BỘ GIÁO DỤC VÀ ĐÀO TẠO

ĐẠI HỌC CÔNG NGHỆ TP.HCM

THỰC HÀNH KỸ THUẬT LẬP TRÌNH NÂNG CAO

Biên Soạn:

ThS. Văn Thị Thiên Trang

www.hutech.edu.vn

THỰC HÀNH KỸ THUẬT LẬP TRÌNH NÂNG CAO

Ấn bản 2018

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 Nâng cao cung cấp cho sinh viên những kỹ năng thực hành nâng cao về ngôn ngữ lập trình C. Môn học này là củng cố kiến thức lập trình và hỗ trợ tiếp thu các môn học chuyên ngành 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 thưc tế.

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

- Các kỹ thuật xử lý mảng một chiều và hai chiều.
- Vận dụng được các hàm thư viện để xử lý chuỗi kí tự.
- Biết xây dựng và xử lý các bài toán trên dữ liệu có cấu trúc do người dùng định nghĩa.
- Cách lưu trữ và xử lý các file trong C.
- Kỹ thuật giải bài toán bằng phương pháp đệ quy.

NỘI DUNG MÔN HỌC

- Bài 1 KỸ THUẬT XỬ LÝ MẢNG MỘT CHIỀU.
- Bài 2 MẢNG HAI CHIỀU.
- Bài 3 KIỂU DỮ LIỆU CÓ CẤU TRÚC
- Bài 4 TẬP TIN (FILE)
- Bài 5 ĐỆ QUY

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 nâng cao là môn học giúp sinh viên luyện tập củng cố phương pháp lập trình trên máy tính, giúp sinh viên rèn luyện 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 cuối kỳ (bài thi thực hành trên máy).

BÀI 1: KỸ THUẬT XỬ LÝ MẢNG MỘT CHIỀU

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

- *Ôn lại cách xây dựng và cách sử dụng hàm trong C.*
- *Các thao tác nâng cao trên mảng một chiều.*
- *Xử lý chuỗi kí tự sử dụng các hàm thư viện.*

1.1 TÓM TẮT LÝ THUYẾT

Chương trình phải tổ chức dưới dạng các hàm, trong đó hàm là một đoạn chương trình con độc lập thực hiện trọn vẹn một công việc nhất định sau đó trả về giá trị cho chương trình gọi nó, hay nói cách khác hàm là sự chia nhỏ của chương trình.

1.1.1 Cấu trúc một chương trình C theo hàm

- Đầu chương trình là các khai báo về sử dụng thư viện, khai báo hằng số, khai báo các biến toàn cục và khai báo các kiểu dữ liệu tự định nghĩa, khai báo hàm con (các nguyên mẫu hàm).
- Nguyên mẫu hàm:

```
<Kiểu dữ liệu của hàm> Tên hàm ([danh sách các tham số]);
```

Nguyên mẫu hàm thực chất là dòng đầu của hàm thêm dấu chấm phẩy (;) vào cuối, tuy nhiên tham số trong nguyên mẫu hàm có thể bỏ phần tên.

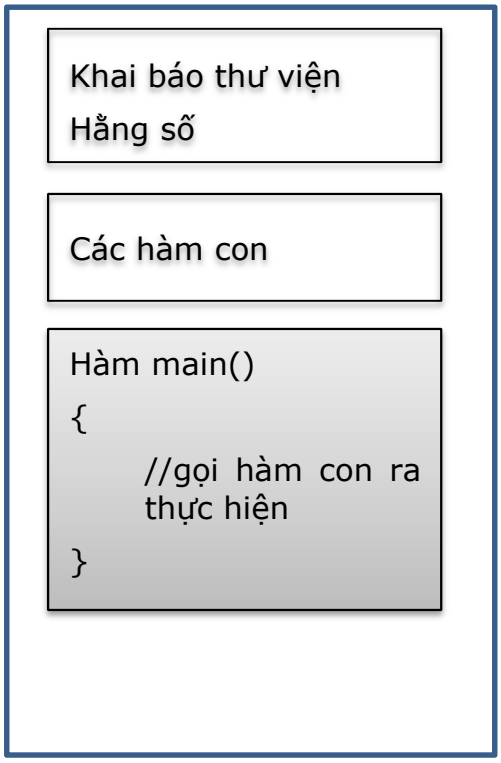
- Hàm chính (main()): Chứa các biến, các lệnh và các lời gọi hàm cần thiết trong chương trình.
- Các hàm con được sử dụng nhằm mục đích:

- Khi có một công việc giống nhau cần thực hiện ở nhiều vị trí.
- Khi cần chia một chương trình lớn phức tạp thành các đơn thể nhỏ (hàm con) để chương trình được trong sáng, dễ hiểu trong việc xử lý, quản lý việc tính toán và giải quyết vấn đề.

Có thể tổ chức chương trình theo cách 1 hoặc cách 2. Tuy nhiên, tổ chức theo cách 1 thể hiện tính chuyên nghiệp hơn.

Cách 1:

Cách 2:



Cách xây dựng một hàm con

- Định nghĩa một hàm bao gồm:
  - Khai báo kiểu hàm
  - Đặt tên hàm
  - Khai báo các tham số
  - Các câu lệnh cần thiết để thực hiện chức năng của hàm.

- Có 2 loại hàm:
  - **void**: Hàm không trả về trị. Những hàm loại này thường rơi vào những nhóm chức năng: **Nhập / xuất dữ liệu , thống kê, sắp xếp, liệt kê.**
  - Kiểu dữ liệu cơ bản (rời rạc/ liên tục) hay kiểu dữ liệu có cấu trúc: Hàm trả về trị. Sau khi thực hiện xong, kết quả được lưu lại. Những hàm loại này thường được sử dụng trong các trường hợp: Đếm, kiểm tra, tìm kiếm, tính trung bình, tổng, tích, …

- Mẫu hàm:

Hàm không trả về trị:	Hàm trả về trị:
void Ten_Ham(ds tham so neu co) { Khai báo các biến cục bộ; Các câu lệnh / khối lệnh hay lời gọi đến hàm khác; }	Kieu_dl Ten_Ham(ds tham so neu co) { Kiểu_dl kq; Khai báo các biến cục bộ; Các câu lệnh / khối lệnh hay lời gọi đến hàm khác; return kq; } *//kiểu dữ liệu của hàm là kiểu của kết quả trả về.*

❖ **Tham số của hàm**

Xác định dựa vào dữ liệu đầu vào của bài toán (Input). Gồm 2 loại :

- Tham trị: Giá trị của biến tham số đầu vào không thay đổi sau khi hàm thực hiện xong. Tham số dạng này chỉ mang ý nghĩa là dữ liệu đầu vào.

  Ví dụ: int KiemTraNguyenTo(int n);

- Tham biến: Có sự thay đổi giá trị của tham số trong quá trình thực hiện và cần lấy lại giá trị đó sau khi ra khỏi hàm. Ứng dụng của tham số loại này có thể là dữ liệu đầu ra (kết quả) hoặc cũng có thể vừa là dữ liệu đầu vào vừa là dữ liệu đầu ra.

  Ví dụ:

  Hàm nhập số nguyên: void NhapSoNguyen(int &n);

  Hàm hoán vị giá trị hai số: void HoanVi(int &a, int &b);

❖ **Cách gọi hàm con ra thực hiện**

Một hàm chỉ được thực thi khi ta có một lời gọi đến hàm đó.

Cú pháp gọi hàm:

```
<Tên hàm>([Danh sách các tham số])
```

1.1.2 Các thao tác nâng cao trên mảng một chiều

Khai báo:

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

Truy cập từng phần tử:

```
Tên_biến_mảng[chỉ số];
```

1.1.2.1 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;
}
```

1.1.2.2 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--;
}
```

1.1.2.3 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++;
}
```

1.1.2.4 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];
	}
}
```

1.1.2.5 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");
```

1.1.3 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;

1.1.3.1 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);**

1.1.3.2 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.

SỐ TT	TÊN HÀM	CHỨC NĂNG
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.

1.2 THỰC HÀNH CƠ BẢN

Câu 1: Viết chương trình 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;
     NhapMang (a,n);u
     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 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?

1.3 THỰC HÀNH NÂNG CAO

Câu 3: Viết chương trình:

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.

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

BÀI 2: MẢNG HAI CHIỀU

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

- *Các thao tác cơ bản trên mảng 2 chiều.*
- *Các thao tác với ma trận vuông*

2.1 TÓM TẮT LÝ THUYẾT

2.1.1 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];
```

2.1.2 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)***

2.2 THỰC HÀNH CƠ BẢN

Câu 1: Viết chương trình thực hiện:

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 trân

d. Tính TBC

e. Tính TBC các phần tử dương

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

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

h. Tìm phần tử lớn nhất

Hướng dẫn:

```
#define MAX 10
//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 (int a[][MAX], 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.

2.3 THỰC HÀNH NÂNG CAO

Câu 3: Viết chương trình nhập vào ma trận vuông kích thước n x n ($2 \le n \le 100$). 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 3: KIỂU DỮ LIỆU CÓ CẤU TRÚC

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

- *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, sắp xếp, thêm phần tử, xóa phần tử theo từng thành phần của kiểu dữ liệu có cấu trúc.*

3.1 TÓM TẮT LÝ THUYẾT

3.1.1 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>;
```

3.1.2 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.

3.2 THỰC HÀNH CƠ BẢN

Câu 1: 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 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 2: 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.

3.3 THỰC HÀNH NÂNG CAO

Câu 3: 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 4: 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 4: TẬP TIN (FILE)

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

- *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.*

4.1 TÓM TẮT LÝ THUYẾT

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);

4.2 THỰC HÀNH CƠ BẢN

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;
}
```

4.3 THỰC HÀNH NÂNG CAO

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.

BÀI 5: 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.*

5.1 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. Thông 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.

5.2 THỰC HÀNH CƠ BẢN

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]);
}
```

5.3 KIỂM TRA THỰC HÀNH

Chúc các bạn thi tốt!

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.