# MẢNG (11)

Câu hỏi và bài tập

1. Viết chương trình nhập vào dãy số với n phần tử, sau đó xuất ra màn hình.

**Bài giải**

#include <stdio.h>

#include <conio.h>

#include <math.h>

void Nhap(int x[100], int n)

{

int i;

for(i=0; i<n; i++)

{

printf("x[%d]=",i);

scanf("%d",&x[i]); #

}

}

void Xuat(int x[100], int n)

{int i;

for(i=0; i<n;i++) printf("%4d",x[i]);

}

main()

{ int n;

int a[100];

printf("Nhap vao day so ");

scanf("%d",&n);

Nhap(a,n);

printf("Day so sau khi nhap");

Xuat(a,n);

getch();

}

1. Viết chương trình nhập tọa độ hai véctơ *n* chiều x, y và tính:
2. véctơ z=x+y
3. b) Tích vô hướng p=xy

**Bài giải**

#include <stdio.h>

#include <conio.h>

#include <math.h>

void Nhap(float x[100], int n, char ten)

{

int i;

for(i=0; i<n; i++)

{

printf("%c[%d]=",ten,i);

scanf("%f",&x[i]);

}

}

void Xuat(float x[100], int n)

{int i;

for(i=0; i<n;i++) printf("%6.2f",x[i]);

}

void Tong(float x[100],float y[200],float z[400], int n)

{ int i;

for(i=0;i<n;i++) z[i]=x[i]+y[i];

}

void Tich(float x[100],float y[200],float z[400], int n)

{ int i;

for(i=0;i<n;i++) z[i]=x[i]\*y[i];

}

main()

{ int n;

float x[100]; float y[200]; float z[400];

printf("Nhap so n phan tu vec to x phai bang phan tu vec to y\n");

printf("Nhap so phan tu n vec to x va y");

scanf("%d",&n);

Nhap(x,n,'X');

Xuat(x,n);

printf("\n");

Nhap(y,n,'Y');

printf("\n");

Xuat(y,n);

printf("\n");

printf("\nTong cua hai vec x va y la\n");

Tong(x,y,z,n);

printf("\n");

Xuat(z,n);

printf("\nTich cua hai vec x va y la\n");

Tich(x,y,z,n);

printf("\n");

Xuat(z,n);

getch();

}

1. Viết chương trình nhập mảng số nguyên A gồm *n* phần tử và đảo ngược thứ tự của chúng.

**Bài giải**

#include <stdio.h>

#include <conio.h>

#include <math.h>

void Nhap(int x[100], int n);

void Xuat(int x[100], int n);

void DaoMang(int x[100], int n);

main()

{ int n;

int a[100];

printf("Nhap vao day so ");

scanf("%d",&n);

Nhap(a,n);

printf("Day so sau khi nhap");

Xuat(a,n);

printf("\n");

printf("Mang sau khi dao");

DaoMang(a,n);

Xuat(a,n);

getch();

}

void Nhap(int x[100], int n)

{

int i;

for(i=0; i<n; i++)

{

printf("x[%d]=",i);

scanf("%d",&x[i]);

}

}

void Xuat(int x[100], int n)

{int i;

for(i=0; i<n;i++) printf("%4d",x[i]);

}

void DaoMang(int x[100], int n)

{ int tam,i;

for(i=0; i<n/2;i++)

{tam =x[i];

x[i]=x[n-i-1];

x[n-i-1]=tam;

}

}

1. Viết chương trình nhập ma trận thực vuông cấp n, rồi sắp xếp tất cả các phần tử của các cột theo thứ tự giảm dần.

**Bài giải**

1. Viết chương trình nhập ma trận nguyên A là ma trận vuông cấp n, rồi tìm phần tử có giá trị lớn nhất trong đường chéo chính của ma trận A.

**Bài giải**

1. Viết chương trình nhập mảng số nguyên A gồm *n* phần tử. Tìm phần tử có giá trị lớn nhất trong mảng A.

**Bài giải**

#include <stdio.h>

#include <conio.h>

#include <math.h>

void Nhap(int a[100], int n);

int PhanTulonNhat(int a[100], int n);

void Xuat(int a[100], int n);

main()

{ int n;

int a[100];

printf("Nhap mang a ");

scanf("%d",&n);

Nhap(a,n);

printf("Mang a sau khi nhap ");

Xuat(a,n);

printf("\n");

printf("Phan tu lon nhat trong mang: %d",PhanTulonNhat(a,n));

getch();

}

void Nhap(int a[100], int n)

{

int i;

for(i=0; i<n; i++)

{

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

scanf("%d",&a[i]);

}

}

int PhanTulonNhat(int a[100], int n)

{ int i,max;

max=a[0];

for(i=0; i<n; i++)

if (a[i]>max) max=a[i];

return max;

}

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

{int i;

for(i=0; i<n;i++) printf("%4d",a[i]);

}

1. Cho biết hàm sau làm gì ?

void in\_ma\_tran (int a[][Max\_Cot], int m, int n);

{

int i, j;

for (i=0; i < m; i++)

for (j=0; j < n; ++j)

printf("%4d%c", a[i][j], j == n-1 ? '\n' : ' ');

}

1. Viết chương trình nhập ma trận A là ma trận nguyên vuông cấp *n*, rồi kiểm tra ma trận A có đối xứng qua đường chéo chính không?

**Bài giải**

#include <stdio.h>

#include <conio.h>

#include <math.h>

void Nhap(int a[][100], int n);

void Xuat(int a[][100],int n);

int KiemTraDoiXung(int a[][100], int n);

main()

{int a[100][100];

int n;

printf("Nhap n");

scanf("%d",&n);

Nhap(a,n);

Xuat(a,n);

printf("\n");

if(KiemTraDoiXung(a,n)) printf("Ma tran doi xung qua duong cheo chinh");

else printf("Ma tran khong doi xung");

getch();

}

void Nhap(int a[][100], int n)

{ int i,j;

for(i=0;i<n; i++)

for(j=0;j<n; j++)

{

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

scanf("%d",&a[i][j]);

}

}

int KiemTraDoiXung(int a[][100], int n)

{int i,j;

for(i=0;i<n;i++)

for(j=0;j<n;j++)

if(a[i][j]!=a[j][i]) return 0;

return 1;

}

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

{int i,j;

for(i=0;i<n;i++)

{

printf("\n");

for(j=0;j<n;j++) printf("%4d",a[i][j]);

}

}

1. Ma phương bậc *n* là ma trận vuông cấp *n* gồm *n2* số từ 1 đến *n2* được sắp xếp thành *n* hàng và *n* cột sao cho tổng hàng, tổng cột và tổng đường chéo bằng nhau. Các bước sinh ma phương với *n* lẻ như sau: số 1 đặt ở giữa hàng đầu (hàng 1). Khi số *k* đặt ở ô (i,j) thì *k+1* đặt tại ô (i-1,j+1); nếu i-1=0, ra ngoài ma trận thì đặt tại hàng cuối (hàng *n*); nếu j+1=n+1, ra ngoài ma trận thì đặt tại cột đầu (cột 1). Viết chương trình nhập *n* lẻ và in ra màn hình ma phương bậc *n*.
2. Viết chương trình nhập mảng số nguyên A gồm *n* phần tử và nhập số nguyên x. Xóa tất cả các phần tử có giá trị *x* ra khỏi mảng A.
3. Viết chương trình nhập mảng số thực A gồm *n* phần tử. Xóa tất cả các phần tử có giá trị lớn nhất ra khỏi mảng A
4. Viết chương trình nhập mảng số nguyên A gồm *n* phần tử. Đếm số phần tử có giá trị lớn nhất trong mảng A.
5. Viết chương trình nhập mảng số nguyên A gồm *n* phần tử và nhập số nguyên *k.* Xóa phần tử có chỉ số *k* ra khỏi mảng A.
6. Viết hàm Merge(X,Y,Z) với hai mảng số nguyên X,Y đã sắp xếp tăng dần. Tạo mảng Z tăng dần bằng cách trộn hai mảng X, Y lại.   
   Ví dụ, X=[5, 8], Y=[2, 3, 7, 9] thì Z=[2, 3, 5, 7, 8, 9].

**Bài giải**

#include <stdio.h>

#include <conio.h>

#include <math.h>

void Nhap(int x[100], int n, char ten)

{

int i;

for(i=0; i<n; i++)

{

printf("%c[%d]=",ten,i);

scanf("%d",&x[i]);

}

}

void Xuat(int x[100], int n)

{int i;

for(i=0; i<n;i++) printf("%4d",x[i]);

}

void SapXep(int x[100],int n)

{

int temp=0,i,j;

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

{

for(j=i+1;j<n;j++)

{

if(x[i]>x[j]){

temp=x[i];

x[i]=x[j];

x[j]=temp;

}

}

}

}

void NoiMang(int x[ ],int y[ ],int z[], int n, int m)

{

int t=n+m;

int k=0,i,j;

for(i=0;i<n;i++)

{z[k]=x[i];

k++;

}

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

{z[k]=y[j];

k++;

}

SapXep(z,t);

}

main()

{ int t,m,n;

int x[100]; int y[200]; int z[400];

printf("Nhap so phan tu n");

scanf("%d",&n);

printf("Nhap so phan tu m");

scanf("%d",&m);

t=n+m;

Nhap(x,n,'X');

printf("Mang X sau khi sap xep tang dan\n");

SapXep(x,n);

Xuat(x,n);

printf("\n");

Nhap(y,m,'Y');

printf("\n");

printf("Mang Y sau khi sap xep tang dan");

SapXep(y,m);

Xuat(y,m);

printf("Mang Z sau khi noi co ket qua la");

NoiMang(x,y,z,n,m);

Xuat(z,t);

getch();

}

1. Viết chương trình nhập ma trận A là ma trận nguyên vuông cấp *n* và nhập số nguyên k. Xóa cột thứ *k* ra khỏi ma trận A.
2. Phương pháp sàng *Eratosthenes* để tìm tập T gồm các số nguyên tố trong khoảng từ 2..MAX có thể mô tả bằng giả lệnh như sau:   
    S={2..MAX}; n=2;   
    While (T=Ø){  
    if (n ){  
    T=T;  
    Loại tất cả các bội n ra khỏi S;  
    }  
    n++;  
    }  
   Viết chương trình in ra tất cả các số nguyên tố trong 2..MAX với MAX=1000.

**Bài giải : Cách giải khác**

Thuật toán

Dùng thuật toán sàng Eratosthène để xác định dãy các số nguyên tố nhỏ hơn 1000.

Thuật toán này xác định các số nguyên tố trong khoảng từ 1 đến N.

Xét dãy số i từ 2 đến SQRT(N) và dãy số từ 2 den N/i.

Bất kỳ số nào là tích của một số thuộc dãy này và một số thuộc dãy kia đều có giá trị nằm trong khoảng từ 2 đến N, và một khi đã là tích của 2 số thì không thể là số nguyên tố. Loại trừ các số này đi thì  còn lại là các số nguyên tố trong khoảng từ 1 đến N.

**#**include <stdio.h>

#include <conio.h>

#define MAX 1000

int main()

{

int A[2000];

int i,j;

for (i = 0; i <= MAX; i++) A[i] = 1;

for (i = 2; i <= MAX/2; i++)

for ( j= 2; j <= MAX/i; j++)

A[i\*j] = 0;

for (i = 2; i < MAX; i++) if (A[i]) printf("%4d", i);

getch();

}

CHUỖI KÝ TỰ (12)

Câu hỏi và bài tập

1. Viết chương trình nhập chuỗi ký tự s gồm các chữ cái. Đổi tất cả các chữ cái trong s sang chữ hoa.
2. Viết chương trình nhập chuỗi ký tự s gồm các chữ cái. Đếm số từ trong s.
3. Viết chương trình nhập chuỗi ký tự s gồm các chữ cái. Đếm số từ khác nhau trong s.
4. Viết chương trình nhập chuỗi ký tự s gồm các chữ cái. In ra bảng tần số xuất hiện của các từ khác nhau trong s.  
   Ví dụ : s= "DAI HOC BACH KHOA, DAI HOC DA NANG " thì bảng tần số là:  
   DAI  : 2  
   HOC  : 2  
   BACH  : 1  
   ...
5. Viết chương trình nhập chuỗi ký tự s gồm các chữ cái. In ra bảng tần số xuất hiện của các chữ cái khác nhau trong s.  
   Ví dụ : s= "DAI HOC BACH KHOA, DAI HOC DA NANG " thì bảng tần số là:  
   A : 6  
   B : 1  
   C : 3  
   ...
6. Một chuỗi ký tự gọi là *palindrome* nếu nó tự đối xứng. Ví dụ: "ABCBA",   
   'lewd I did live , evil did I dwel'.   
   Viết hàm *palind(s)* để kiểm tra chuỗi ký tự s có phải *palindrome* hay không.
7. Giả sử chuỗi ký tự *s* chỉ gồm các chữ số. Viết hàm *atoi(s)* để đổi *s* sang số nguyên. Ví dụ, s="0123" thì trả về số nguyên 123.
8. Viết hàm *Trim(s)* để xóa tất cả các ký tự trống (dấu cách) ở hai đầu của *s*.
9. Viết chương trình nhập một chuỗi ký tự và thay một hoặc nhiều dấu cách bởi một dấu cách đơn.
10. Viết hàm *Proper (s)* để đổi chuỗi ký tự *s* sang dạng danh từ riêng. Nghĩa là đầu mỗi từ là chữ hoa, còn lại là chữ thường.  
    Ví dụ, s="DAI hoC baCH KhOA" sau khi gọi hàm s="Dai Hoc Bach Khoa".
11. Viết hàm itob(n,s,b) để đổi số nguyên dương n sang cơ số b và lưu kết quả trong chuỗi ký tự s.
12. Viết hàm strindex(s,t) để tìm vị trí cuối cùng (bên phải nhất) của t trong s. Trả về -1 nếu không có.
13. Viết hàm đệ quy *reverse(s)* để đảo chuỗi ký tự s.
14. Viết hàm *strend(s, t)* để trả về 1 nếu chuỗi *t* ở cuối chuỗi *s*; ngược lại trả về 0.
15. Viết hàm *squeeze(s, c)* để xóa tất cả ký tự *c* trong chuỗi ký tự *s*.
16. Viết hàm *month\_name(n)* để trả về tên của tháng *n*.   
    Ví dụ, month\_name(7)= "July"