KHOA CÔNG NGHỆ THÔNG TIN
Faculty of Computer Science and Engineering-Thuyloi University

LẬP TRÌNH NÂNG CAO

Giảng viên: TS.GVC Bùi Thị Thanh Xuân
Bộ môn: Tin học và KTTT
Năm học: 2020-2021

Chương 2: Mảng và xâu ký tự

2.1. Kiểu mảng

2.1.1. Mảng dữ liệu một chiều, hai chiều

2.1.2. Tìm kiếm và sắp xếp dữ liệu trong mảng

2.1.3 Truyền tham số là mảng

2.2. Kiểu xâu kí tự

2.2.1. Xâu ký tự và các phép toán trên xâu

2.2.2. Một số bài toán trên xâu ký tự

2.2.3. Chèn/ xóa xâu kí tự

Khái niệm mảng

- Kiểu mảng là một kiểu dữ liệu gồm
  - Một số hữu hạn thành phần.
  - Các thành phần có cùng một kiểu: kiểu cơ sở hay là kiểu thành phần.
- Mỗi phần tử của mảng được tham khảo thông qua
  - Tên mảng và
  - Chỉ số của phần tử trong mảng

Tên_mảng[Chỉ_số_phần_tử]

Khai báo mảng

```
Kiểu_dữ_liệu  Tên_mảng[Kích_thước];
```

- Kiểu_dữ_liệu: kiểu của các phần tử trong mảng (nguyên, thực, ký tự, chuỗi, mảng,…)
- Tên_mảng: tên của mảng
- Kích_thước: số phần tử trong mảng

Ví dụ

```
int diemThi[50];   // Khai báo mảng 50 phần tử có kiểu số nguyên
float A[10];       // Mảng 10 phần tử kiểu số thực
```

Cấp phát bộ nhớ cho mảng

- Các phần tử trong mảng được cấp phát các ô nhớ kế tiếp nhau trong bộ nhớ
- Kích thước của mảng bằng kích thước một phần tử nhân với số phần tử

Ví dụ: int A[10]; //Mảng A gồm 10 phần tử nguyên

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]	A[9]

Kích thước của mảng A: 10 x 4 = 40 bytes

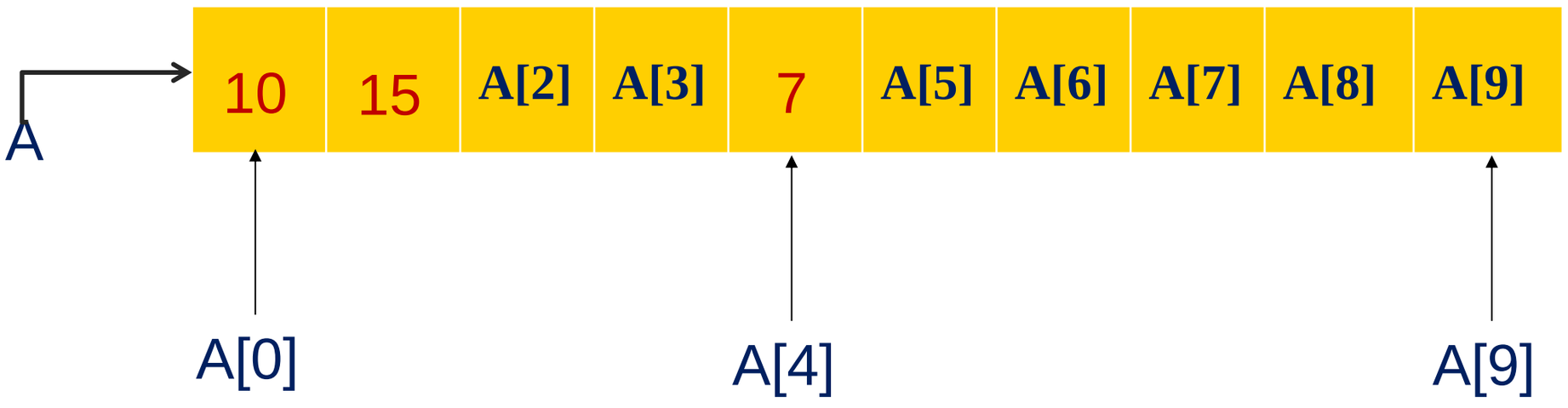
Truy nhập phần tử của mảng

- Biến mảng lưu trữ địa chỉ ô nhớ đầu tiên trong vùng nhớ được cấp phát
- Ngôn ngữ C++ đánh chỉ số các phần tử trong mảng bắt đầu từ 0
- Các phần tử của mảng được truy nhập thông qua
  - Tên mảng và
  - Chỉ số của phần tử của phần tử trong mảng

Tên_Mang[Chỉ_số_phần_tử];

Ví dụ

int A[10]; //Mảng A gồm 10 phần tử là số nguyên




```
A[0] = 10;
A[1] = 15;
A[4] = 7;
int N = A[1] + A[4]; // N = 22
```

Ví dụ

```
int A[10];
for(int i = 0; i < 10; i++)  A[i]= 2* i;
```

0	2	4	6	8	10	12	14	16	18

C++ không kiểm tra vượt quá giới hạn của mảng khi truy nhập.

Ví dụ:

```
int A[3], B[4], C[3];
cout << A[5]; //không báo lỗi
cout <<B[6];
```

A[0]	A[1]	A[2]	...	B[0]	B[1]	B[2]	B[3]	...	C[0]	C[1]	C[2]

Ví dụ

```
int A[3] = {1,2,3}, B[4] = {4,5,6,7}, C[3] = {8,9,10};
cout << A[5]; //không báo lỗi
cout <<B[6];
```

A[0]	A[1]	A[2]	...	B[0]	B[1]	B[2]	B[3]	...	C[0]	C[1]	C[2]

```
Mang A:
1       0x6ffdf0
2       0x6ffdf4
3       0x6ffdf8
Mang B:
4       0x6ffde0
5       0x6ffde4
6       0x6ffde8
7       0x6ffdec
Mang C:
8       0x6ffdd0
9       0x6ffdd4
10      0x6ffdd8
A[5] = 3        Dia chi: 0x6ffe04
B[6] = 3        Dia chi: 0x6ffdf8
```

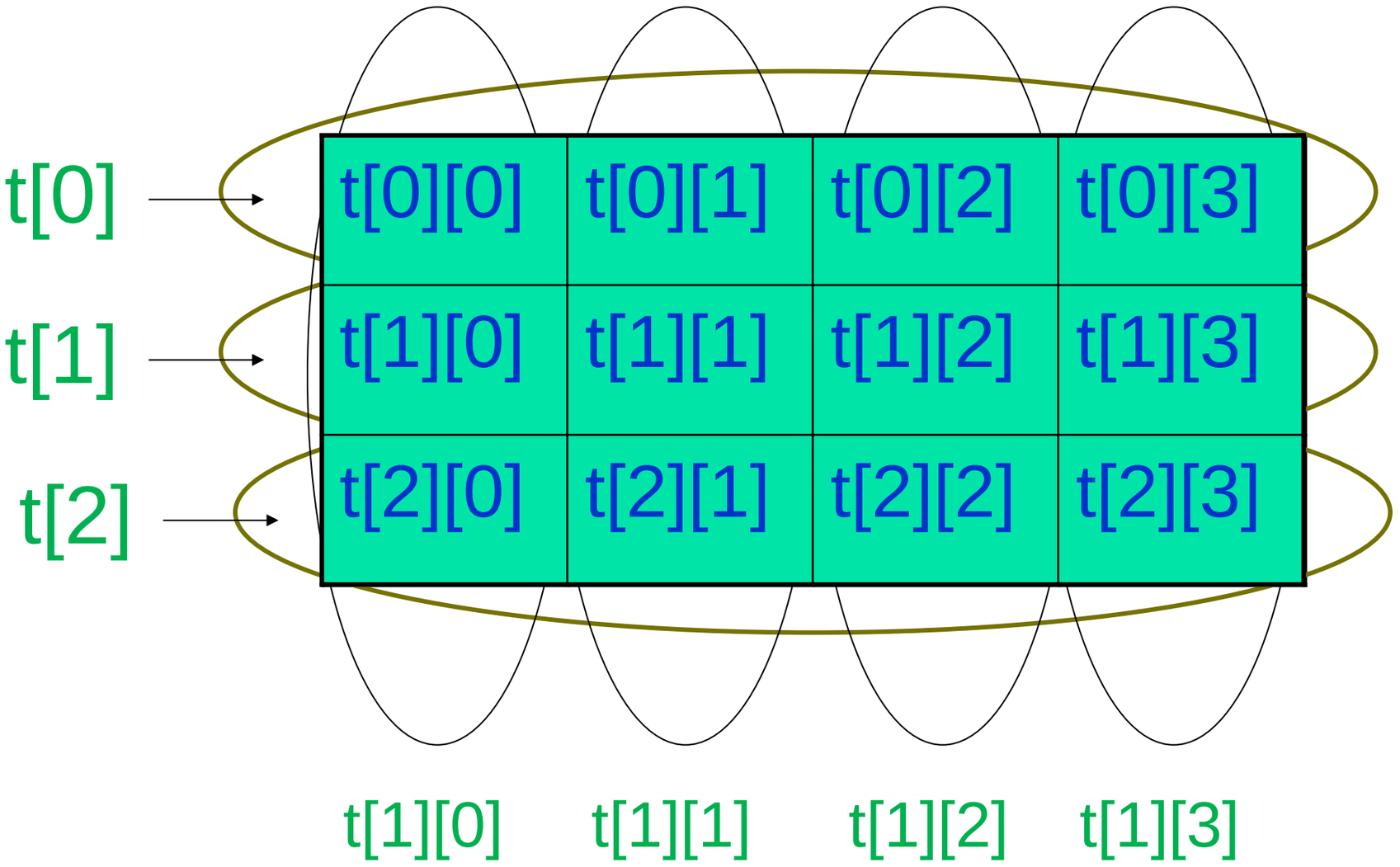
Mảng nhiều chiều

Kiểu_dữ_liệu Tên_mảng[Chiều_1] [Chiều_2]… [Chiều_N];

- Kiểu_dữ_liệu: Kiểu của mỗi phần tử trong mảng
- Tên_mảng: Tên mảng
- Chiều_1, Chiều_2,..,Chiều_N: Các hằng số nguyên, cho biết kích thước (số phần tử) của mỗi chiều
- Mảng gồm: Chiều_1 x Chiều_2 x...x Chiều_N phần tử được lưu trữ trong vùng nhớ liên tục. Các phần tử thuộc kiểu Kiểu_dữ_liệu.

Mảng nhiều chiều→Ví dụ

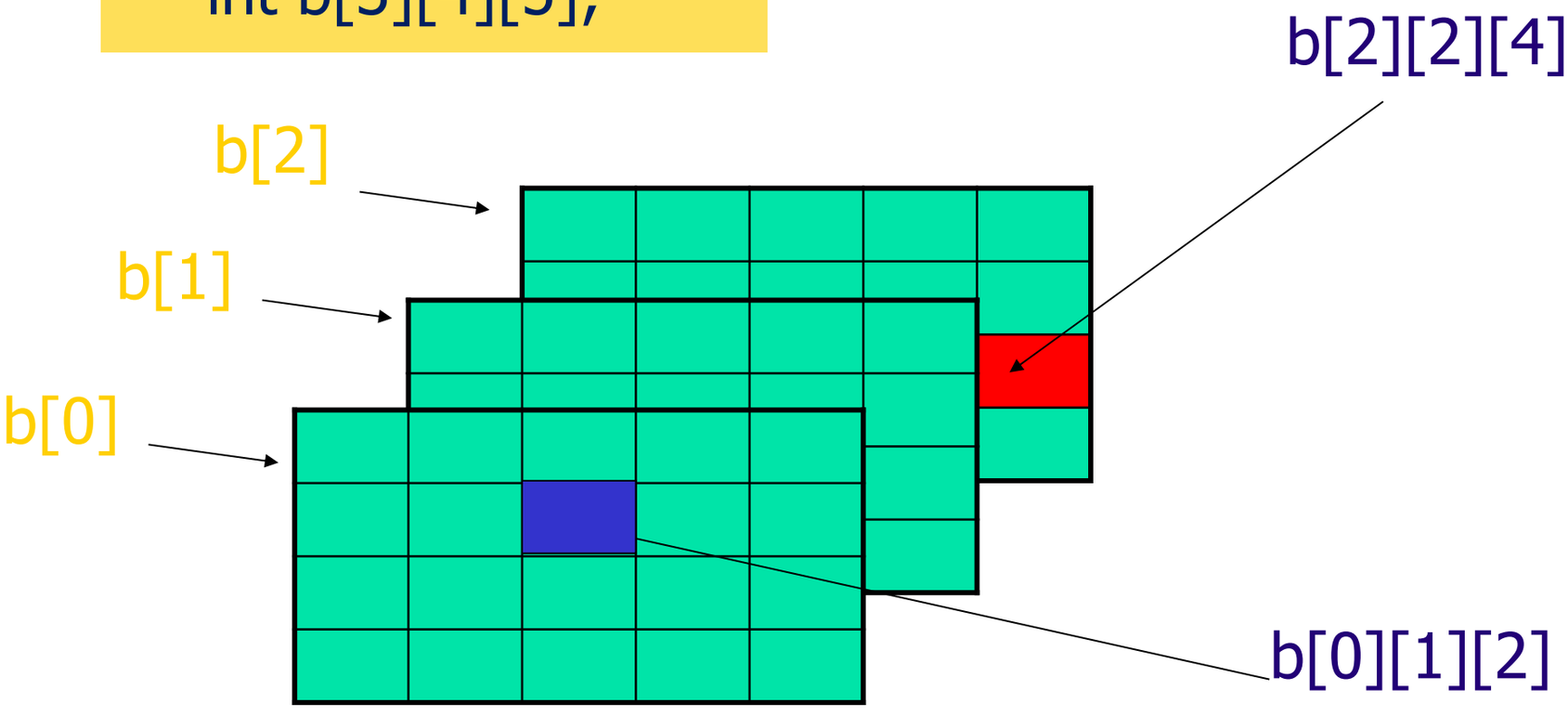
Khai báo: int t[3][4];



Mỗi phần tử của mảng có thể là một Mảng nhiều chiều

Mảng nhiều chiều→Ví dụ

```
int b[3][4][5];
```



- Mảng b gồm 3 phần tử b[0], b[1], b[2]
- Mỗi phần tử là mảng hai chiều gồm 4 hàng (hàng 0, 1, 2, 3) và 5 cột (0, 1, 2, 3, 4)
- Mỗi phần tử là một số nguyên có dấu 4 byte

Chú ý

```
int A [3][5];  // tương đương với
int A [15];    // (3 * 5 = 15)
```

Mảng nhiều chiều	Mảng giả đa chiều

#define WIDTH 5
#define HEIGHT 3
int A[HEIGHT][WIDTH];
int n, m;
int main () {
for (n=0; n<HEIGHT; n++)
for (m=0; m<WIDTH; m++)
A[n][m]=(n+1)*(m+1);
}
	#define WIDTH 5
	#define HEIGHT 3
	int A [HEIGHT * WIDTH];
	int n,m;
	int main () {
	for (n=0; n<HEIGHT; n++)
	for (m=0; m<WIDTH; m++)
	A[n*WIDTH+m]=(n+1)*(m+1);
	}

A

	0	1	2	3	4
0	1	2	3	4	5
1	2	4	6	8	10
2	3	6	9	12	15

Khởi tạo giá trị cho mảng

Các phần tử của mảng có thể được khởi tạo giá trị ngay khi khai báo

Ví dụ

```
int a[4] = {1,4,6,2};
int b[2][3]={ {1,2,3}, {4,5,6} };
int t[3][4] = {
                {1, 2, 3, 4},
                {5, 6, 7, 8},
                {9, 10, 11, 12},
                };
```

Khởi tạo giá trị cho mảng → Chú ý

- Số lượng giá trị khởi tạo không được lớn hơn số lượng phần tử trong mảng
  - Nếu số lượng này nhỏ hơn, các phần tử còn lại được khởi tạo giá trị 0

```
int A[3][4] = { {1}, {4,5} };
int B[3][4] = { };←Tất cả đều mang giá trị 0
```

```
In ma tran A:
1 0 0 0
4 5 0 0
0 0 0 0
In ma tran B:
0 0 0 0
0 0 0 0
0 0 0 0
```

Khởi tạo giá trị cho mảng → Chú ý

- Có thể xác định kích thước mảng thông qua số giá trị khởi tạo nếu để trống kích thước mảng

```
int  A[8] = {2, 4, 6, 8, 10, 12, 14, 16};
int B[][3] = {2, 4, 6, 8, 10, 12};
```

```
In day so A: 2 4 6 8 10 12 14 16
In ma tran B:
        2       4       6
        8       10      12
```

Các thao tác thường gặp

- Nhập/Xuất dữ liệu cho mảng
  - Mảng 1 chiều, ma trận
- Bài toán đếm
  - Đếm số phần tử
  - Tính toán trên các phần tử..
- Tìm kiếm phần tử
  - Lớn nhất/nhỏ nhất/bất kỳ
- Sắp xếp phần tử trong mảng
  - Theo thứ tự, theo nguyên tắc
- Chèn thêm phần tử, xóa phần tử

Nhập dữ liệu: lệnh cin >>

Ví dụ: int Table[10];

- Nhập dữ liệu cho một phần tử

```
cin >>Table[2]; //Phần tử thứ 3 của mảng
```

- Nhập dữ liệu cho cả mảng
  - Dùng vòng lặp for

```
for(int i = 0; i<10; i++)
    cin >> Table[i];
```

  - Nên in ra chỉ số phần tử khi nhập

```
for(int i = 0; i<10; i++)
    {
        cout<< "Table["<<i<<"] = ";
        cin>> Table[i];
    }
```

Nhập dữ liệu → Ví dụ 1

Nhập vào lượng mưa (mm) trong 1 năm?

```
#include <iostream>
using namespace std;
#define MONTHS 12 //khai báo hằng số tháng trong 1 năm
int main(){
  int rainfall[MONTHS]; //khai báo mảng
  for(int i=0; i<MONTHS;i++)
   {
      cout <<"Nhap luong mua tháng "<<i+1<<" la: ";
      cin >> rainfall[i]; //Nhập dữ liệu cho phần tử của mảng
  }
  return 0;
}
```

Nhập dữ liệu → Lưu ý

- Nếu số phần tử của mảng chỉ được biết tại thời điểm thực hiện chương trình (nhưng biết số phần tử tối đa)
  - Khai báo mảng với kích thước tối đa
  - Sử dụng biến nguyên lưu số phần tử thực sự của mảng.
- **Ví dụ:** Nhập vào mảng không quá 100 số thực
  - Khai báo mảng thực Table có tối đa 100 phần tử.
  - Nhập số phần tử thực sự của mảng
  - Nhập giá trị cho từng phần phần tử (dùng for)

Nhập dữ liệu→Ví dụ 2

```
#include<iostream>
using namespace std;
int main(){
  float A[100];
  int n;
  do{
     cout<<"Cho biet so phan tu cua mang: ";
     cin>>n;
  }while(n>100||n<=0);
  for(int i=0; i<n; i++){
     cout<<"A["<<i<<"] = ";
     cin >>A[i];
  }
}
```

Khai báo 1 mảng gồm tối đa 100 số thực

Nhập dữ liệu cho phần tử A[i]

Xuất dữ liệu: lệnh cout <<

Ví dụ: int Table[10];

- Hiển thị dữ liệu cho một phần tử

```
cout << Table[2];  //Phần tử thứ 3 của mảng
```

- Hiển thị dữ liệu cho cả mảng
  - Dùng vòng lặp for

```
for(int i = 0; i<10; i++)
    cout << Table[i];
```

Hiển thị dữ liệu phần tử A[i]

  - Nên in ra chỉ số phần tử khi nhập

```
for(int i = 0; i<10; i++)
    cout<< "Table["<<i<<"] = "<<Table[i];
```

- Các kiểu xuất dữ liệu
  - Hiển thị tất cả/một phần theo dòng/cột..
  - Hiển thị từng k phần tử trên một dòng…

Xuất dữ liệu trong mảng →Ví dụ 1

```
#include <iostream>
using namespace std;
int main()
{
   int  A[8] = {2, 4, 6, 8, 10, 12, 14, 16};
   for(int i=0;i<8;i++)
      cout<<A[i]<<" ";
   cout<<endl;
   for(int i=0;i<8;i++)
   {
      cout<<"\t"<<A[i];
      if (i==3)
         cout<<endl;
   }
   return 0;
}
```

```
2 4 6 8 10 12 14 16
        2       4       6       8
        10      12      14      16
--------------------------------
```

Xuất dữ liệu trong mảng →Ví dụ 2

```
#include <iostream>
using namespace std;
int main()
{
   int A[8] = {2, 4, 6, 8, 10, 12, 14, 16};
   cout<<"In day so A:";
   for(int i=0;i<8;i++)
      cout<<" "<<A[i];
   int B[3][3] = {2, 4, 6, 8, 10, 12, 14, 16,18};
   cout<<"\nIn ma tran B:"<<endl;
   for(int i=0;i<3;i++)
   {
      for(int j=0;j<3;j++)
         cout<<"\t"<<B[i][j];
      cout<<endl;
   }
   return 0;
}
```

```
In day so A: 2 4 6 8 10 12 14 16
In ma tran B:
        2       4       6
        8       10      12
        14      16      18
```

Ví dụ áp dụng
Nhập và đưa ra màn hình một ma trận m hàng và n cột

```
Nhap so hang m = 3
Nhap so cot n = 2
A[0][0] = 1
A[0][1] = 2
A[1][0] = 3
A[1][1] = 4
A[2][0] = 5
A[2][1] = 6
Ma tran da nhap:
          1          2
          3          4
          5          6
```

Nhập và đưa ra màn hình một ma trận

```
#include <iostream>
using namespace std;
int main(){
    int A[20][20],m,n;
    cout<<"Nhap so hang m = ";cin>>m;
    cout<<"Nhap so cot n = ";cin>>n;
    for(int i=0; i < m; i++ )
        for(int j=0; j < n; j++){
            cout<<"A["<<i<<"]["<<j<<"] = ";
            cin >>A[i][j];
        }
    cout<<"Ma tran da nhap:\n";
    for(int i=0; i < m; i++ ){
        for(int j=0; j < n; j++)
            cout<<"\t"<<A[i][j];
        cout<<endl;
        }
    return 0;
}
```

```
Nhap so hang m = 3
Nhap so cot n = 2
A[0][0] = 1
A[0][1] = 2
A[1][0] = 3
A[1][1] = 4
A[2][0] = 5
A[2][1] = 6
Ma tran da nhap:
        1       2
        3       4
        5       6
```

Sử dụng "\t" để căn các cột của ma trận

Ví dụ: Nhập và đưa ra màn hình một ma trận

```
#include <iostream>
#include <iomanip>
using namespace std;
int main(){
    int A[20][20],m,n;
    cout<<"Nhap so hang m = ";cin>>m;
    cout<<"Nhap so cot n = ";cin>>n;
    for(int i=0; i < m; i++ )
        for(int j=0; j < n; j++){
            cout<<"A["<<i<<"]["<<j<<"] = ";
            cin >>A[i][j];
        }
    cout<<"Ma tran da nhap:\n";
    for(int i=0; i < m; i++ ){
        for(int j=0; j < n; j++)
            cout<<setw(4)<<A[i][j];
        cout<<endl;
        }
    return 0;
}
```

```
Nhap so hang m = 3
Nhap so cot n = 3
A[0][0] = 1
A[0][1] = 2
A[0][2] = 3
A[1][0] = 4
A[1][1] = 5
A[1][2] = 6
A[2][0] = 7
A[2][1] = 8
A[2][2] = 9
Ma tran da nhap:
   1   2   3
   4   5   6
   7   8   9
```

Nhập và đưa ra màn hình một ma trận: sử dụng hàm không tham số và biến toàn cục

```
#include <iostream>
#include <iomanip>
using namespace std;

int A[20][20], m, n;
void Nhap();
void Xuat();

int main(){
    Nhap();
    Xuat();
    return 0;
}
void Nhap()
{}
void Xuat()
{}
```

Khai bao bao bien toan cuc

Dinh nghia cac ham

```
Nhap so hang m = 3
Nhap so cot n = 3
A[0][0] = 1
A[0][1] = 2
A[0][2] = 3
A[1][0] = 4
A[1][1] = 5
A[1][2] = 6
A[2][0] = 7
A[2][1] = 8
A[2][2] = 9
Ma tran da nhap:
   1   2   3
   4   5   6
   7   8   9
```

Nhập và đưa ra màn hình một ma trận: sử dụng hàm không tham số và biến toàn cục

```
#include <iostream>
#include <iomanip>
using namespace std;
int A[20][20],m,n;
void Nhap()
{
    cout<<"Nhap so hang m = ";cin>>m;
    cout<<"Nhap so cot n = ";cin>>n;
    for(int i=0; i < m; i++ )
        for(int j=0; j < n; j++)
            {
                cout<<"A["<<i<<"]["<<j<<"] = ";
                cin >>A[i][j];
            }
}
void Xuat()
{
    cout<<"Ma tran da nhap:\n";
    for(int i=0; i < m; i++ ){
            for(int j=0; j < n; j++)
                cout<<setw(4)<<A[i][j];
            cout<<endl;
        }
}
```

```
int main(){
    Nhap();
    Xuat();
    return 0;
}
```

```
Nhap so hang m = 3
Nhap so cot n = 3
A[0][0] = 1
A[0][1] = 2
A[0][2] = 3
A[1][0] = 4
A[1][1] = 5
A[1][2] = 6
A[2][0] = 7
A[2][1] = 8
A[2][2] = 9
Ma tran da nhap:
   1   2   3
   4   5   6
   7   8   9
```

Cộng 2 ma trận cùng kích thước
C = A + B

Viết chương trình:
- Nhập vào 2 ma trận A và B gồm mxn số nguyên.
- In ra ma trận tổng C = A + B

$$C_{ij} = A_{ij} + B_{ij}$$

```
Nhap so hang m = 3
Nhap so cot n = 2
Nhap ma tran A:
A[0][0] = 1
A[0][1] = 2
A[1][0] = 3
A[1][1] = 4
A[2][0] = 5
A[2][1] = 6
Nhap ma tran B:
B[0][0] = 2
B[0][1] = 1
B[1][0] = 3
B[1][1] = 4
B[2][0] = 8
B[2][1] = 7
Ma tran tong C = A+B:
   3   3
   6   8
  13  13
```

Ví dụ: Cộng 2 ma trận cùng kích thước C = A + B

```
#include <iostream>
#include <iomanip>
using namespace std;
int main(){
    int A[100][100],B[100][100],m,n;
    cout<<"Nhap so hang m = ";cin>>m;
    cout<<"Nhap so cot n = ";cin>>n;
    cout<<"Nhap ma tran A:\n";
    for(int i=0; i < m; i++ )
        for(int j=0; j < n; j++)
            {
                cout<<"A["<<i<<"]["<<j<<"] = ";
                cin >>A[i][j];
            }
    cout<<"Nhap ma tran B:\n";
    for(int i=0; i < m; i++ )
        for(int j=0; j < n; j++)
            {
                cout<<"B["<<i<<"]["<<j<<"] = ";
                cin >>B[i][j];
            }
```

Khai bao 2 ma tran cung cap

Ví dụ: Cộng 2 ma trận cùng kích thước C = A + B

```
   cout<<"Ma tran tong C = A+B:\n";
   for(int i = 0; i < m; i++ )
      {
         for(int j = 0; j < n; j++)
            cout<<setw(4)<< A[i][j]+ B[i][j];
            cout<<endl;
      }
   return 0;
}
```

C[i][j]=A[i][j]+B[i][j];

```
Nhap so hang m = 3
Nhap so cot n = 2
Nhap ma tran A:
A[0][0] = 1
A[0][1] = 2
A[1][0] = 3
A[1][1] = 4
A[2][0] = 5
A[2][1] = 6
Nhap ma tran B:
B[0][0] = 2
B[0][1] = 1
B[1][0] = 3
B[1][1] = 4
B[2][0] = 8
B[2][1] = 7
Ma tran tong C = A+B:
   3   3
   6   8
  13  13
```

Ví dụ áp dụng

Viết chương trình:

- Viết hàm nhập vào một ma trận A gồm m hàng n cột các số nguyên
- Viết hàm hiển thị ma trận vừa nhập
- Viết hàm in ra các số nguyên tố có trong ma trận
- Viết hàm min, max trong ma trận

```
void nhap();
void xuat();
int tim_max();
int tim_min();
bool nguyento(int n);
void nguyento();
```

Ví dụ áp dụng

```
#include <iostream>
#include <cmath>
using namespace std;
int A[100][100], m, n;
void nhap();
void xuat();
int tim_max();
int tim_min();
bool nguyento(int n);
void nguyento();
```

```
int main ()
{   nhap();
    xuat();
    nguyento();
    cout<<"\nMin cua ma
tran: "<<tim_min();
    cout<<"\nMax cua ma
tran: "<<tim_max();
    return 0;
}
```

Ví dụ áp dụng

```
using namespace std;
int A[100][100], m, n;
void nhap()
{
    cout<<"Nhap ma tran: "<<endl;
    cout<<"Nhap so hang m = "; cin>>m;
    cout<<"Nhap so cot n = "; cin>>n;
    for(int i=0; i< m; i++)
        for(int j=0; j< n; j++)
            {
                cout<<"A["<<i<<"]["<<j<<"] = ";
                cin>>A[i][j];
            }
}
void xuat()
{
    cout<<"In ma tran: "<<endl;
    for(int i=0; i< m; i++)
        {
            for(int j=0; j< n; j++)
                cout<<" "<<A[i][j];
            cout<<endl;
        }
}
```

Ví dụ áp dụng

```
int tim_max() {
    int max = A[0][0];
    for(int i=0; i< m; i++)
        for(int j=0; j< n; j++)
            if(max<A[i][j])
                max = A[i][j];
    return max;
}
int tim_min() {
    int min = A[0][0];
    for(int i=0; i< m; i++)
        for(int j=0; j< n; j++)
            if(min>A[i][j])
                min = A[i][j];
    return min;
}
```

```
bool nguyento(int n) {
    if (n <= 1)
        return false;
    for (int i = 2; i <= sqrt(n); i++)
        if (n%i == 0)
            return false;
    return true;
}
void nguyento() {
    cout<<"Cac so nguyen to trong ma tran:";
    for(int i=0; i< m; i++)
        for(int j=0; j< n; j++)
            if (nguyento(A[i][j]))
                cout<<" "<<A[i][j];
}
int main ()
{   nhap();
    xuat();
    nguyento();
    cout<<"\nMin cua ma tran: "<<tim_min();
    cout<<"\nMax cua ma tran: "<<tim_max();
    return 0;
}
```

Bài tập áp dụng

Viết chương trình:

- Nhập vào một ma trận A vuông gồm n hàng n cột các số nguyên
- Hiển thị ma trận vừa nhập
- In ra ma trận tam giác trên và tam giác dưới của ma trận A

Bài tập áp dụng

Ma trận tam giác trên và tam giác dưới của ma trận A

$$A = \left(A_{ij}\right)_{nxn}$$

Ma trận tam giác trên $B = \left(B_{ij}\right)_{nxn} \Rightarrow B_{ij} = \begin{cases} A_{ij} & i \le j \\ 0 & i > j \end{cases}$

Ma trận tam giác dưới $C = \left(C_{ij}\right)_{nxn} \Rightarrow C_{ij} = \begin{cases} 0 & i < j \\ A_{ij} & i \ge j \end{cases}$

Bài tập áp dụng

```
#include <iostream>
#include <iomanip>
using namespace std;
int main(){
    int A[100][100],n;
    cout<<"Nhap kich thuoc n = ";cin>>n;
    cout<<"Nhap ma tran A:\n";
    for(int i=0; i < n; i++ )
        for(int j=0; j < n; j++)
           {
                cout<<"A["<<i<<"]["<<j<<"] = ";
                cin >>A[i][j];
            }
```

```
cout<<"Ma tran tam giac tren B:\n";
    for(int i=0; i < n; i++ )
        {
        for(int j=0; j < n; j++)
           if(i>j) cout<<setw(2)<<0;
           else
                cout<<setw(2)<<A[i][j];
        cout<<endl;
        }
    cout<<"Ma tran tam giac duoi C:\n";
    for(int i=0; i < n; i++ )
        {
        for(int j=0; j < n; j++)
           if(i<j) cout<<setw(2)<<0;
           else
                cout<<setw(2)<<A[i][j];
        cout<<endl;
        }
    return 0;
}
```

```
Nhap kich thuoc n = 3
Nhap ma tran A:
A[0][0] = 1
A[0][1] = 1
A[0][2] = 1
A[1][0] = 1
A[1][1] = 1
A[1][2] = 1
A[2][0] = 1
A[2][1] = 1
A[2][2] = 1
Ma tran tam giac tren B:
 1 1 1
 0 1 1
 0 0 1
Ma tran tam giac duoi C:
 1 0 0
 1 1 0
 1 1 1
```

Truyền mảng vào hàm

Định nghĩa hàm

Bản chất truyền kiểu tham chiếu cho biến mảng

Truyền mảng vào hàm

```
#include <iostream>
using namespace std;
void nhap(int a[],int n)
{
    for (int i=0; i<n; i++)
      {
      cout<<"a["<<i<<"]="; cin>>a[i];
      }
}
void hienThi(int a[],int n)
{
    for(int i=0; i<n; i++)
      cout<<"a["<<i<<"]="<<a[i]<<endl;
}
int main ()
{
   int a[100], n;
   cout<<"Nhap so phan tu cua day (<100) =";
   cin>>n;
   nhap(a,n);
   cout<<"Day so vua nhap:"<<endl;
   hienThi(a,n);
   return 0;
}
```

Truyền mảng vào hàm

Định nghĩa hàm

Ngăn ngừa việc thay đổi giá trị của mảng – Từ khóa const

Truyền mảng vào hàm

```
#include <iostream>
using namespace std;
void nhap(int a[],int n)
{
    for(int i=0; i<n; i++)
      {
      cout<<"a["<<i<<"]="; cin>>a[i];
      }
}
void hienThi(const int a[],int n)
{
    for(int i=0; i<n; i++)
      cout<<"a["<<i<<"]="<<a[i]<<endl;
}
int main ()
{
   int a[100], n;
   cout<<"Nhap so phan tu cua day (<100) =";
   cin>>n;
   nhap(a,n);
   cout<<"Day so vua nhap:"<<endl;
   hienThi(a,n);
   return 0;
}
```

Ví dụ áp dụng

Viết chương trình:

- Viết hàm nhập vào một dãy gồm n số nguyên A[0], A[1],...,A[n-1] từ bàn phím.
- Viết hàm hiển thị dãy số vừa nhập
- Viết hàm in ra các số nguyên tố có trong dãy
- Viết hàm tính tổng các số chính phương có trong dãy.

Ví dụ áp dụng

```
So phan tu n = 6
Phan tu thu 0: 5
Phan tu thu 1: 4
Phan tu thu 2: 3
Phan tu thu 3: 6
Phan tu thu 4: 3
Phan tu thu 5: 7
In day: 5 4 3 6 3 7
So nguyen to co trong day: 5 3 3 7
Tong so chinh phuong trong day la: 4
```

```
#include <iostream>
#include <cmath>
using namespace std;
void nhap(int b[], int &n);
void xuat(const int b[], int n);
bool nguyento(int n);
void nguyento(const int b[], int n);
bool chinhphuong(int n);
int chinhphuong(const int b[], int n);
int main() {
   int n,a[100];
   nhap(a,n);
   xuat(a,n);
   nguyento(a,n);
   cout<<"\nTong so chinh phuong trong day la: "<<chinhphuong(a,n);
   return 0;
}
```

Ví dụ áp dụng

Viết chương trình:

- Viết hàm nhập vào một ma trận A gồm m hàng n cột các số nguyên
- Viết hàm hiển thị ma trận vừa nhập

Truyền mảng vào hàm

```
#include <iostream>
#include <iomanip>
using namespace std;

void Nhap(int A[100][100],int &m,int &n)
{
    cout<<"Nhap so hang m = ";cin>>m;
    cout<<"Nhap so cot n = ";cin>>n;
    for(int i=0; i < m; i++ )
        for(int j=0; j < n; j++)
           {
               cout<<"A["<<i<<"]["<<j<<"] = ";
               cin >>A[i][j];
            }
}
```

```
void Xuat(const int A[100][100],int m,int n)
{
    cout<<"Ma tran da nhap:\n";
    for(int i=0; i < m; i++ )
        {
            for(int j=0; j < n; j++)
                cout<<setw(4)<<A[i][j];
            cout<<endl;
        }
}
int main(){
    int A[100][100],m,n;
    Nhap(A,m,n);
    Xuat(A,m,n);
    return 0;
}
```

```
Nhap so hang m = 2
Nhap so cot n = 2
A[0][0] = 1
A[0][1] = 2
A[1][0] = 3
A[1][1] = 4
Ma tran da nhap:
   1   2
   3   4
```

Truyền mảng vào hàm

```
void Nhap(int A[][100],int &m,int &n){
    cout<<"Nhap so hang m = ";cin>>m;
    cout<<"Nhap so cot n = ";cin>>n;
    for(int i=0; i < m; i++ )
        for(int j=0; j < n; j++)
            {
                cout<<"A["<<i<<"]["<<j<<"] = ";
                cin >>A[i][j];
            }
}
void Xuat(const int A[][100],int m,int n){
    cout<<"Ma tran da nhap:\n";
    for(int i=0; i < m; i++ )
        {
            for(int j=0; j < n; j++)
                cout<<setw(4)<<A[i][j];
            cout<<endl;
        }
}
```

```
Nhap so hang m = 2
Nhap so cot n = 2
A[0][0] = 1
A[0][1] = 2
A[1][0] = 3
A[1][1] = 4
Ma tran da nhap:
   1   2
   3   4
```

Ví dụ áp dụng

Viết chương trình:

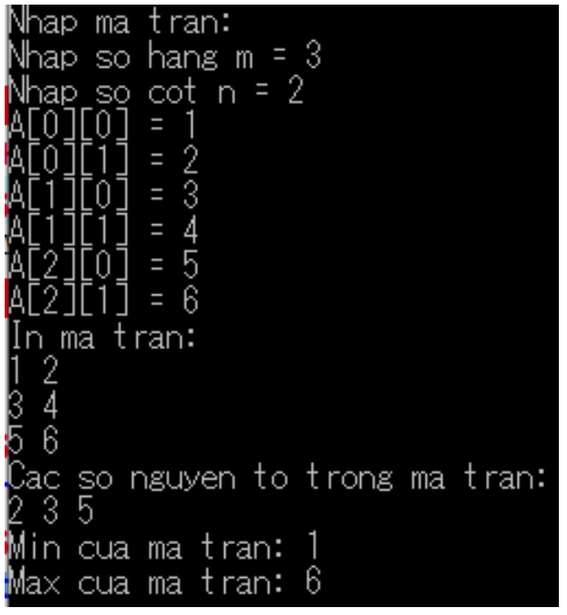
- Viết hàm nhập vào một ma trận A gồm m hàng n cột các số nguyên
- Viết hàm hiển thị ma trận vừa nhập
- Viết hàm min, max trong ma trận
- Viết hàm in ra các số nguyên tố có trong ma trận

```
Nhap ma tran:
Nhap so hang m = 3
Nhap so cot n = 2
A[0][0] = 1
A[0][1] = 2
A[1][0] = 3
A[1][1] = 4
A[2][0] = 5
A[2][1] = 6
In ma tran:
1 2
3 4
5 6
Cac so nguyen to trong ma tran:
2 3 5
Min cua ma tran: 1
Max cua ma tran: 6
```

Ví dụ áp dụng

Viết chương trình, trong đó:

- Viết hàm nhập vào một ma trận A gồm m hàng n cột các số nguyên
- Viết hàm hiển thị ma trận vừa nhập
- Viết hàm min, max trong ma trận
- Viết hàm in ra các số nguyên tố có trong ma trận




```
void nhap(int A[][100], int &m, int &n);
void xuat(const int A[][100], int m, int n);
int tim_max(const int A[][100], int m, int n);
int tim_min(const int A[][100], int m, int n);
void nguyento(const int A[][100], int m, int n);
```

Ví dụ áp dụng

Viết chương trình:

- Viết hàm nhập vào ma trận gồm các số nguyên với m hàng và n cột
- Viết hàm hiển thị ma trận vừa nhập
- Nhập ma trận A gồm m hàng q cột và ma trận B gồm q hàng và n cột. Tìm ma trận tích C = A.B (Điều kiện: số cột của A phải bằng số hàng của B)

$$A = \left(A_{ik}\right)_{mxp}; \quad B = \left(B_{kj}\right)_{pxn}$$

$$C = \left(C_{ij}\right)_{mxn}; C_{ij} = \sum_{k=1}^{p} A_{ik} B_{kj} \; i = 1,..,m, \; j = 1,..,n$$

Ví dụ áp dụng

Chèn thêm từ khóa **const**

```
void nhap(int A[][100], int &m, int &n)
{
    cout<<"Nhap so hang: "; cin>>m;
    cout<<"Nhap so cot: "; cin>>n;
    for(int i=0; i< m; i++)
        for(int j=0; j< n; j++)
            {
                cout<<"Phan tu ["<<i<<"]["<<j<<"] = ";
                cin>>A[i][j];
            }
}
void xuat(int A[][100], int m, int n)
{
    for(int i=0; i< m; i++)
        {
            for(int j=0; j< n; j++)
                cout<<A[i][j]<<" ";
            cout<<endl;
        }
}
```

Ví dụ áp dụng

```
int main ()
{   int A[100][100], B[100][100], C[100][100], m1, n1, m2, n2;
    cout<<"Nhap ma tran A:\n"; nhap(A,m1,n1);
    cout<<"Nhap ma tran B:\n"; nhap(B,m2,n2);
    if(n1==m2)
        {
            for(int i = 0; i < m1; i++)
                for(int j = 0; j < n2; j++)
                {
                    C[i][j] = 0;
                    for(int k = 0; k < m2; k++)
                        C[i][j] = C[i][j] + A[i][k]*B[k][j];
                }
            cout<<"In ma tran tich:\n"; xuat(C,m1,n2);
        }
    else
        cout<<"Hai ma tran khong thoa man cho phep nhan!";
    return 0;
}
```

Ví dụ áp dụng

```
void NhanMT(const int A[][100], int m1, int n1, const int B[][100], int m2, int n2, int C[][100])
{
    for(int i=0; i< m1; i++)
        for(int j=0; j< n2; j++)
            {
                C[i][j] = 0;
                for(int k=0; k<m2; k++)
                    C[i][j] = C[i][j] + A[i][k]*B[k][j];
            }
}
```

```
int main ()
{   int A[100][100], B[100][100], C[100][100], m1, n1, m2, n2;
    cout<<"Nhap ma tran A:\n"; nhap(A,m1,n1);
    cout<<"Nhap ma tran B:\n"; nhap(B,m2,n2);
    if(n1==m2)
        {
            NhanMT(A,m1,n1,B,m2,n2,C);
            cout<<"In ma tran tich:\n"; xuat(C,m1,n2);
        }
    else
        cout<<"Hai ma tran khong thoa man cho phep nhan!";

    return 0;
}
```

Đếm số phần tử thỏa mãn điều kiện

- Duyệt từng phần tử của dãy (dùng for)
- Nếu phần tử xét thỏa mãn điều kiện
  - Ghi nhận
- Chuyển sang xem xét phần tử tiếp theo

Ví dụ: Đếm số tháng có lượng mưa lớn hơn 50mm

```
int dem = 0;
for(i = 0; i < MONTHS; i++)
 if(rainfall[i] > 50)
      dem++;
cout<<"Thang mua nhieu hon 50mm: " << dem;
```

Ví dụ: Nhập mảng gồm n số nguyên (0<n<100), đưa ra TBC các số chia hết cho 7 có trong mảng.

```
#include <iostream>
using namespace std;
int main(){
   int A[100], n, d = 0, S = 0;
   do{
    cout<<"So phan tu cua mang (<100) n = "; cin>>n;
   }while(n<=0||n>=100);
   for(int i = 0; i<n; i++){
    cout<<"A["<<i<<"] = ";
    cin>>A[i];
   }
   for(int i = 0; i<n; i++)
    if(A[i]%7 == 0){
        d++;
        S += A[i];
   }
   if(d>0)
    cout<<"TBC so chia het cho 7: "<<S*1.0/d;
   else
    cout<<"Trong day khong co so chia het cho 7.";
   return 0;
}
```

```
So phan tu cua mang (<100) n = 8
A[0] = 7
A[1] = 14
A[2] = 2
A[3] = 56
A[4] = 24
A[5] = 12
A[6] = 72
A[7] = 49
TBC so chia het cho 7: 31.5
```

Tìm kiếm phần tử

Tìm phần tử lớn nhất (nhỏ nhất)

- Giả sử phần tử đó là phần tử đầu tiên
- Lần lượt so sánh với các phần tử còn lại
  - Nếu phần tử mới của dãy lớn hơn ⇒ coi đây là phần tử lớn nhất và tiếp tục so sánh với phần tử kế
  - Nếu không đúng, so sánh tiếp với phần tử kế

Ví dụ: Tìm tháng có lượng mưa nhiều nhất trong năm

```
max = rainfall[0];
for(int i=1;i<MONTHS;i++)
   if(rainfall[i]>max)
      max = rainfall[i];
cout <<Luong mua nhieu nhat la: << max;
```

Tìm kiếm phần tử

- Tìm kiếm các phần tử thỏa mãn điều kiện (*giống bài toán đếm*)
  - Dùng for duyệt toàn bộ
  - Nếu cần thiết, dùng thêm mảng ghi lại chỉ số

Ví dụ: Đưa ra danh sách các tháng có lượng mưa nhiều hơn 50mm

```
cout<<“Thang co luong mua lon hon 500mm”;
for(int i = 0; i < MONTHS; i++)
 if(rainfall[i] > 50)
     cout<<"\nThang " <<i+1;
```

Tìm kiếm phần tử (tiếp)

- Tìm phần tử đầu tiên của danh sách
  - Dùng vòng lặp for kết hợp với break;
  - Dùng vòng lặp while

Ví dụ

Đưa ra phần tử đầu của mảng có giá trị bằng k;

Tìm kiếm phần tử →Ví dụ

```
int Table[100];
int N, i, k, f;  //N: số phần tử, k phần tử cần tìm
```

Dùng for

```
   for(i = 0; i < N; i++)
      if(Table[i] == k) break;
   if(i< N) cout<<"Tim thay tai vi tri "<<i;
```

Dùng while

```
   i=0; f = 0;   //f: found. f = 1 ⇔ k is found
   while(i < N && f==0){
     if(Table[i] == k) f = 1;
     else i++;}
   if (f==1)
     cout<<"Tim thay tai vi tri "<< i;
```

Đếm, tìm kiếm phần tử thỏa mãn điều kiện

Viết chương trình:

- Nhập số nguyên n (10<n<100)
- Nhập dãy số nguyên gồm n phần tử
- Hiển thị các số nguyên tố có trong dãy
- Tính tổng và TBC các số chia hết cho 5 trong dãy
- Tìm các số chính phương có trong dãy số
- Tìm min, max

Ví dụ: Đếm, tìm kiếm phần tử thỏa mãn điều kiện

```
void nhap(int a[], int &n)
{
    do{
        cout<<"So phan tu n = "; cin>>n;
    }while(n<=10||n>=100);
    for(int i=0;i<n;i++)
        {
            cout<<"Phan tu thu "<<i<<": ";
            cin>>a[i];
        }
}
void xuat(const int a[], int n)
{
    cout<<"In day:";
    for(int i=0;i<n;i++)
        cout<<" "<<a[i];
}
```

```
bool nguyento(int n) {
    if (n <= 1)
        return false;
    for (int i = 2; i <= sqrt(n); i++)
        if (n%i == 0)
            return false;
    return true;
}
void nguyento(const int a[], int n)
{
    cout<<"\nSo nguyen to co trong day:";
    for(int i=0;i<n;i++)
        if(nguyento(a[i]))
            cout<<" "<<a[i];
}
```

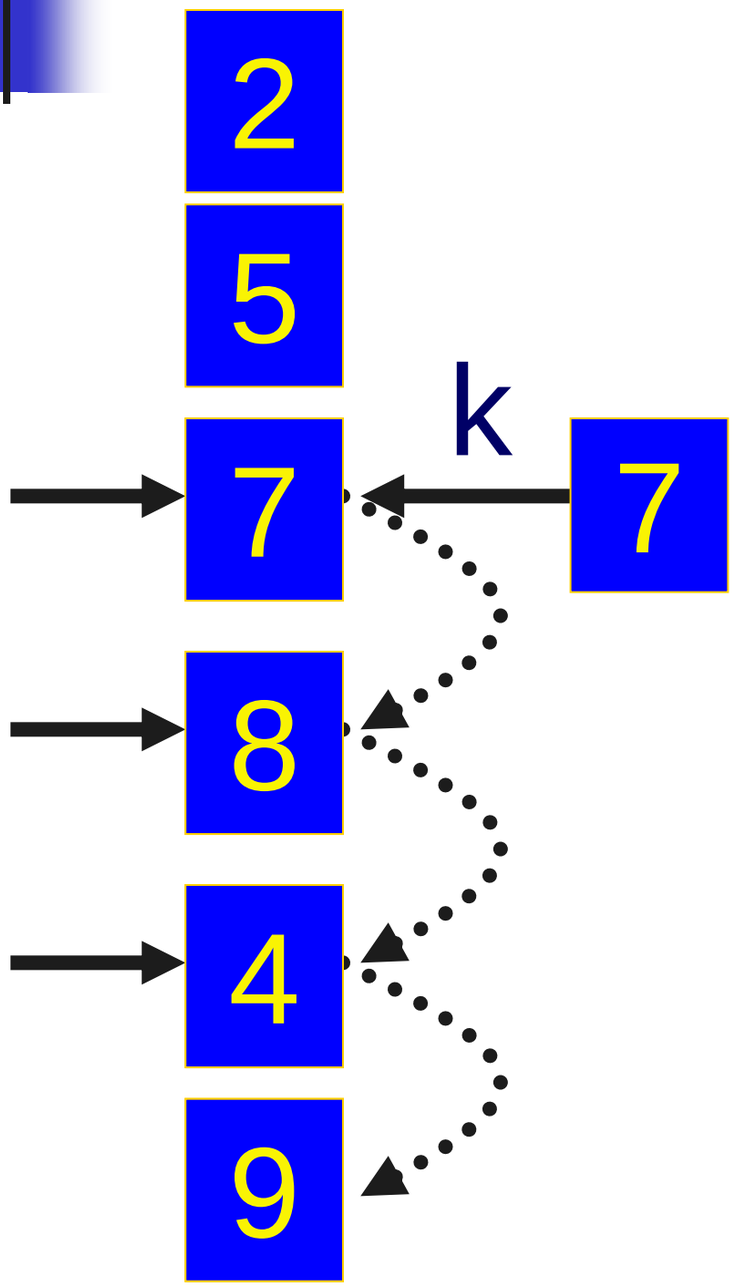
```
void chia5(const int a[], int n)
{
    int tong = 0; int dem =0;
    for(int i=0;i<n;i++)
        if(a[i]%5 ==0) {
            tong +=a[i];
            dem++;
        }
    cout<<"\nTong cac so chia het cho 5 trong day: "<<tong;
    cout<<"\nTBC cac so chia het cho 5 trong day: "<<(float)tong/dem;

}
bool chinhphuong(int n) {
    if (n>=0 && sqrt(n) == floor(sqrt(n)))
        return true;
    else
        return false;
}
void chinhphuong(const int a[], int n){
    cout<<"\nSo chinh phuong co trong day:";
    for(int i=0;i<n;i++)
        if(chinhphuong(a[i]))
            cout<<" "<<a[i];
}
int tim_max(const int a[], int n) {
    int max = a[0];
    for(int i=1;i<n;i++)
        if(max<a[i])
            max = a[i];
    return max;
}
```

Ví dụ: Đếm, tìm kiếm phần tử thỏa mãn điều kiện

```
int tim_min(const int a[], int n) {
    int min = a[0];
    for(int i=1;i<n;i++)
        if(min>a[i])
            min = a[i];
    return min;
}
int main() {
    int n, a[100];
    nhap(a,n); xuat(a,n);
    nguyento(a,n); chia5(a,n);
    chinhphuong(a,n);
    cout<<"\nKet qua so lon nhat cua day so la: "<<tim_max(a,n);
    cout<<"\nKet qua so nho nhat cua day so la: "<<tim_min(a,n);
    return 0;
}
```

Bài toán chèn phần tử x vào vị trí k




```
for(i = N; i > k; i--)
    A[i] = A[i-1];
A[k] = x;
N = N + 1;
```

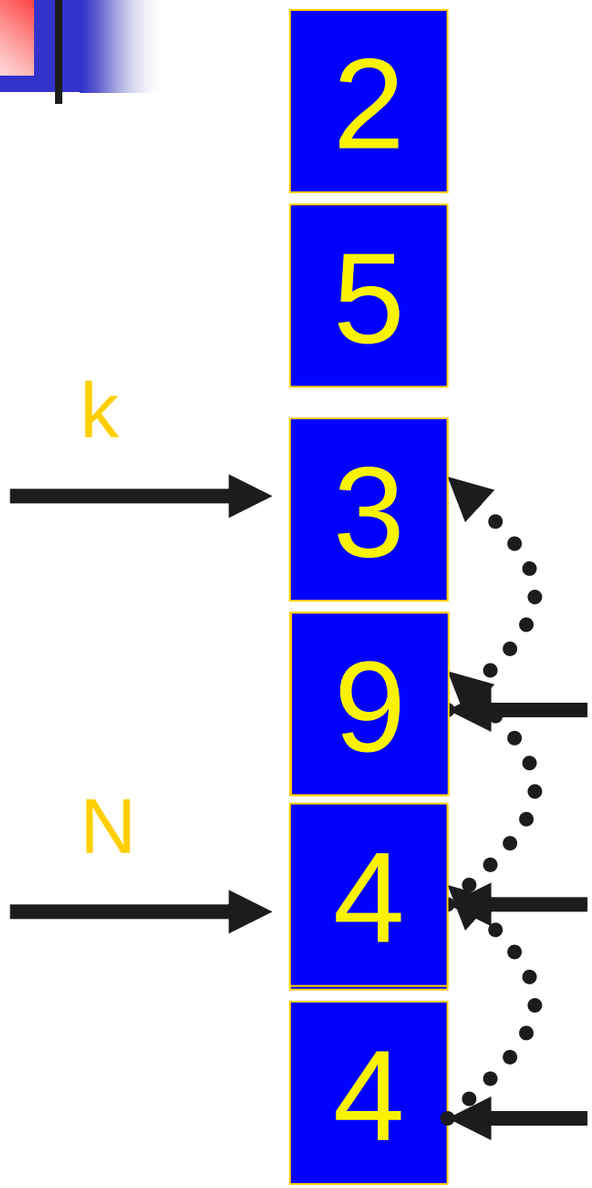
Chú ý:

N = MAX: không chèn được

k > N → Chèn vào vị trí N;

k< 0 → Chèn vào vị trí 0

Bài toán xóa phần tử ở vị trí k (0 ≤ k < N)




```
for(i = k+1; i < N; i++)
    A[i-1] = A[i];
N = N - 1;
```

Bài toán sắp xếp theo thứ tự

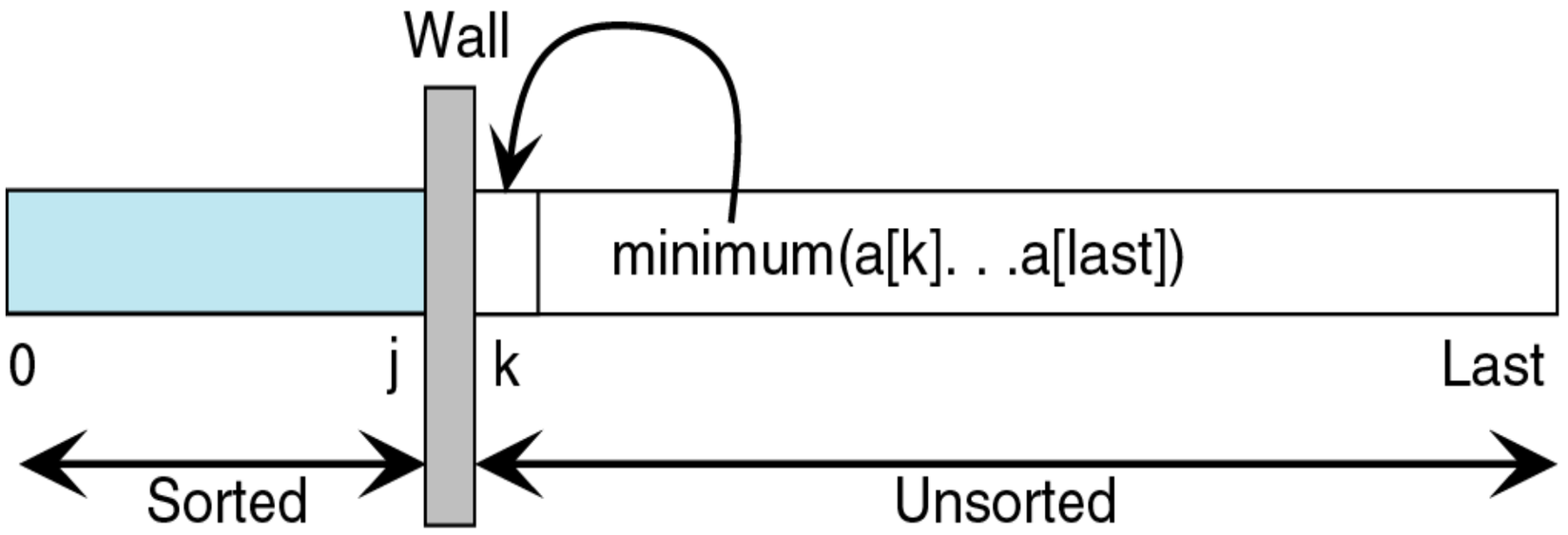
- Cho mảng phần tử, sắp xếp theo thứ tự tăng/giảm
- Các thuật toán
  - Sắp xếp lựa chọn (Selection sort)
  - Sắp xếp thêm dần (Insertion sort)
  - Sắp xếp nổi bọt (Bubble sort)
  - Sắp xếp vun đống (Heap sort)
  - Sắp xếp nhanh (Quick sort)
  - Sắp xếp trộn (Merge sort)

Bài toán sắp xếp tăng →Thuật toán lựa chọn

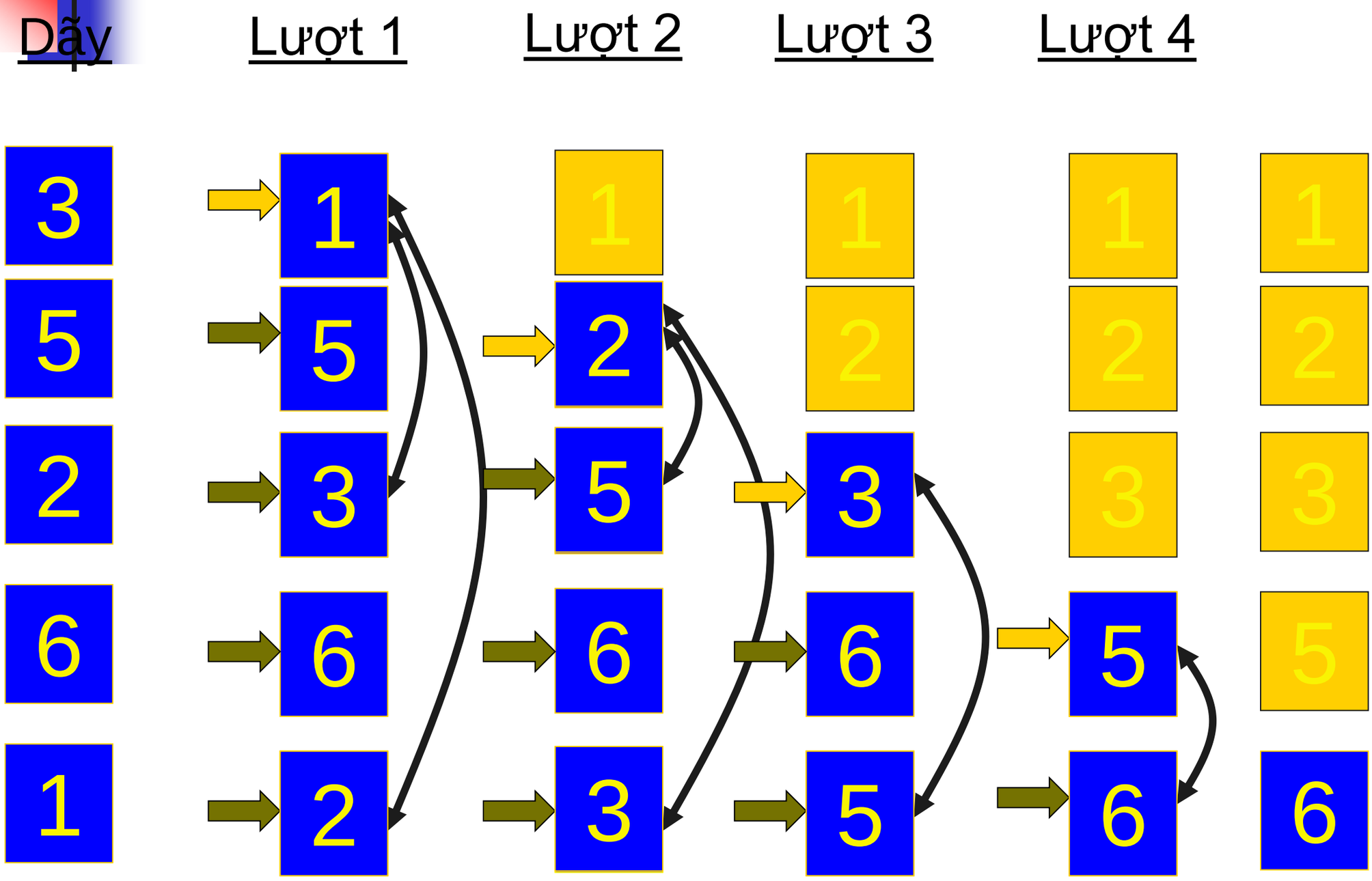
Nguyên tắc

Tại lượt sắp thứ k, tìm phần tử nhỏ nhất trong số các phần tử chưa được sắp xếp ([k..last]) và đổi chỗ cho phần tử thứ k (có chỉ số k-1)

- Khi k = 1, phần tử thứ nhất (chỉ số 0) đúng vị trí
- Khi k = 2, phần tử thứ hai (chỉ số 1) đúng vị trí…



Bài toán sắp xếp tăng →Thuật toán lựa chọn



Bài toán sắp xếp tăng →Thuật toán lựa chọn

```
//Khai báo các biến
  int A[100], N;//Mảng chứa dữ liệu
//Sắp xếp
  for(int i = 0; i < N – 1; i++)
    for(int j = i + 1; j < N; j++)
      if(A[i] > A[j]) {
          int tmp = A[i];
          A[i] = A[j];
          A[j] = tmp;
      }
```

Ví dụ

- Nhập vào từ bàn phím một mảng các số nguyên không quá 100 phần tử
- Hiển thị dãy số vừa nhập
- Sắp xếp dãy theo thứ tự tăng dần bằng thuật toán chọn
- Hiển thị dãy tại mỗi lượt sắp xếp

```
for(int i = 0; i < N - 1; i++)
   for(int j = i + 1; j < N; j++)
      if(A[i] > A[j]) {
         int tmp = A[i];
         A[i] = A[j];
         A[j] = tmp;
      }
```

```
So phan tu (0<N<100), N = 5
Hay nhap day so...
A[0] = 3
A[1] = 2
A[2] = 4
A[3] = 6
A[4] = 1

Day vua nhap: 3 2 4 6 1
Sap xep day theo thuat toan lua chon:
Luot 1: 1 3 4 6 2
Luot 2: 1 2 4 6 3
Luot 3: 1 2 3 6 4
Luot 4: 1 2 3 4 6
```

Ví dụ

```
using namespace std;
int A[100], N;
void nhap()
{
   cout<<"So phan tu (0<N<100), N = "; cin>>N;
   cout<<"Hay nhap day so...\n";
   for(int i=0; i < N; i++){
      cout<<"A["<<i<<"] = "; cin>>A[i];
   }
}
void xuat()
{
   cout<<"\nDay vua nhap:";
   for(int i=0; i < N; i++)
      cout<<" "<<A[i];
}
```

Ví dụ

```
void sap_xep_chon()
{  cout<<"\nSap xep day theo thuat toan lua chon:";
   for(int i=0; i < N-1; i++){
      for(int j = i+1; j < N; j++)
         if(A[i] > A[j]) {
            int t = A[i];
            A[i] = A[j];
            A[j] = t; }
      cout<<"\nLuot "<<i+1<<":";
      for(int j=0; j < N; j++)
         cout<<" "<<A[j];
   }
}
int main(){
   nhap(); xuat();
   sap_xep_chon();
   return 0;
}
```

```
So phan tu (0<N<100), N = 5
Hay nhap day so...
A[0] = 3
A[1] = 2
A[2] = 4
A[3] = 6
A[4] = 1

Day vua nhap: 3 2 4 6 1
Sap xep day theo thuat toan lua chon:
Luot 1: 1 3 4 6 2
Luot 2: 1 2 4 6 3
Luot 3: 1 2 3 6 4
Luot 4: 1 2 3 4 6
```

Ví dụ sắp xếp giảm theo tt lựa chọn

```
void sap_xep_chon()
{  cout<<"\nSap xep day giam dan theo thuat toan lua chon:";
   for(int i=0; i < N-1; i++){
      for(int j = i+1; j < N; j++)
         if(A[i] < A[j]) {
            int t = A[i];
            A[i] = A[j];
            A[j] = t; }
      cout<<"\nLuot "<<i+1<<":";
      for(int j=0; j < N; j++)
         cout<<" "<<A[j];
   }
}int main(){
   nhap();
   xuat();
   sap_xep_chon();
   return 0;
}
```

```
So phan tu (0<N<100), N = 5
Hay nhap day so...
A[0] = 1
A[1] = 4
A[2] = 7
A[3] = 2
A[4] = 3

Day vua nhap: 1 4 7 2 3
Sap xep day giam dan theo thuat toan lua chon:
Luot 1: 7 1 4 2 3
Luot 2: 7 4 1 2 3
Luot 3: 7 4 3 1 2
Luot 4: 7 4 3 2 1
```

Thuật toán Sắp xếp nổi bọt (Bubble Sort)

Thuật toán:

1. Gán i = 0
2. Gán j = 0
3. Nếu A[j] > A[j + 1] thì đổi chỗ A[j] và A[j + 1]
4. Nếu j < n – i – 1:
   a. Đúng thì j = j + 1 và quay lại bước 3
   b. Sai thì sang bước 5
5. Nếu i < n – 1:
   a. Đúng thì i = i + 1 và quay lại bước 2
   b. Sai thì dừng lại

Thuật toán Sắp xếp nổi bọt (Bubble Sort)

```
void BubbleSort(int A[], int n)
{
    for (int i = 0; i < n - 1; i++)
        for (int j = 0; j < n - i - 1; j++)
            if (A[j] > A[j + 1])
                swap(A[j], A[j + 1]);
}
```

Ví dụ: Viết chương trình thực hiện nhập vào 1 dãy số nguyên gồm n phần tử. In ra dãy đã sắp xếp tăng dần theo thuật toán Bubble Sort.

Ví dụ về sắp xếp nổi bọt (Bubble Sort)

Viết chương trình thực hiện:

- Nhập vào 1 dãy số nguyên gồm n phần tử.
- In ra dãy đã nhập
- In ra dãy đã sắp xếp tăng dần theo thuật toán Bubble Sort.

```
So phan tu n = 5
Phan tu thu 1: 1
Phan tu thu 2: 2
Phan tu thu 3: 3
Phan tu thu 4: 7
Phan tu thu 5: 4
In day so: 1 2 3 7 4
In day so da sap xep: 1 2 3 4 7
--------------------------------
Process exited after 11.94 seconds with return value 0
Press any key to continue . . .
```

Ví dụ về sắp xếp nổi bọt (Bubble Sort)

```
#include <iostream>
using namespace std;

void nhap(int A[], int &n)
{
    cout<<"So phan tu n = "; cin>>n;
    for(int i=0;i<n;i++)
        {
            cout<<"Phan tu thu "<<i+1<<": ";
            cin>>A[i];
        }
}
void xuat(int A[], int n)
{
    for(int i=0;i<n;i++)
        cout<<" "<<A[i];
}
```

```
void swap(int &a, int &b) {
    int x = a;
    a = b;
    b = x;
}
void BubbleSort(int A[], int n)
{
    for (int i = 0; i < n - 1; i++)
        for (int j = 0; j < n - i - 1; j++)
            if (A[j] > A[j + 1])
                swap(A[j], A[j + 1]);
}
int main() {
    int n,A[100];
    nhap(A,n);
    cout<<"In day so:"; xuat(A,n);
    BubbleSort(A,n);
    cout<<"\nIn day so da sap xep:"; xuat(A,n);
    return 0;
}
```

Thuật toán Sắp xếp lựa chọn

- Tìm phần tử nhỏ nhất đưa vào vị trí 1
- Tìm phần tử nhỏ tiếp theo đưa vào vị trí 2
- Tìm phần tử nhỏ tiếp theo đưa vào vị trí 3
- ....

```
void SelectionSort(int A[], int n)
{
    int min;
    for (int i = 0; i < n - 1; i++)
    {
        min = i;
        for (int j = i + 1; j < n; j++)
            if (A[j] < A[min]) min = j;
        swap(A[i], A[min]);
    }
}
```

Thuật toán Sắp xếp lựa chọn

```
#include <iostream>
using namespace std;

void nhap(int A[], int &n)
{
    cout<<"So phan tu n = "; cin>>n;
    for(int i=0;i<n;i++)
        {
            cout<<"Phan tu thu "<<i+1<<": ";
            cin>>A[i];
        }
}
void xuat(int A[], int n)
{
    for(int i=0;i<n;i++)
        cout<<" "<<A[i];
}
```

```
void swap(int &a, int &b) {
    int x = a;
    a = b;
    b = x;
}
```

```
void SelectionSort(int A[], int n)
{
    int min;
    for (int i = 0; i < n - 1; i++)
    {
        min = i;
        for (int j = i + 1; j < n; j++)
            if (A[j] < A[min]) min = j;
        swap(A[i], A[min]);
    }
}
int main() {
    int n,A[100];
    nhap(A,n);
    cout<<"In day so:"; xuat(A,n);
    SelectionSort(A,n);
    cout<<"\nIn day so da sap xep:"; xuat(A,n);
    return 0;
}
```

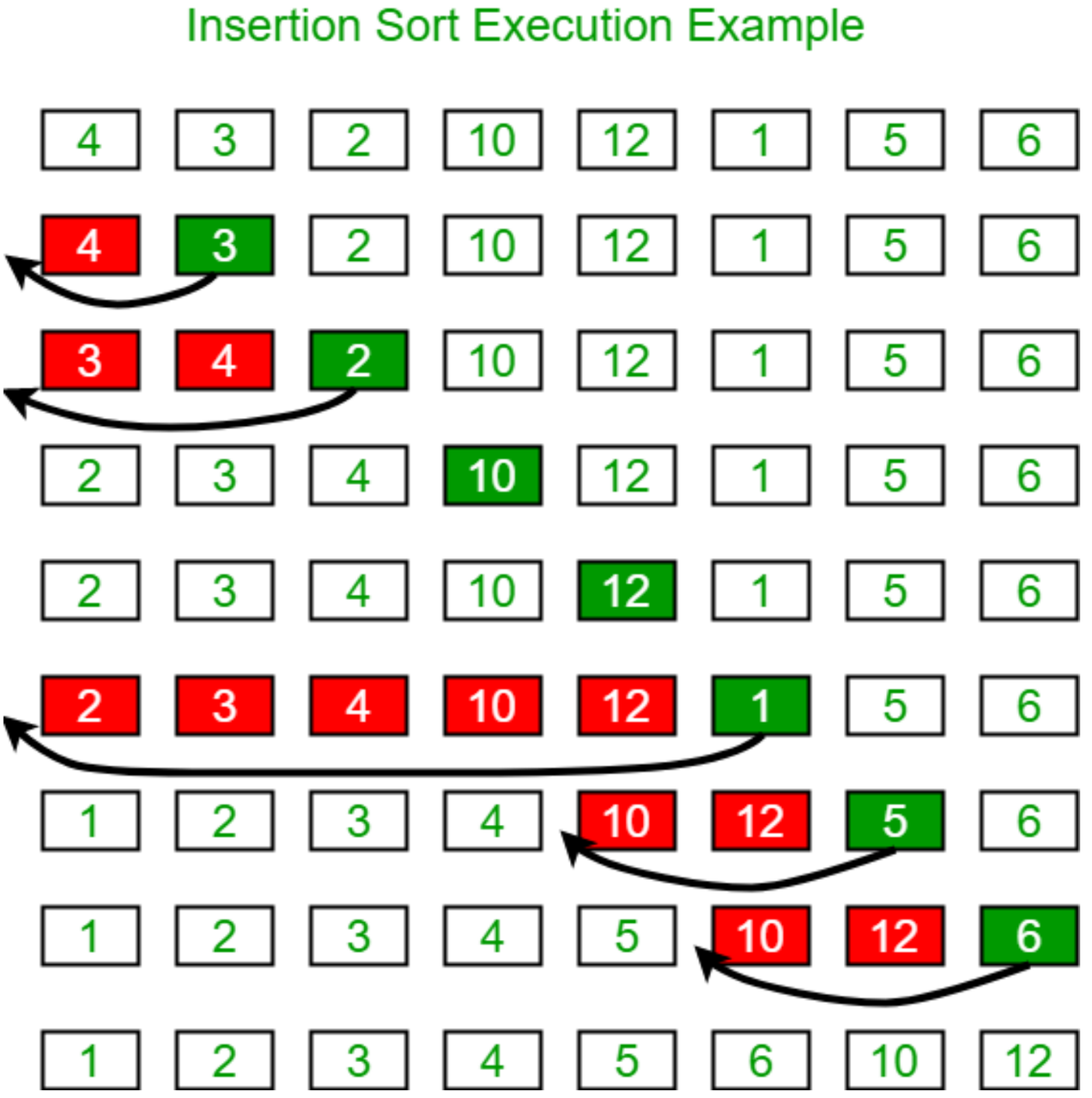
Thuật toán sắp xếp chèn (Insertion sort)

Thuật toán:

- Tại bước k =1,2,..,n đưa phần tử thứ k trong mảng đã cho vào đúng vị trí trong dãy gồm k phần tử đầu tiên.
- Kết quả là sau bước k, k phần tử đầu tiên đã được sắp theo thứ tự

```
void InsertionSort(int A[], int n)
{
    int key, j;
    for(int i = 1; i<n; i++) {
        key = A[i];
        j = i;
        while(j > 0 && A[j-1]>key)
        {
            A[j] = A[j-1];
            j--;
        }
        A[j] = key;
    }
}
```

Thuật toán sắp xếp chèn (Insertion sort)



Thuật toán sắp xếp chèn (Insertion sort)

```cpp
#include <iostream>
using namespace std;
void nhap(int A[], int &n){
    cout<<"So phan tu n = "; cin>>n;
    for(int i=0;i<n;i++){
        cout<<"Phan tu thu "<<i+1<<": ";
        cin>>A[i];
    }
}
void xuat(int A[], int n){
    for(int i=0;i<n;i++)
        cout<<" "<<A[i];
}
void swap(int &a, int &b){
    int x = a;
    a = b;
    b = x;
}
```

```
So phan tu n = 8
Phan tu thu 1: 4
Phan tu thu 2: 3
Phan tu thu 3: 2
Phan tu thu 4: 10
Phan tu thu 5: 12
Phan tu thu 6: 1
Phan tu thu 7: 5
Phan tu thu 8: 6
In day so: 4 3 2 10 12 1 5 6
Luot 1: 3 4 2 10 12 1 5 6
Luot 2: 2 3 4 10 12 1 5 6
Luot 3: 2 3 4 10 12 1 5 6
Luot 4: 2 3 4 10 12 1 5 6
Luot 5: 1 2 3 4 10 12 5 6
Luot 6: 1 2 3 4 5 10 12 6
Luot 7: 1 2 3 4 5 6 10 12
In day so da sap xep: 1 2 3 4 5 6 10 12
```

```cpp
void InsertionSort(int A[], int n){
   int key, j;
   for(int i = 1; i<n; i++) {
        key = A[i];
        j = i;
        while(j > 0 && A[j-1]>key){
            A[j] = A[j-1];
            j--;
        }
        A[j] = key;
        cout<<"\nLuot "<<i<<":"; xuat(A,n);
    }
}
int main(){
    int n,A[100];
    nhap(A,n);
    cout<<"In day so:"; xuat(A,n);
    InsertionSort(A,n);
    cout<<"\nIn day so da sap xep:"; xuat(A,n);
    return 0;
}
```

Thuật toán sắp xếp Nhanh (Quick Sort)

Sắp xếp nhanh (quick sort) hay sắp xếp phân đoạn (Partition) là thuật toán sắp xếp dựa trên kỹ thuật chia để trị, cụ thể ý tưởng là: chọn một điểm làm chốt (gọi là pivot), sắp xếp mọi phần tử bên trái chốt đều nhỏ hơn chốt và mọi phần tử bên phải đều lớn hơn chốt, sau khi xong ta được 2 dãy con bên trái và bên phải, áp dụng tương tự cách sắp xếp này cho 2 dãy con vừa tìm được cho đến khi dãy con chỉ còn 1 phần tử.

Cụ thể áp dụng thuật toán cho mảng như sau:

1. Chọn một phần tử làm chốt
2. Sắp xếp phần tử bên trái nhỏ hơn chốt
3. Sắp xếp phần tử bên phải nhỏ hơn chốt
4. Sắp xếp hai mảng con bên trái và bên phải chốt

Thuật toán sắp xếp Nhanh (Quick Sort)

- Phần tử được chọn làm chốt rất quan trọng, nó quyết định thời gian thực thi của thuật toán.
- Phần tử được chọn làm chốt tối ưu nhất là phần tử trung vị, phần tử này làm cho số phần tử nhỏ hơn trong dãy bằng hoặc xấp xỉ số phần tử lớn hơn trong dãy.
- Tuy nhiên, việc tìm phần tử này rất tốn kém, phải có thuật toán tìm riêng, từ đó làm giảm hiệu suất của thuật toán tìm kiếm nhanh, do đó, để đơn giản, người ta thường sử dụng phần tử chính giữa làm chốt.

Thuật toán sắp xếp nhanh (QuickSort)

```
void Partition(int A[], int left, int right)
{
    if (left < right)
    {
        int pivot = A[(left + right) / 2];
        int i = left, j = right;
        while (i < j)
        {
            while (A[i] < pivot)    i++;
            while (A[j] > pivot)    j--;
            if (i <= j)
            {
                if (i < j) swap(A[i], A[j]);
                i++;
                j--;
            }
        }
        Partition(A, left, j);
        Partition(A, i, right);
    }
}
```

```
void QuickSort(int A[], int n)
{
    Partition(A, 0, n - 1);
}
int main() {
    int n,A[100];
    nhap(A,n);
    cout<<"In day so:"; xuat(A,n);
    QuickSort(A,n);
    cout<<"\nIn day so da sap xep:"; xuat(A,n);
    return 0;
}
```

Bài toán Tìm kiếm

- Tìm kiếm tuần tự
- Tìm kiếm nhị phân

Tìm kiếm tuần tự (Linear Search)

Tìm kiếm tuần tự được thực hiện từ ý tưởng trực tiếp sau đây: Bắt đầu từ phần tử đầu tiên, duyệt qua từng phần tử cho đến khi tìm được đích hoặc kết luận không tìm được.

```
int LinearSearch(int A[], int n, int x)
{
    for(int i = 0; i<n;i++)
     if(A[i] == x)
        return i;
    return -1;
}
```

Thuật toán Tìm kiếm tuần tự (Linear Search)

Ví dụ: Hãy nhập vào một dãy số nguyên gồm n phần tử và số nguyên x. Hãy tìm kiếm xem x có trong dãy hay không bằng thuật toán tìm kiếm tuần tự (trả về vị trí đầu tiên thấy x, nếu không thấy trả về -1)

```
Nhap so phan tu cua day, n = 5
A[0] = 4
A[1] = 3
A[2] = 2
A[3] = 5
A[4] = 1
Cho phan tu can tim x = 3
So 3 o vi tri thu 1 trong day!
--------------------------------
Process exited after 10.38 seconds with return value 0
Press any key to continue . . .
```

Thuật toán Tìm kiếm tuần tự (Linear Search)

```
#include <iostream>
using namespace std;

void nhap(int A[], int n)
{
    for(int i = 0; i<n; i++)
    {
        cout<<"A["<<i<<"] = "; cin>>A[i];
    }
}
int LinearSearch(int A[], int n, int x)
{
    for(int i = 0; i<n;i++)
        if(A[i] == x)
            return i;
    return -1;
}


int main()
{
    int n, x;
    cout<<"Nhap so phan tu cua day, n = "; cin>>n;
    int a[n];
    nhap(a,n);
    cout<<"Cho phan tu can tim x = "; cin >> x;
    cout<<"So "<<x<< " o vi tri thu " <<LinearSearch(a,n,x)<<" trong day!";
    return 0;
}
```

```
Nhap so phan tu cua day, n = 5
A[0] = 2
A[1] = 5
A[2] = 3
A[3] = 7
A[4] = 6
Cho phan tu can tim x = 10
So 10 o vi tri thu -1 trong day!
```

```
Nhap so phan tu cua day, n = 5
A[0] = 4
A[1] = 3
A[2] = 2
A[3] = 5
A[4] = 6
Cho phan tu can tim x = 5
So 5 o vi tri thu 3 trong day!
```

Tìm kiếm nhị phân (Binary Search)

Tìm kiếm nhị phân (Binary search) hay còn một số tên gọi khác nữa như tìm kiếm nửa khoảng (half-interval search), tìm kiếm logarit (logarithmic search), chặt nhị phân (binary chop) là thuật toán tìm kiếm dựa trên việc chia đôi khoảng đang xét sau mỗi lần lặp, sau đó xét tiếp trong nửa khoảng có khả năng chứa giá trị cần tìm, cứ như vậy cho đến khi không chia đôi khoảng được nữa. Thuật toán tìm kiếm nhị phân **chỉ áp dụng được cho danh sách đã có thứ tự hay đã được sắp xếp.**

Thuật toán Tìm kiếm nhị phân (Binary Search)

```
int BinarySearch(int A[], int n, int x)
{
    int left = 0, right = n – 1, mid;
    while (left <= right)
    {
        mid = (left + right) / 2;
        if (A[mid] == x)
            return mid;
        if (A[mid] > x)
            right = mid - 1;
        else if (A[mid] < x)
            left = mid + 1;
    }
    return -1;
}
```

Ví dụ về Tìm kiếm nhị phân (Binary Search)

Ví dụ: Hãy nhập vào một dãy số nguyên gồm n phần tử và số nguyên x.
- Hãy sắp xếp dãy số theo thứ tự tăng dần
- Hãy tìm kiếm xem x có trong dãy hay không bằng thuật toán tìm kiếm nhị phân!

```
nhap so phan tu cua day n = 5
A[0] = 1
A[1] = 2
A[2] = 3
A[3] = 4
A[4] = 5
Cho x = 3
3 co trong day!
--------------------------------
Process exited after 11.27 seconds wi
Press any key to continue . . .
```

Ví dụ về Tìm kiếm nhị phân (Binary Search)

```
#include <iostream>
using namespace std;
void Nhap(int A[], int n)
{
    for(int i =0; i<n; i++)
    {
        cout<<"A["<<i<<"] = "; cin>>A[i];
    }
}
void Sort(int a[], int n)
{
    for (int i=0;i<n-1;i++)
     for(int j=i+1;j<n;j++)
        if(a[i]>a[j])
            {
                int tam = a[i];
                a[i] = a[j];
                a[j] = tam;
            }
}
```

```
int BinarySearch(int A[], int n, int x)
{
    int left = 0, right = n - 1, mid;
    while (left <= right)
    {
        mid = (left + right) / 2;
        if (A[mid] == x)
            return mid;
        if (A[mid] > x)
            right = mid - 1;
        else if (A[mid] < x)
            left = mid + 1;
    }
    return -1;
}
```

```
int main()
{
    int n, A[100], x;
    cout<<"nhap so phan tu cua day n = "; cin>>n;
    Nhap(A,n); Sort(A,n);
    cout<<"Cho x = "; cin >> x;
    if(BinarySearch(A,n,x))
        cout<<x<< " co trong day!";
    else
        cout<<x<<" khong co trong day!";
    return 0;
}
```

Bài tập 1

1. Nhập vào từ bàn phím một dãy số nguyên (<100 phần tử). Sắp xếp dãy theo nguyên tắc: Bên trên là số chẵn chia hết cho 3. Bên dưới là số lẻ chia hết cho 3. Giữa là các số còn lại. Đưa cả 2 dãy ra màn hình.
2. Đọc vào dãy số có n phần từ (n<100). Đọc số x và số k nguyên. Chèn x vào vị trí k của dãy. Nếu k>n, chèn x vào vị trí n+1.
3. Nhập vào một dãy số (<100 phần tử) và sắp xếp theo thứ tự tăng dần. Nhập thêm vào một số và chèn số mới nhập vào đúng vị trí
4. Nhập vào một dãy (<100 phần tử); xóa đi các phần tử chia hết cho 5 và đưa kết quả ra màn hình

Bài tập 2: Ma trận

1. Viết chương trình nhập vào một ma trận vuông, các phần tử nguyên, sau đó
   - Đưa ra ma trận tam giác dưới
   - Đưa ra ma trận tam giác trên
2. Nhập M, N (M, N < 30) và một ma trận MxN. Đưa ma trận ra màn hình
   - Tìm hàng/cột có tổng các phần tử lớn nhất
   - Tìm số lớn nhất/nhỏ nhất và vị trí trong ma trận
   - Đưa ra ma trận S cùng kích thước thỏa mãn

$$s_{i,j} = \begin{cases} 1 & \text{nếu } u_{i,j} > 0 \\ 0 & \text{nếu } u_{i,j} = 0 \\ -1 & \text{nếu } u_{i,j} < 0 \end{cases}$$