**HƯỚNG DẪN**

**Phần I. Mảng một chiều**

**Bài tập ví dụ:** Cho mảng một chiều số nguyên a, kích thước n (0<n<=100). Viết các hàm sau:

1. Nhập các phần tử vào mảng từ bàn phím và xuất mảng

**void** NhapMang(**int** a[], **int** n) {  
 **for** (**int** i = 0; i < n; i++) {  
 printf(**"Nhap phan tu tai vi tri %d: "**, i);  
 scanf(**"%d"**, &a[i]);  
 }  
}

**void** XuatMang(**int** a[], **int** n) {

**for** (**int** i = 0; i < n; i++) {

printf(**"%d "**, a[i]);

}

}

1. Viết hàm xuất các phần tử dương trong mảng

**void** XuatDuong(**int** a[], **int** n) {

**for** (**int** i = 0; i < n; i++) {

**if** (a[i] > 0)

printf(**"%d "**, a[i]);

}

}

1. Tìm vị trí phần tử có giá trị x (cho trước), nếu không có x thì trả về -1

**int** TimVTX(**int** a[], **int** n, **int** x){

**for** (**int** i = 0; i < n; i++){

**if** (a[i] == x)

**return** i;

}

**return** -1;

}

1. Xóa phần tử x (nếu có)

**void** XoaTaiVTk(**int** a[], **int** &n, **int** k){

**for** (**int** i = k; i < n - 1; i++) {

a[i] = a[i + 1];

}

n--;

}

**void** XoaX(**int** a[], **int** &n, **int** x){

**int** vt = TimVTX(a, n, x);

**if** (vt != -1){

XoaTaiVTk(a, n, vt);

}

}

1. Hàm main() gọi thực hiện yêu cầu từ 1 đến 4

**int** main(){

**int** a[100], n, x, vtx;

printf(**"Nhap vao kich thuoc mang: "**);

scanf(**"%d"**, &n);

NhapMang(a, n);

printf(**"\nCac gia tri trong mang: "**);

XuatMang(a, n);

printf(**"\nCac phan tu duong trong mang: "**);

XuatDuong(a, n);

printf(**"\nNhap vao gia tri can tim: "**);

scanf(**"%d"**, &x);

vtx= TimVTX(a, n, x);

**if**(vtx==-1)

printf(**"\nKhong co %d trong mang"**, x);

**else** {

printf(**"\n%d xuat hien tai: %d"**, x, vtx);

XoaX(a, n, x);

printf(**"\nMang sau khi xoa x: "**);

XuatMang(a, n);

}

}

**Phần II. Ma trận hoặc ma trận vuông**

*Các nội dung ôn tập:*

* *Nhập, xuất (nếu là ma trận vuông thì chỉ cần lưu 1 kích thước)*
* *Tìm kiếm: tìm x, tìm max, tìm min*
* *Thao tác trên đường chéo chính (chỉ số dòng = chỉ số cột)*
* *Thao tác trên đường chéo phụ (chỉ số dòng + chỉ số cột = n -1)*
* *Chèn dòng/ cột*
* *Xóa dòng/ cột*

**Bài tập ví dụ:** Cho ma trận vuông số nguyên, kích thước n ***(0<n<=100)***. Hãy cài đặt các hàm sau:

1. Xuất ma trận.

**void** Xuat(int a[100][100], **int** n){

**for**(**int** i=0; i<n; i++){

**for**(**int** j=0; j<n; j++){

printf(**"%d\t"**, a[i][j]);

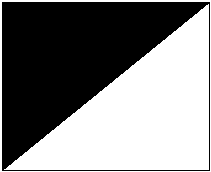
}

printf(**"\n"**);

}

}

1. Xuất các phần tử thuộc tam giác được tô đen như trong hình minh họa.



**void** XuatTrenCheoChinh(int a[100][100], **int** n) {

**for** (**int** i = 0; i < n; i++) {

**for** (**int** j = 0; j < n; j++) {

**if** (j >= i)

printf(**"%d\t"**, a[i][j]);

**else**

printf(**"-\t"**);

}

printf(**"\n"**);

}

}

1. Tính tổng các phần tử có giá trị lẻ.

**int** TongLe(int a[100][100], **int** n) {

**int** s = 0;

**for** (**int** i = 0; i < n; i++) {

**for** (**int** j = 0; j < n; j++) {

**if** (a[i][j] % 2 != 0)

s += a[i][j];

}

}

**return** s;

}

1. Đếm các phần tử lớn hơn giá trị X cho trước.

**int** DemLonHonX(int a[100][100], **int** n, **int** x){

**int** d = 0;

**for** (**int** i = 0; i < n; i++) {

**for** (**int** j = 0; j < n; j++) {

**if** (a[i][j] > x)

d++;

}

}

**return** d;

}

**Phần III. Chuỗi ký tự và kiểu dữ liệu có cấu trúc**

*Các nội dung ôn tập:*

* *Khai báo chuỗi và các hàm xử lý chuỗi cơ bản*
* *Khai báo kiểu dữ liệu có cấu trúc*
* *Truy xuất thành phần bên trong cấu trúc*
* *Xử lý mảng có cấu trúc*
* *Xem lại cú pháp và cách sử dụng một số hàm thư viện xử lý chuỗi ký tự: so sánh chuỗi, tính độ dài chuỗi, tìm kiếm chuỗi con, …*

**Bài tập ví dụ:** Cho danh sách các linh kiện máy tính, thông tin của mỗi linh kiện gồm:

* *Mã linh kiện* ***(maLK)****: chuỗi có 8 ký tự*
* *Tên linh kiện* ***(tenLK)****: chuỗi có tối đa 30 ký tự*
* *Số lượng* ***(soLuong)****: số nguyên*
* *Giá bán* ***(gia)****: số nguyên*

Hãy thực hiện các yêu cầu sau:

1. Khai báo kiểu dữ liệu có cấu trúc biểu diễn thông tin của linh kiện.

**struct** ttLinhKien {

**char** malk[9];

**char** tenLK[31];

**int** soLuong;

**int** gia;

};

**typedef struct** ttLinhKien LinhKien;

1. Cài đặt hàm nhập danh sách linh kiện (tối đa 1000 linh kiện).

**void** NhapLK(LinhKien &lk) {

printf(**"Ma linh kien: "**);

gets(lk.malk);

printf(**"Ten linh kien: "**);

gets(lk.tenLK);

printf(**"So luong: "**);

scanf(**"%d"**, &lk.soLuong);

printf(**"Gia ban: "**);

scanf(**"%d"**, &lk.gia);

flushall();

}

**void** NhapDSLK(LinhKien ds[], **int** n) {

**for** (**int** i = 0; i < n; i++) {

printf(**"\nLinh kien thu %d\n"**, i + 1);

NhapLK(ds[i]);

}

}

1. Cài đặt hàm xuất những linh kiện có số lượng lớn hơn x cho trước.

**void** XuatLK(LinhKien lk) {

printf(**"Ma linh kien: %s\n"**, lk.malk);

printf(**"Ten linh kien: %s\n"**, lk.tenLK);

printf(**"So luong: %d\n"**, lk.soLuong);

printf(**"Gia ban: %d\n"**, lk.gia);

}

**void** XuatDSLK(LinhKien ds[], **int** n, **int** x) {

**for** (**int** i = 0; i < n; i++) {

**if** (ds[i].soLuong > x)

XuatLK(ds[i]);

}

}

1. Cài đặt hàm sắp xếp danh sách linh kiện theo thứ tự giảm dần của tên linh kiện.

**void** HoanVi(LinhKien &a, LinhKien &b) {

LinhKien tam = a;

a = b;

b = tam;

}

**void** SapXep(LinhKien ds[], **int** n) {

**for** (**int** i = 0; i < n - 1; i++) {

**for** (**int** j = i + 1; j < n; j++) {

**if** (stricmp(ds[i].tenLK, ds[j].tenLK) < 0)

HoanVi(ds[i], ds[j]);

}

}

}

**Phần IV. Lập trình đệ quy**

*Các nội dung ôn tập:*

* *Kỹ thuật cài đặt hàm đệ quy*
* *Đệ quy tuyến tính và đệ quy nhị phân*

**Bài tập ví dụ:**

1. Viết hàm nhập vào mảng một chiều số nguyên gồm n phần tử (0<n<50) bằng phương pháp đệ quy.

**void** NhapMang(**int** a[], **int** n) {

**if** (n == 0)

**return**;

NhapMang(a, n - 1);

printf(**"Gia tri tai vi tri %d: "**, n - 1);

scanf(**"%d"**, &a[n - 1]);

}

1. Viết hàm tìm ước số chung lớn nhất của hai số nguyên dương a và b.

**int** USCLN(**int** a, **int** b) {

**if** (a == b)

**return** a;

**if** (a > b)

**return** USCLN(a - b, b);

**return** USCLN(a, b - a);

}

**Phần V. Lập trình xử lý tập tin**

*Các nội dung ôn tập:*

* *Các thao tác trên file*
* *Các hàm xử lý trên file text và file nhị phân*

**Bài tập ví dụ:** Cho file text “***C:\filemau.txt***” gồm có 2 dòng:

* Dòng 1: 2 số nguyên dương **n** và **m** cách nhau bởi khoảng trắng (*0<n,m<=100*)
* Dòng 2: 2 số nguyên dương **min** và **max** cách nhau bởi khoảng trắng (*min<max*)

Hãy viết chương trình tạo đọc file trên và tạo ra một ma trận số nguyên kích thước **n** dòng và **m** cột. Các phần tử trong ma trận có giá trị ngẫu nhiên trong khoảng từ **min** đến **max**. Xuất ma trận vừa được phát sinh ra màn hình.

*Ví dụ:* ***filemau.txt có nội dung***

*4 7*

*50 200*

***Kết quả chương trình***

*90 66 83 149 168 173 127*

*157 161 63 54 144 80 76*

*110 90 194 141 76 85 190*

*103 51 52 191 131 70 165*

**int** DocFile(**char** \*fileName, **int** &n, **int** &m, **int** &min,

**int** &max) {

FILE \*f;

f = fopen(fileName, **"r"**);

**if** (f == **NULL**)

**return** 0;

fscanf(f, **"%d"**, &n);

fscanf(f, **"%d"**, &m);

fscanf(f, **"%d"**, &min);

fscanf(f, **"%d"**, &max);

fclose(f);

**return** 1;

}

**void** PhatSinhMaTran(**int** a[100][100], **int** n, **int** m,

**int** min, **int** max) {

**for** (**int** i = 0; i < n; i++) {

**for** (**int** j = 0; j < m; j++)

a[i][j] = rand() % (max - min) + min - 1;

}

}

**void** XuatMaTran(**int** a[100][100], **int** n, **int** m) {

**for** (**int** i = 0; i < n; i++) {

**for** (**int** j = 0; j < m; j++) {

printf(**"%d\t"**, a[i][j]);

}

printf(**"\n"**);

}

}

**int** main() {

**int** a[100][100], n, m, min, max;

**if** (DocFile(**"C:\\filemau.txt"**, n, m, min, max)==0)

printf(**"Khong doc duoc file"**);

**else** {

PhatSinhMaTran(a, n, m, min, max);

XuatMaTran(a, n, m);

}

getch();

**return** 0;

}