Chương 3:

Kỹ Thuật Lập Trình mảng, Chuỗi Ký Tự

Nội dung

Đặt vấn đề

- Ví dụ
  - Chương trình cần lưu trữ 3 số nguyên?
    => Khai báo 3 biến int a1, a2, a3;
  - Chương trình cần lưu trữ 100 số nguyên?
    => Khai báo 100 biến kiểu số nguyên!
  - Người dùng muốn nhập n số nguyên?
    => Không thực hiện được!
- Giải pháp
  - Kiểu dữ liệu mới cho phép lưu trữ một dãy các số nguyên và dễ dàng truy xuất.

Dữ liệu kiểu mảng

- Khái niệm
  - Là một kiểu dữ liệu có cấu trúc do người lập trình định nghĩa.
  - Biểu diễn một dãy các biến có cùng kiểu.

Ví dụ: dãy các số nguyên, dãy các ký tự…

  - Kích thước được xác định ngay khi khai báo và không bao giờ thay đổi.
  - NNLT C luôn chỉ định một khối nhớ liên tục cho một biến kiểu mảng.

Khai báo biến mảng

- Cú pháp:

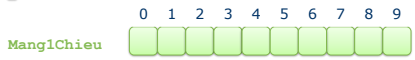
```
<kiểu cơ sở> <tên biến mảng>[<số phần tử>];
<kiểu cơ sở> <tên biến mảng>[<N1>][<N2>]…[<Nn>];
```

  - <N1>, …, <Nn> : số lượng phần tử của mỗi chiều.
- Lưu ý
  - Phải xác định <số phần tử> cụ thể (hằng) khi khai báo.
  - Mảng nhiều chiều: <tổng số phần tử> = N1*N2*…*Nn
  - Bộ nhớ sử dụng = <tổng số phần tử>*sizeof(<kiểu cơ sở>)
  - Bộ nhớ sử dụng phải ít hơn 64KB (65535 Bytes)
  - Một dãy liên tục có chỉ số từ 0 đến <tổng số phần tử>-1

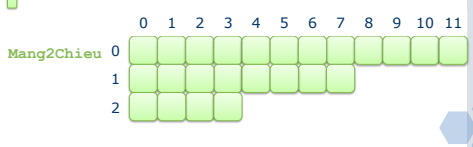
Khai báo biến mảng

- Ví dụ

```
int Mang1Chieu[10];
```




```
int Mang2Chieu[3][4];
```



Số phần tử của mảng

- Phải xác định cụ thể số phần tử ngay lúc khai báo, không được sử dụng biến hoặc hằng thường

```
int n1 = 10; int a[n1];
const int n2 = 20; int b[n2];
```

- Nên sử dụng chỉ thị tiền xử lý #define để định nghĩa số phần tử mảng

```
#define n1 10
#define n2 20
int a[n1];          // ⇔ int a[10];
int b[n1][n2];      // ⇔ int b[10][20];
```

7

Khởi tạo giá trị cho mảng lúc khai báo

- Gồm các cách sau
  - Khởi tạo giá trị cho mọi phần tử của mảng

```
int a[4] = {2912, 1706, 1506, 1904};
```

	0	1	2	3
a	2912	1706	1506	1904

  - Khởi tạo giá trị cho một số phần tử đầu mảng

```
int a[4] = {2912, 1706};
```

	0	1	2	3
a	2912	1706		

8

Khởi tạo giá trị cho mảng lúc khai báo

- Gồm các cách sau
  - Khởi tạo giá trị 0 cho mọi phần tử của mảng

```
int a[4] = {0};
```

	0	1	2	3
a	0	0	0	0

  - Tự động xác định số lượng phần tử

```
int a[] = {2912, 1706, 1506, 1904};
```

	0	1	2	3
a	2912	1706	1506	1904

9

Truy xuất đến một phần tử

- Thông qua chỉ số

```
<tên biến mảng>[<gt cs1>][<gt cs2>]…[<gt csn>]
```

- Ví dụ
  - Cho mảng như sau

```
int a[4];
```

	0	1	2	3

  - Các truy xuất
    - Hợp lệ: a[0], a[1], a[2], a[3]
    - Không hợp lệ: a[-1], a[4], a[5], …
      - ➔ Cho kết quả thường không như mong muốn!

10

Lấy địa chỉ của phần tử mảng

- Sử dụng toán tử &

```
&<tên biến mảng>[i]   (i là chỉ số của mảng)
```

- Chú ý:
  - Tên của mảng chứa địa chỉ đầu của mảng.
  - Ví dụ có khai báo

    int a[10] ;

    thì a = &a[0]

11

Gán dữ liệu kiểu mảng

- Không được sử dụng phép gán thông thường mà phải gán trực tiếp giữa các phần tử tương ứng

```
<biến mảng đích> = <biến mảng nguồn>; //sai
<biến mảng đích>[<chỉ số thứ i>] := <giá trị>;
```

- Ví dụ

```
#define MAX 3
typedef int MangSo[MAX];
MangSo a = {1, 2, 3}, b;

b = a;      // Sai
for (int i = 0; i < 3; i++) b[i] = a[i];
```

12

Một số lỗi thường gặp

- Khai báo không chỉ rõ số lượng phần tử
  - int a[]; => int a[100];
- Số lượng phần tử liên quan đến biến hoặc hằng
  - int n1 = 10; int a[n1]; => int a[10];
  - const int n2 = 10; int a[n2]; => int a[10];
- Khởi tạo cách biệt với khai báo
  - int a[4]; a = {2912, 1706, 1506, 1904};
    => int a[4] = {2912, 1706, 1506, 1904};
- Chỉ số mảng không hợp lệ
  - int a[4];
  - a[-1] = 1; a[10] = 0;

13

Ví dụ

Viết chương trình thực hiện các yêu cầu sau.

- **Nhập số nguyên dương n thỏa mãn 1≤n≤20.**
- **Nhập mảng có n số thực.**
- **Hiển thị mảng vừa nhập ra màn hình.**
- **Thống kê và hiển thị ra màn hình các số có giá trị âm trong mảng và giá trị trung bình cộng của các số đó.**

14

Ví dụ

```
#include <iostream>
using namespace std;
void nhapmang(float a[20], int n)
{    for (int i=0; i<n; i++)
   {
    cout<<"a["<<i<<"]=";
          cin>>a[i];
   }
}
void hienmang(float a[20], int n)
{ for (int i=0; i<n; i++)
          cout<<a[i]<<" ";
   cout<<endl;
}
```

15

Ví dụ

```
void thongke(float a[20], int n)
{ float t=0;
  int dem=0;
  cout<<"Cac so am trong mang: ";
  for (int i=0; i<n; i++)
    if (a[i]<0)
          {     cout<<a[i]<<" ";
                t +=a[i];
                dem++;
          }
   if (dem == 0)
         cout<<"Khong co so am";
   else{
          float tb = t/dem;
          cout<<"\nTBC cac so am la "<<tb;
       }
}
```

16

Ví dụ

```
int main()
{
   float a[20],max;
   int n;
   do{
          cout<<"Nhap so phan tu (1<=n<=20): ";
          cin>>n;
   }while(n<1||n>20);
   cout<<"Nhap mang "<<n<<" so thuc\n";
   nhapmang(a,n); //gọi hàm nhập mảng
   cout<<"Mang vua nhap: ";
   hienmang(a,n);//gọi hàm hiển thị mảng
   thongke(a,n); //gọi hàm thống kê

}
```

17

Truyền mảng cho hàm

- Truyền mảng cho hàm
  - Tham số kiểu mảng trong khai báo hàm giống như khai báo biến mảng

```
void SapXepTang(int a[100]);
```

  - Tham số kiểu mảng truyền cho hàm chính là địa chỉ của phần tử đầu tiên của mảng
    - Có thể bỏ số lượng phần tử hoặc sử dụng con trỏ.
    - Mảng có thể thay đổi nội dung sau khi thực hiện hàm.

```
void SapXepTang(int a[]);
void SapXepTang(int *a);
```

18

Truyền mảng cho hàm

- Truyền mảng cho hàm
  - Số lượng phần tử thực sự truyền qua biến khác

```
void SapXepTang(int a[100], int n);
void SapXepTang(int a[], int n);
void SapXepTang(int *a, int n);
```

- Lời gọi hàm

```
void NhapMang(int a[], int &n);
void XuatMang(int a[], int n);
void main()
{
	int a[100], n;
	NhapMang(a, n);
	XuatMang(a, n);
}
```


Một số bài toán cơ bản

- Viết hàm thực hiện từng yêu cầu sau
  - Nhập mảng
  - Xuất mảng
  - Tìm kiếm một phần tử trong mảng
  - Kiểm tra tính chất của mảng
  - Tách mảng / Gộp mảng
  - Tìm giá trị nhỏ nhất/lớn nhất của mảng
  - Sắp xếp mảng giảm dần/tăng dần
  - Thêm/Xóa/Sửa một phần tử vào mảng

Một số quy ước

- Số lượng phần tử

```
#define MAX 100
```

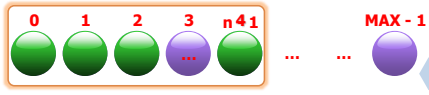
- Các hàm
  - Hàm void HoanVi(int &x, int &y): hoán vị giá trị của hai số nguyên.
  - Hàm int LaSNT(int n): kiểm tra một số có phải là số nguyên tố. Trả về 1 nếu n là số nguyên tố, ngược lại trả về 0.

Hàm HoanVi & Hàm LaSNT

```
void HoanVi(int &x, int &y)
{
	int tam = x; x = y; y = tam;
}

int LaSNT(int n)
{
	int i, dem = 0;
	for (i = 1; i <= n; i++)
		if (n%i == 0)
			dem++;

	if (dem == 2)
		return 1;
	else return 0;
}
```


Nhập mảng

- Yêu cầu
  - Cho phép nhập mảng a, số lượng phần tử n
- Ý tưởng
  - Cho trước một mảng có số lượng phần tử là MAX.
  - Nhập số lượng phần tử thực sự n của mảng.
  - Nhập từng phần tử cho mảng từ chỉ số 0 đến n – 1.



Hàm Nhập Mảng

```
void NhapMang(int a[], int &n)
{
	cout<<"Nhap so luong phan tu n:";
	cin>>n;

	for (int i = 0; i < n; i++)
	{
		cout<<"Nhap phan tu thu: "<<i;
		cin>>a[i];
	}
}
```

Xuất mảng

- Yêu cầu
  - Cho trước mảng a, số lượng phần tử n. Hãy xuất nội dung mảng a ra màn hình.
- Ý tưởng
  - Xuất giá trị từng phần tử của mảng từ chỉ số 1 đến n.

25

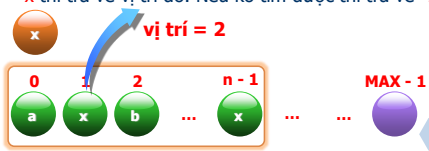
Hàm Xuất Mảng

```
void XuatMang(int a[], int n)
{
	cout<<"Noi dung cua mang la: ";

	for (int i = 0; i < n; i++)
		cout<<"   "<<a[i];

	cout<<'\n';
}
```

26

Tìm kiếm một phần tử trong mảng

- Yêu cầu
  - Tìm xem phần tử x có nằm trong mảng a kích thước n hay không? Nếu có thì nó nằm ở vị trí đầu tiên nào.
- Ý tưởng
  - Xét từng phần của mảng a. Nếu phần tử đang xét bằng x thì trả về vị trí đó. Nếu ko tìm được thì trả về -1.



27

Hàm Tìm Kiếm (dùng while)

```
int TimKiem(int a[], int n, int x)
{
	int vt = 0;

	while (vt < n && a[vt] != x)
		vt++;

	if (vt < n)
		return vt+1;
	else
		return -1;
}
```

28

Hàm Tìm Kiếm (dùng for)

```
int TimKiem(int a[], int n, int x)
{
	for (int vt = 0; vt < n; vt++)
		if (a[vt] == x)
			return vt + 1;

	return -1;
}
```

29

Kiểm tra tính chất của mảng

- Yêu cầu
  - Cho trước mảng a, số lượng phần tử n. Mảng a có phải là mảng toàn các số nguyên tố hay không?
- Ý tưởng
  - Cách 1: Đếm số lượng số ngtố của mảng. Nếu số lượng này bằng đúng n thì mảng toàn ngtố.
  - Cách 2: Đếm số lượng số không phải ngtố của mảng. Nếu số lượng này bằng 0 thì mảng toàn ngtố.
  - Cách 3: Tìm xem có phần tử nào không phải số ngtố không. Nếu có thì mảng không toàn số ngtố.

30

Hàm Kiểm Tra (Cách 1)

```
int KiemTra_C1(int a[], int n)
{
	int dem = 0;

	for (int i = 0; i < n; i++)
		if (LaSNT(a[i]) == 1) // có thể bỏ == 1
			dem++;

	if (dem == n)
		return 1;
	return 0;
}
```

31

Hàm Kiểm Tra (Cách 2)

```
int KiemTra_C2(int a[], int n)
{
	int dem = 0;

	for (int i = 0; i < n; i++)
		if (LaSNT(a[i]) == 0) // Có thể sử dụng !
			dem++;

	if (dem == 0)
		return 1;
	return 0;
}
```

32

Hàm Kiểm Tra (Cách 3)

```
int KiemTra_C3(int a[], int n)
{
	for (int i = 0; i < n ; i++)
		if (LaSNT(a[i]) == 0)
			return 0;

	return 1;
}
```

33

Tách các phần tử thỏa điều kiện

- Yêu cầu
  - Cho trước mảng a, số lượng phần tử na. Tách các số nguyên tố có trong mảng a vào mảng b.
- Ý tưởng
  - Duyệt từ phần tử của mảng a, nếu đó là số nguyên tố thì đưa vào mảng b.

34

Hàm Tách Số Nguyên Tố

```
void TachSNT(int a[], int na, int b[], int &nb)
{
	nb = 0;

	for (int i = 0; i < na; i++)
		if (LaSNT(a[i]) == 1)
		{
			b[nb] = a[i];
			nb++;
		}
}
```

35

Tách mảng thành 2 mảng con

- Yêu cầu
  - Cho trước mảng a, số lượng phần tử na. Tách mảng a thành 2 mảng b (chứa số nguyên tố) và mảng c (các số còn lại).
- Ý tưởng
  - Cách 1: viết 1 hàm tách các số nguyên tố từ mảng a sang mảng b và 1 hàm tách các số không phải nguyên tố từ mảng a sang mảng c.
  - Cách 2: Duyệt từ phần tử của mảng a, nếu đó là số nguyên tố thì đưa vào mảng b, ngược lại đưa vào mảng c.

36

Hàm Tách 2 Mảng

```
void TachSNT2(int a[], int na,
              int b[], int &nb, int c[], int &nc)
{
      nb = 0;
      nc = 0;

      for (int i = 0; i < na; i++)
            if (LaSNT(a[i]) == 1)
            {
                  b[nb] = a[i]; nb++;
            }
            else
            {
                  c[nc] = a[i]; nc++;
            }
}
```


Gộp 2 mảng thành một mảng

- Yêu cầu
  - Cho trước mảng a, số lượng phần tử na và mảng b số lượng phần tử nb. Gộp 2 mảng trên theo tứ tự đó thành mảng c, số lượng phần tử nc.
- Ý tưởng
  - Chuyển các phần tử của mảng a sang mảng c
    => nc = na
  - Tiếp tục đưa các phần tử của mảng b sang mảng c
    => nc = nc + nb

Hàm Gộp Mảng

```
void GopMang(int a[], int na, int b[], int nb,
             int c[], int &nc)
{
      nc = 0;

      for (int i = 0; i < na; i++)
      {
            c[nc] = a[i]; nc++; // c[nc++] = a[i];
      }

      for (int i = 0; i < nb; i++)
      {
            c[nc] = b[i]; nc++; // c[nc++] = b[i];
      }
}
```


Tìm giá trị lớn nhất của mảng

- Yêu cầu
  - Cho trước mảng a có n phần tử. Tìm giá trị lớn nhất trong a (gọi là max)
- Ý tưởng
  - Giả sử giá trị max hiện tại là giá trị phần tử đầu tiên a[0]
  - Lần lượt kiểm tra các phần tử còn lại để cập nhật max.

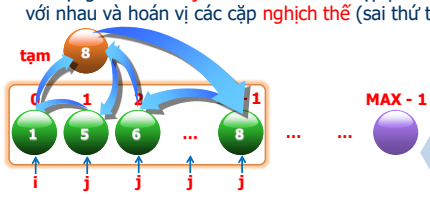
Hàm tìm Max

```
int TimMax(int a[], int n)
{
      int max = a[0];

      for (int i = 1; i < n; i++)
            if (a[i] > max)
                  max = a[i];

      return max;
}
```

Yêu cầu đếm xem trong mảng a có bao nhiêu phần tử bằng max

Sắp xếp mảng thành tăng dần

- Yêu cầu
  - Cho trước mảng a kích thước n. Hãy sắp xếp mảng a đó sao cho các phần tử có giá trị tăng dần.
- Ý tưởng
  - Sử dụng 2 biến i và j để so sánh tất cả cặp phần tử với nhau và hoán vị các cặp nghịch thế (sai thứ tự).



Hàm Sắp Xếp Tăng

```
void SapXepTang(int a[], int n)
{
	int i, j;

	for (i = 0; i < n - 1; i++)
	{
		for (j = i + 1; j < n; j++)
		{
			if (a[i] > a[j])
				HoanVi(a[i], a[j]);
		}
	}
}
```

43

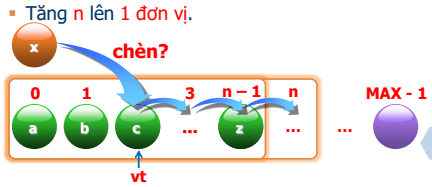
Hàm Sắp Xếp Tăng

```
void sort(int X[], int n)
{
   for (int i=0; i<n-1; i++)
   {
      int m=i;
      for (int j=i+1; j<n; j++)
         if (X[j]<X[m])
            m=j; //kết thúc for 2
      if(m!=i)
            {int tg=X[m];
            X[m]=X[i];
            X[i]=tg;
            }
   }
}
```

44

Thêm một phần tử vào mảng

- Yêu cầu
  - Thêm phần tử x vào mảng a kích thước n tại vị trí vt.
- Ý tưởng
  - "Đẩy" các phần tử bắt đầu tại vị trí vt sang phải 1 vị trí.
  - Đưa x vào vị trí vt trong mảng.
  - Tăng n lên 1 đơn vị.



45

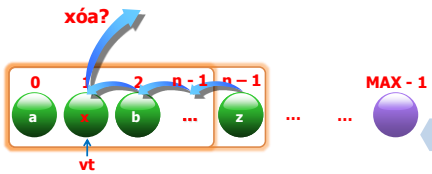
Hàm Thêm

```
void Them(int a[], int &n, int vt, int x)
{
	if (vt >= 0 && vt <= n)
	{
		for (int i = n; i > vt; i--)
			a[i] = a[i - 1];

		a[vt] = x;
		n++;
	}
}
```

46

Xóa một phần tử trong mảng

- Yêu cầu
  - Xóa một phần tử trong mảng a kích thước n tại vị trí vt
- Ý tưởng
  - "Kéo" các phần tử bên phải vị trí vt sang trái 1 vị trí.
  - Giảm n xuống 1 đơn vị.



47

Hàm Xóa

```
void Xoa(int a[], int &n, int vt)
{	if (vt >= 0 && vt < n)
	{
		for (int i = vt; i < n - 1; i++)
			a[i] = a[i + 1];
		n--;
	}
}
//xóa phần tử đầu tiên bằng max
//xác định vt
	duyệt mảng a
		nếu a[i]==max
			vt=i+1
//xóa tất cả các phần tử bằng max
	Duyệt mảng
		nếu a[i]==max
			gọi hàm Xoa(a,n,i)
```

48

Bài tập thực hành

4. Các thao tác nhập xuất
   a. Nhập mảng
   b. Xuất mảng
5. Các thao tác kiểm tra
   a. Mảng có phải là mảng toàn chẵn
   b. Mảng có phải là mảng toàn số nguyên tố
   c. Mảng có phải là mảng tăng dần

Bài tập thực hành

6. Các thao tác tính toán
   a. Có bao nhiêu số chia hết cho 4 nhưng không chia hết cho 5
   b. Tổng các số nguyên tố có trong mảng
7. Các thao tác tìm kiếm
   a. Vị trí cuối cùng của phần tử x trong mảng
   b. Vị trí số nguyên tố đầu tiên trong mảng nếu có
   c. Tìm số nhỏ nhất trong mảng
   d. Tìm số dương nhỏ nhất trong mảng

Bài tập thực hành

8. Các thao tác xử lý
   a. Tách các số nguyên tố có trong mảng a đưa vào mảng b.
   b. Tách mảng a thành 2 mảng b (chứa các số nguyên dương) và c (chứa các số còn lại)
   c. Sắp xếp mảng giảm dần
   d. Sắp xếp mảng sao cho các số dương đứng đầu mảng giảm dần, kế đến là các số âm tăng dần cuối cùng là các số 0.

Bài tập thực hành

9. Các thao tác thêm/xóa/sửa
   a. Sửa các số nguyên tố có trong mảng thành số 0
   b. Chèn x vào đầu mảng
   c. Chèn x vào cuối
   d. Chèn số x vào mảng tăng dần sao cho sau khi chèn mảng vẫn còn thứ tự
   e. Chèn số 0 đằng sau các số nguyên tố trong mảng
   f. Xóa đầu/cuối
   g. Xóa tất cả số nguyên tố có trong mảng

Mảng 2 chiều

- Nhập, xuất
- Tính tổng, tích
- Tìm kiếm
- Đếm
- Sắp xếp
- Thêm, xoá, thay thế

Khai báo

❖ Cách 1: Con trỏ hằng

<Kiểu dữ liệu> <Tên mảng> [<Số dòng>][< Số cột];

❖ 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 xuất phần tử của mảng

- Để truy xuất các thành phần của mảng hai chiều ta phải dựa vào chỉ số dòng và chỉ số cột.
- Ví dụ:

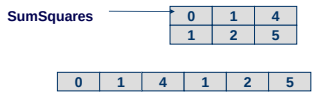
int A[3][4] = { {2,3,9,4} , {5,6,7,6} , {2,9,4,7} };

Với các khai báo như trên ta có :

A[0][0] = 2; A[0][1] = 3;
A[1][1] = 6; A[1][3] = 6;

Mảng nhiều chiều

- Ví dụ



- int SumSquares[2][3] = { {0,1,4}, {1,2,5} };
- int SumSquares[2][3] = { 0,1,4,1,2,5 };
- int SumSquares[2][3] = { {0,1,4} };
- int SumSquares[ ][3] = { {0,1,4}, {1,2,5} };

Nhập/xuất Mảng 2 chiều

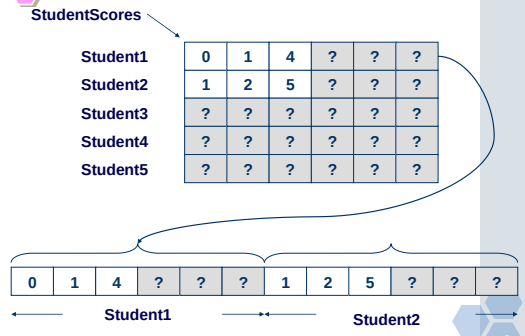
1. Nhập phần tử mảng

```
cout<<"Nhập số hàng của ma trận: "; cin>>m;
cout<<"Nhập số cột của ma trận: "; cin>>n;
for(int i=0; i<m; i++)
  for(int j=0; j<n; j++)
  {cout<<"b["<<i<<"]= "; cin>>b[i][j];}
```

2. Xuất phần tử mảng

```
for(int i=0;i<m;i++)
   {  for(int j=0;j<n;j++)
            cout<<a[i][j]<<"     ";
    cout<<endl;
   }
```


Biểu diễn mảng 2 chiều



Nhập/xuất Mảng 2 chiều

- Ví dụ: Nhập, in và tìm phần tử lớn nhất của một ma trận.

```
#include <iostream>
#include <iomanip.h>
#include <conio.h>
int main()
{   float a[10][10] ;
    int m, n ;                  // số hàng, cột của ma trận
    int i, j ;                  // các chỉ số trong vòng lặp
    int amax, imax, jmax ;      // số lớn nhất và chỉ số của nó
    cout << "Nhập số hàng và cột: " ; cin >> m >> n ;
    for (i=0; i<m; i++)
        for (j=0; j<n; j++)
       {
            cout << "a[" << i << "," << j << "] = " ;
            cin >> a[i][j] ;
        }
```


Nhập/xuất Mảng 2 chiều

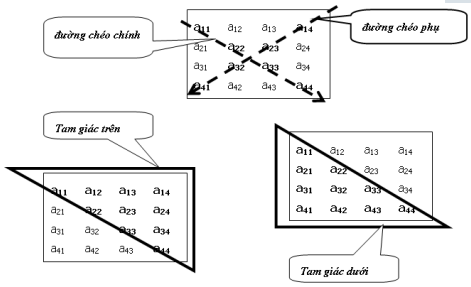
```
amax = a[0][0]; imax = 0; jmax = 0;
for (i=0; i<m; i++)
   for (j=0; j<n; j++)
     if (amax < a[i][j])
       { amax = a[i][j]; imax = i; jmax = j; }
cout << "Ma trận đã nhập\n" ;
cout << setiosflags(ios::showpoint) << setprecision(1) ;
for (i=0; i<m; i++)
    for (j=0; j<n; j++)
       {        if (j==0) cout << endl;
                cout << setw(6) << a[i][j] ;
        }
cout << "Số lớn nhất là " << setw(6) << amax << endl;
cout << "tại vị trí (" << imax << "," << jmax << ")" ;
}
```

Ma trận vuông và các khái niệm liên quan

- Đường chéo chính
- Đường chéo phụ
- Tam giác trên
- Tam giác dưới

61

Ma trận vuông và các khái niệm liên quan



62

Một số bài toán cơ bản

- Viết hàm thực hiện từng yêu cầu sau
  - Nhập mảng
  - Xuất mảng
  - Xuất đường chéo chính/phụ/tam giác trên/tam giác dưới
  - Tính tổng
  - Đếm
  - Tìm kiếm một phần tử trong mảng
  - Tìm giá trị nhỏ nhất/lớn nhất của mảng
  - Kiểm tra tính chất của mảng

63

Một số bài toán cơ bản

- Nhập, xuất: Viết các hàm
  - Nhập ma trận kích thước nxm các số nguyên dương (3<n, m<10)
  - Tạo ma trận kích thước nxm (3<n, m<10) các số nguyên dương có giá trị ngẫu nhiên trong khoảng -10 đến 20.
  - Xuất ma trận ra màn hình.
  - Viết hàm tính tổng các phần tử chẵn trong ma trận
  - Viết hàm tính tổng các phần tử trên cùng một dòng.

64

Kỹ Thuật Lập Trình

KÝ TỰ VÀ CHUỖI KÝ TỰ

65

Ký tự (character)

- Kiểu char:
  - ký tự "in được" gồm: 26 chữ thường (a..z), 26 chữ hoa (A..Z), 10 chữ số (0..9), khoảng trắng, các ký tự: ! " # $ % & ' ( ) * + , - . / : ; < = > ? @ [\ ] ^ _ { | } ~
  - các ký tự "không in được": tab, lert (bell), newline, formfeed,...
- các ký tự "in được" đặc biệt: '\\', '\"', '\'"
- các ký tự "không in được" đặc biệt:

\n	new line
\a	bell
\0	null character
\b	backspace
\t	horizontal tab
...	

66

Nhập xuất Ký tự

- cin
  ```
  char ch;
  cin>>ch;
  ```
- sử dụng các đoạn macro có trong thư viện <stdio.h>
  putchar: đưa ký tự ra thiết bị xuất chuẩn (stdout)
  ```
  putchar('\n');
  ```
  getchar: lấy ký tự từ thiết bị nhập chuẩn (stdin)
  ```
  ch = getchar();
  ```
- getch: lấy trực tiếp ký tự từ bàn phím không hiển thị ra màn hình
  ```
  ch = getch();
  ```
  getche(): lấy trực tiếp ký tự từ bàn phím và hiển thị ký tự ra màn hình.
  ```
  ch = getche();
  ```


getchar

```
1. #include <iostream.h>
2. int main(void)
3. {
4.    int c;
5. /* Note that getchar reads from stdin and is line
   buffered; this means it will not return until you
   press ENTER. */
6.    while ((c = getchar()) != '\n')
7.       cout<<c;
8.    return 0;
9. }
```


chuỗi ký tự (string)

- Chuỗi ký tự
- Khai báo biến kiểu chuỗi ký tự.
- Làm việc với các biến kiểu chuỗi ký tự.

Ký tự và chuỗi

- các hằng ký tự
  'a', 'N', '9', '%', '\n', ' ', '\0', …

  '\0': ký tự null
- các hằng chuỗi ký tự:
  "So n khong la so nguyen to.\n"
- các biến kiểu ký tự:

  ```
  char ch1='c', ch2='n', ch3='t';
  cout<<"Khoa:"<< ch1<<ch2<<ch3<<ch3;
  ```


Chuỗi (string)

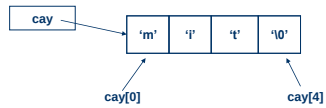
- *Khai báo*
  ```
  char <tên xâu>[độ dài] ;        // không khởi tạo
  char <tên xâu>[độ dài] = "xâu kí tự" ; // có khởi tạo
  char <tên xâu>[ ] =" xâu kí tự" ;      // có khởi tạo
  ```

(Độ dài xâu là số kí tự tối đa có thể có trong xâu. Độ dài thực sự của xâu chỉ tính từ đầu máng đến dấu kết thúc xâu)

Chuỗi(string)

- Một chuỗi là một mảng một chiều các ký tự,

  ```
  char cay[6]={'m','i','t','\0'};
  cout<<"Trong vuon co cay "<<cay;
  ```
- chính xác hơn chuỗi là mảng một chiều các ký tự kết thúc bằng ký tự kết thúc (null-terminated array of char)



Chuỗi(string)

- Lỗi khi tạo một chuỗi
  - chú ý: không có phép gán trong kiểu dữ liệu chuỗi
  - Vi dụ

```
char ten[10];
ten = "hoahong";        //không được phép
```


Chuỗi(string)

- **Một số chú ý**
- Không :
  sử dụng toán tử gán = để chép nội dung của một chuỗi sang chuỗi khác.

```
char  a[4]="hi";
char b[4];
b = a; //???
```

- Không:
  dùng toán tử == để so sánh nội dung hai chuỗi

```
char a[ ] = "hi";
char b[ ] = "there";
if(a==b) //???
   { }
```


Chuỗi (string)

- *Phương thức nhập xâu (#include <iostream.h>)*

Do toán tử nhập >> có hạn chế đối với xâu kí tự nên C++ đưa ra hàm riêng (còn gọi là phương thức) cin.getline(s,n) để nhập xâu kí tự

Khi gặp hàm cin.getline(s,n), chương trình sẽ nhìn vào bộ đệm bàn phím, lấy ra tối đa n-1 ký tự gán cho s

Chuỗi (string)

- *Ví dụ:* Nhập một ngày tháng dạng Mỹ (mm/dd/yy), đổi sang ngày tháng dạng Việt Nam rồi in ra màn hình.

```
#include <iostream.h>
main()
{
    char US[9], VN[9] = " / / " ;// khởi tạo hai dấu /
    cin.getline(US, 9) ;  // nhập ngày tháng,
                          //ví dụ "05/01/99"
    VN[0] = US[3]; VN[1] = US[4] ;        // ngày
    VN[3] = US[0]; VN[4] = US[1] ;        // tháng
    VN[6] = US[6]; VN[7] = US[7] ;        // năm
    cout << VN << endl ;
}
```


(#include <string.h>)

- **strcpy(s, t)** ;

Gán nội dung của xâu t cho xâu s (thay cho phép gán = không được dùng)

- Ví dụ:

```
char s[10], t[10] ;
t = "Face" ;                // không được dùng
s = t ;                     // không được dùng
strcpy(t, "Face") ;         // được, gán "Face" cho t
strcpy(s, t) ;              // được, sao chép t sang s
cout << s << " to " << t ;// in ra: Face to Face
```


(#include <string.h>)

- **strncpy(s, t, n)** ;

Sao chép n kí tự của t vào s.

Chú ý: Hàm không tự động gắn kí tự kết thúc xâu cho s.

Ví dụ:

```
char s[10], t[10] = "Steven";
strncpy(s, t, 5) ;          // copy 5 kí tự "Steve" vào s
s[5] = '\0' ;               // đặt dấu kết thúc xâu
// in câu: Steve is young brother of Steven
cout << s << " is young brother of " << t ;
```

(#include <string.h>)

- **strncpy(s, t, n) ;**
  Hàm này còn cho phép copy một xâu con bất kỳ của t và đặt vào s

Ví dụ các dòng lệnh chuyển đổi ngày tháng trong ví dụ trước có thể viết lại bằng cách dùng hàm strncpy như sau:

```
strncpy(VN+0, US+3, 2) ;        // ngày
strncpy(VN+3, US+0, 2) ;        // tháng
strncpy(VN+6, US+6, 2);         // năm
```


(#include <string.h>)

- **strcat(s, t);**
  Nối một bản sao của t vào sau s

Ví dụ:

```
char a[100] = "Mẫn", b[4] = "tôi";
strcat(a, " và ");
strcat(a, b);
cout << a                       // Mẫn và tôi
```


(#include <string.h>)

- **strncat(s, t, n);**
  Nối bản sao n kí tự đầu tiên của xâu t vào sau xâu s

Tương tự, có thể sử dụng cách viết strncat(s, t+k, n) để nối n kí tự từ vị trí thứ k của xâu t cho s

Ví dụ:

```
char s[20] = "Nhà " ;
char t[ ] = "vua chúa"
strncat(s, t, 3) ;              // s = "Nhà vua"
```

hoặc:

```
strncat(s, t+4, 4) ;            // s = "Nhà chúa"
```


(#include <string.h>)

- **Hàm strcmp(s, t);**
  Hàm so sánh 2 xâu s và t (thay cho các phép toán so sánh). Giá trị trả lại là hiệu 2 kí tự khác nhau đầu tiên của s và t

Ví dụ:

```
if (strcmp(s,t)) cout << "s khác t"; else cout << "s bằng t" ;
```

- **Hàm strncmp(s, t, n)** ;
  Giống hàm strcmp(s, t) nhưng chỉ so sánh tối đa n kí tự đầu tiên của hai xâu.

Ví dụ:

```
char s[ ] = "Hà Nội" , t[ ] = "Hà nội" ;
cout << strcmp(s,t) ;           // -32 (vì 'N' = 78, 'n' = 110)
cout << strncmp(s, t, 3) ;      // 0 (vì 3 kí tự đầu của s và t
                                // là như nhau)
```


(#include <string.h>)

- **stricmp(s, t) ;**
  Như strcmp(s, t) nhưng không phân biệt chữ hoa, thường.
  Ví dụ:

```
char s[ ] = "Hà Nội" , t[ ] = "hà nội" ;
cout << stricmp(s, t) ;         // 0 (vì s = t)
```

- **strupr(s);**
  Hàm đổi xâu s thành in hoa, và cũng trả lại xâu in hoa đó.
  Ví dụ:

```
char s[10] = "Ha noi" ;
cout << strupr(s) ;  // HA NOI
cout << s ;          // HA NOI (s cũng thành in hoa)
```


(#include <string.h>)

- **strlwr(s);**
  Hàm đổi xâu s thành in thường, kết quả trả lại là xâu s.
  Ví dụ:

```
char s[10] = "Ha Noi" ;
cout << strlwr(s) ;  // ha noi
cout << s ;          // ha noi (s cũng thành in thường)
```

- **strlen(s) ;**
  Hàm trả giá trị là độ dài của xâu s.
  Ví dụ:

```
char s[10] = "Ha Noi" ;
cout << strlen(s) ;          // 6
```

Biến đổi chuỗi sang số

- atoi(), atof(), atol():
  đổi chuỗi ký tự sang số.

```
int atoi(const char *s);
double atof(const char *s);
long atol(const char *s);
...
float f;
char *str = "12345.67";

f = atof(str);
cou<<str<<f;
...
```


Nhập/xuất chuỗi

- **gets:** lấy chuỗi ký tự từ thiết bị nhập chuẩn stdin
- **puts:** đưa chuỗi ký tự ra thiết bị xuất chuẩn stdout

```
char *gets(char *s);
int puts(const char *s);
```

vd:

```
char string[80];
cout<<"Input a string:";
fflush(stdin); //xóa bộ đệm bàn phím
gets(string);
cout<<"The string input was:"<<string;
puts(string);
```


Nhập/xuất chuỗi

Viết chương trình C nhập vào một câu từ bàn phím. Hiển thị mỗi từ của câu trên một dòng biết rằng các từ cách nhau bởi một khoảng trắng (space bar).

Ví dụ màn hình kết quả khi chạy chương trình

Moi nhap vao mot cau: *Happy New Year*

Happy
New
Year

Nhập/xuất chuỗi

Mã giả (Pseudo code)

```
BEGIN
DECLARE str {nhap vao xau khong qua 80 ky tu}
        INPUT str
        FOR i=0 TO do dai xau -  1 DO
                IF str[i]<>'' THEN
                        DISPLAY str[i]
                ELSE
                        IF str[i+1]<>'' THEN
                                Dua con tro xuong dong
                        END_IF
                END_IF
        END_FOR
END
```


Nhập/xuất chuỗi

```
#include<stdio.h>
#include<string.h>
int main()
{       char str[81];
        int i;
        //nhap xau
        printf("\nMoi nhap mot cau:");
        gets(str);
        for(i=0;i<strlen(str);i++)
                if(str[i]!=' ')
                        printf("%c",str[i]);
                else if(str[i+1]!=' ')
                        printf("\n");
return 0;
}
```


Chuỗi ký tự – Một số hàm thư viện

- **Tách chuỗi:**

```
char *strtok(char *s,const char *sep);
```

Trả về địa chỉ của đoạn đầu tiên. Muốn tách đoạn kế tiếp tham số thứ nhất sẽ là NULL

Chuỗi ký tự – ví dụ strtok

```
#include <stdio.h>

#define SEPARATOR "., "

int main()
{
    char s[]= "Thu strtok: 9,123.45";
    char *p;

    p = strtok(s, SEPARATOR);
    while (p != NULL)   {
        printf("%s\n", p);
        p = strtok(NULL, SEPARATOR);
    }
    return 0;
}
```

```
Thu
strtok:
9
123
45
```

91

Chuỗi ký tự – Một số hàm thư viện

- **Tìm một ký tự trên chuỗi:**

```
char *strchr(const char *s, int c);
```

Hàm trả lại vị trí đầu tiên xuất hiện của ký tự c trong chuỗi s

- **Tìm một đoạn ký tự trên chuỗi:**

```
char *strstr(const char *s1,const char *s2);
```

Hàm trả lại vị trí đầu tiên xuất hiện của chuỗi s2 trong chuỗi s1

92

Chuỗi ký tự – ví dụ tìm kiếm

```
#include <stdio.h>

int main()
{
    char s[]= "Thu tim kiem chuoi";
    char *p;

    p = strchr(s, 'm');
    printf("%s\n", p);
    p = strstr(s, "em");
    printf("%s\n", p);
    return 0;
}
```

```
m kiem chuoi
em chuoi
```

93

Chuỗi ký tự – chèn một đoạn ký tự

```
#include <stdio.h>

void StrIns(char *s, char *sub)
{
    int  len = strlen(sub);
    memmove(s + len, s, strlen(s)+1);
    strncpy(s, sub, len);
}
int main()
{
    char s[]= "Thu chen";

    StrIns(s, "123");       printf("%s\n", s);
    StrIns(s + 8, "45");    printf("%s\n", p);
    return 0;
}
```

```
123 Thu chen
123 Thu 45chen
```

94

Chuỗi ký tự – xóa một đoạn ký tự

```
#include <stdio.h>

void StrDel(char *s, int n)
{
    memmove(s, s + n, strlen(s+n)+1);
}
int main()
{
    char s[]= "Thu xoa 12345";

    StrDel(s, 4);       printf("%s\n", s);
    StrDel(s + 4, 3);   printf("%s\n", p);
    return 0;
}
```

```
xoa 12345
xoa 45
```

95

Ví dụ

- Nhập vào một chuỗi ký tự, xuất ra màn hình chuỗi bị đảo ngược thứ tự các ký tự.

```
#include<stdio.h>
#include<string.h>
#include<conio.h>
void DaoChuoi(char s1[], char s2[])
{
    int l=strlen(s1);
    for(int i=0; i<l; i++)
        s2[i]=s1[l-i-1];
    s2[i]='\0';
}
```

```
void main()
{
    char s1[100], s2[100];
    clrscr();
    printf("\nNhap vao chuoi ky tu: ");
    gets(s1);
    DaoChuoi(s1, s2);
    printf("\nKet qua sau khi dao nguo
        chuoi: %s", s2);
}
```

96

Lưu ý: kết thúc chuỗi

```
#include <stdio.h>

int  main()
{
     char other[] = "Tony Blurt";

     printf("%s\n", other);

     other[4] = '\0';

     printf("%s\n", other);

     return 0;
}
```

"Blurt" sẽ không được in ra

```
Tony Blurt
Tony
```

other

'T'	'o'	'n'	'y'	32	'B'	'l'	'u'	'r'	't'	0

97

Chuỗi ký tự – **Bài tập**

Bài 1. *chuyển chuỗi sang hoa/thường*
Bài 2. *Viết chương trình đếm số lần xuất hiện của một ký tự trong một xâu ký tự*
Bài 3: **Viết chương trình nhập vào một số nhỏ hơn 1000. Trình bày dòng chữ cho biết giá trị của số đó.**
Bài 4: **Viết chương trình cộng, trừ hai số nguyên có nhiều chữ số (dùng chuỗi).**

98