu_trúc
{
   //khai báo các biến thành phần của cấu trúc;
   ...
}Tên_cấu_trúc_viết_gọn;

VD:
typedef struct CuonSach
{
   int masach;
   char tensach[40];
   float gia;
}Sach;
```

- Chương trình có thể tổ chức gồm các hàm sau:
  - void nhap1Sach(Sach &x)
  - void xuat1Sach(Sach x)
  - void nhapn(int &n)
  - void nhapDS(Sach a[], int n)
  - void xuatDS(Sach a[], int n)
  - int timTuanTu(Sach a[], int n, int X)

- int timNhiPhan(Sach a[], int n, int X)

- **Hàm nhập 1 cuốn sách:** nhập thông tin cho 1 cuốn sách

  void nhap1Sach(Sach &x)

- **Hàm xuất 1 cuốn sách:** xuất thông tin của 1 cuốn sách

  void xuat1Sach(Sach x)

- **Hàm nhập số lượng phần tử của danh sách:** nhập số lượng cuốn sách

  void nhapn(int &n)

- **Hàm nhập danh sách các cuốn sách:** dùng 1 mảng một chiều để lưu danh các cuốn sách, mỗi phần tử trong mảng là 1 cuốn sách.

```
void nhapDS(Sach a[], int n)
{
    //Nhập thông tin cho từng cuốn sách (Nhập a[i], i=0, 1, ..., n-1) bằng cách gọi hàm nhập 1 cuốn sách cho phần tử a[i]
    for(int i=0; i<n; i++)
    {
        printf("Nhap cuon sach thu %d: ", i);
        nhap1Sach(a[i]);
    }
}
```

- **Hàm xuất danh sách các cuốn sách:** Xuất thông tin từng cuốn sách a[i], i = 0, ..., n-1 bằng cách gọi hàm xuất 1 cuốn sách.

- **Hàm tìm cuốn sách mã là X:** Làm theo hai cách tìm tuyến tính và nhị phân.

  - int timTuanTu(Sach a[], int n, int X)
  - int timNhiPhan(Sach a[], int n, int X)

Lưu ý: X là mã cuốn sách cần tìm do người dùng nhập vào (nhập X trước khi gọi hàm tìm).

- **Hàm xuất ra các cuốn sách có tên là Y:** Duyệt danh sách, nếu gặp cuốn nào có tên là Y thì xuất thông tin cuốn sách đó ra màn hình. So sánh chuỗi dùng hàm strcmp() trong thư viện <string.h>.
- **Hàm tìm cuốn sách giá lớn nhất:**
  - Bước 1: Tìm giá lớn nhất
  - Bước 2: Duyệt mảng sách, nếu cuốn sách nào có giá = giá lớn nhất (tìm được ở bước 1) thì xuất ra màn hình.

Câu 3: Cải tiến giải thuật tìm kiếm tuyến tính bằng kỹ thuật đặt lính canh.

Câu 4: Hãy làm lại câu 2 với yêu cầu mã sách có kiểu là chuỗi tối đa 10 ký tự.

BÀI 6: SẮP XẾP

Sau khi thực hành xong bài này, sinh viên có thể nắm được:

- *Các thuật toán sắp xếp danh sách tăng dần/giảm dần theo trường khóa.*
- *Phần cuối cùng dành để kiểm tra đánh giá kết quả môn học.*

6.1 TÓM TẮT LÝ THUYẾT

6.1.1 Định nghĩa

Sắp xếp là quá trình bố trí lại các phần tử của một tập đối tượng theo một thứ tự nhất định (tăng dần/giảm dần).

6.1.2 Các phương pháp sắp xếp:

Một số phương pháp sắp xếp:

- Sắp xếp chọn (Selection sort)
- Sắp xếp nổi bọt (Bubble sort)
- Sắp xếp đổi chỗ trực tiếp (Interchange sort)
- Sắp xếp chèn (Insertion sort)

6.2 THỰC HÀNH CƠ BẢN

6.2.1 Sắp xếp trên dãy số nguyên

Câu 1: Viết chương trình thực hiện:

a. Sinh mảng ngẫu nhiên gồm n số nguyên (n>0), mỗi phần tử có giá trị $\in$(0, 100).

b. Xuất mảng ra màn hình.

c. Sắp xếp mảng tăng dần (giảm dần) bằng các thuật toán sắp xếp đã học.

d. Yêu cầu: Viết chương trình theo menu chức năng (cho phép người dùng chọn lựa công việc cần thực hiện).

Hướng dẫn:

Bạn cài đặt các hàm:

- Sinh mảng: sử dụng thư viện <stdlib.h> và <string.h>

```
void SinhMang (int a[], int n)
{
  srand(time(NULL)) ;
  for (int i = 0; i < n; i ++)
  {
        printf (" a[%d] =  ", i);
        a[i]=rand()%100 ;
  }
}
```

- Xuất mảng.
- Sắp xếp mảng bằng các giải thuật đã học.
- Sau khi test hết các hàm đã chạy ra kết quả đúng, tổ chức lại hàm main() theo menu chức năng như sau:

```
void xuatMenu(int &chon)
{
  //nhập chọn lựa của người dùng
  printf("1: Sinh mang\n");
  printf("2: Xuat mang\n");
  printf("3: Interchange Sort\n");
  printf("4: Bubble Sort\n");
  ...
  printf("0: Thoat\n");
  printf("Hay chon cong viec:"); scanf("%d", &chon);
}
int main()
{
  //khai báo các biến cần dùng
```

```
...
//cài đặt chương trình theo menu chức năng
//dùng một biến nguyên để lưu công việc mà người dùng chọn
int chon;
do{
        xuatMenu(chon) ;
        //thực hiện công việc cho lựa chọn tương ứng
        switch (chon){
                case 1:      //Gọi hàm Sinh mảng
                        ...
                break;
                case 2:      //Gọi hàm xuất mảng
                        ...
                break;
                case 3:
                        //Gọi hàm sắp xếp bằng Interchange
                        ...
                        //Xuất mảng để xem kết quả sau khi sắp
                        printf("Mang sau khi sap la: \n");
                        XuatMang(a,n);
                break;
                case 4:
                        //Gọi hàm sắp xếp bằng Bubble
                        ...
                        //Xuất mảng để xem kết quả sau khi sắp
                        printf("Mang sau khi sap la: \n");
                        XuatMang(a,n);
                break;
                case 5: ... //tương tự
                        ...
                default:
                        chon=0;
                break;
        }
        getch();
 }while (chon!=0);
}
```

6.2.2 Sắp xếp trên dãy phần tử kiểu struct

Câu 2: Viết chương trình quản lý nhân viên, thông tin mỗi nhân viên gồm: mã nhân viên, họ tên, lương.

a. Nhập danh sách gồm N nhân viên.

b. Xuất danh sách nhân viên ra màn hình.

c. Sắp xếp danh sách tăng dần theo lương bằng các thuật toán sắp xếp đã học.

d. Sắp xếp danh sách tăng dần theo lương bằng các thuật toán sắp xếp đã học.

TÀI LIỆU THAM KHẢO

1. Phạm Văn Ất, Kỹ thuật Lập trình C- cơ bản và nâng cao, NXB KH & KT - 2003.
2. Nguyễn Tấn Trần Minh Khang, Bài giảngKỹ Thuật Lập trình, Khoa Công Nghệ Thông Tin, Đại học Khoa học Tự nhiên
3. Nguyễn Thanh Thủy (chủ biên), Nhập môn lập trình ngôn ngữ C, Nhà xuất bản Khoa học kỹ thuật – 2000.
4. Trần Minh Thái, Bài giảng và bài tập Lập trình căn bản; Khoa Công Nghệ Thông Tin, Đại học Khoa học Tự nhiên
5. Trần Minh Thái, Giáo Trình Bài Tập Kỹ Thuật Lập Trình, Cao đẳng Công Nghệ Thông Tin Tp. Hồ Chí Minh
6. Brain W. Kernighan & Dennis Ritchie, The C Programming Language, Prentice Hall Publisher, 1988.