



THUẬT TOÁN INTERCHANGE SORT

1. Hồ Thái Ngọc
2. ThS. Võ Duy Nguyên
3. TS. Nguyễn Tấn Trần Minh Khang



BÀI TOÁN DẪN NHẬP

Bài toán dẫn nhập



— Bài toán: Viết hàm liệt kê tất cả các cặp giá trị trong mảng một chiều các số nguyên. Lưu ý: cặp (1,2) và cặp (2,1) là giống nhau.

— Ví dụ:

| | | | | |
|----|----|---|----|----|
| 12 | 43 | 1 | 34 | 22 |
|----|----|---|----|----|

— Các cặp giá trị trong mảng là:

+ (12,43),(12,01),(12,34),(12,22)

+ (43,01), (43,34), (43,22)

+ (01,34), (01,22)

+ (34,22)

Bài toán dẫn nhập



```
11. void LietKe(int a[], int n)
12. {
13.     for(int i=0; i<=n-2; i++)
14.     {
15.         for(int j=i+1; j<=n-1; j++)
16.         {
17.             cout << "(" << a[i] << "," << a[j] << ")";
18.         }
19.     }
20. }
```

Bài toán dẫn nhập



```
11. void LietKe(int a[], int n)
12. {
13.     for(int i=0; i<=n-2; i++)
14.         for(int j=i+1; j<=n-1; j++)
15.             cout << "(" << a[i] << "," << a[j] << ")";
16. }
```



BÀI TOÁN LIÊN QUAN THUẬT TOÁN

Bài toán liên quan



— Bài toán: Định nghĩa hàm đếm số lượng cặp giá trị trong mảng một chiều các số nguyên có n phần tử.

— Ví dụ:

| | | | | |
|----|----|---|----|----|
| 12 | 43 | 1 | 34 | 22 |
|----|----|---|----|----|

— Các cặp giá trị trong mảng là: (12,43),(12,01),(12,34),(12,22), (43,01), (43,34), (43,22), (01,34), (01,22), (34,22).

— Kết quả: 10.

Bài toán liên quan



```
11. int DemSoCap(int a[], int n)
12. {
13.     int dem = 0;
14.     for(int i=0; i<=n-2; i++)
15.         for(int j=i+1; j<=n-1; j++)
16.             dem++;
17.     return dem;
18. }
```


Bài toán liên quan



```
11.int DemSoCap(int a[], int n)
12.{
13.|    return  $n * (n - 1) / 2$ ;
14.}
```

Bài toán liên quan



```
11. int DemSoCap(int a[], int n)
12. {
13.     if(n%2==0)
14.         return n/2*(n-1);
15.     return (n-1)/2*n;
16. }
```



NGHỊCH THỂ

Nghịch thế



- Khái niệm: Một cặp giá trị (a_i, a_j) được gọi là nghịch thế khi a_i và a_j không thỏa điều kiện sắp thứ tự.
- Ví dụ 1: Cho mảng một chiều các số thực a có n phần tử: $a_0, a_1, a_2, \dots, a_{n-2}, a_{n-1}$. Hãy sắp mảng theo thứ tự tăng dần. Khi đó cặp giá trị (a_i, a_j) ($i < j$) được gọi là nghịch thế khi $a_i \geq a_j$
- Ví dụ 2: Cho mảng một chiều các số thực a có n phần tử: $a_0, a_1, a_2, \dots, a_{n-2}, a_{n-1}$. Hãy sắp mảng theo thứ tự giảm dần. Khi đó cặp giá trị (a_i, a_j) ($i < j$) được gọi là nghịch thế khi $a_i \leq a_j$

Nghịch thế



- Ví dụ 3: Hãy liệt kê các cặp giá trị nghịch thế trong mảng sau, biết rằng yêu cầu là sắp xếp mảng tăng dần.

| | | | | | |
|----|----|----|----|---|----|
| 14 | 29 | -1 | 10 | 5 | 23 |
|----|----|----|----|---|----|

- Kết quả:

+ (14, -1), (14,10), (14,5)

+ (29,-1), (29,10), (29,5), (29,23)

+ (10,5)

Nghịch thế



— Hãy định nghĩa hàm liệt kê các cặp giá trị nghịch thế trong mảng một chiều số nguyên, biết rằng yêu cầu là sắp xếp mảng tăng dần.

```
11. void LietKe(int a[], int n)
12. {
13.     for(int i=0; i<=n-2; i++)
14.         for(int j=i+1; j<=n-1; j++)
15.             if(a[i]>a[j])
16.                 cout<<"("<<a[i]<<","<<a[j]<<")";
17. }
```



TƯ TƯỞNG THUẬT TOÁN

Tư tưởng thuật toán



- Thuật toán interchange sort sẽ duyệt qua tất cả các cặp giá trị trong mảng và hoán vị hai giá trị trong một cặp nếu cặp giá trị đó là nghịch thế.



HÀM CÀI ĐẶT CHUẨN

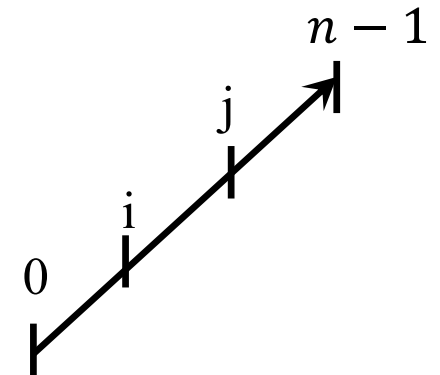
Hàm cài đặt chuẩn



— Bài toán: Định nghĩa hàm sắp mảng một chiều các số nguyên tăng dần bằng thuật toán Interchange sort.

— Hàm cài đặt

```
11. void HoanVi(int& a, int& b)
12. {
13.     int temp = a;
14.     a = b;
15.     b = temp;
16. }
```

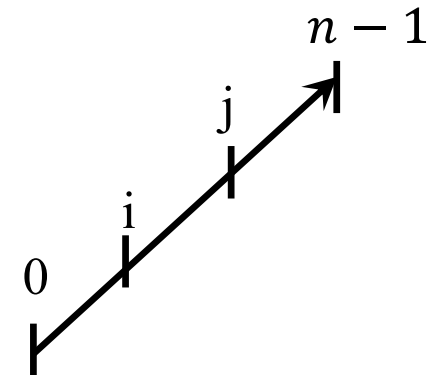


Hàm cài đặt chuẩn



— Hàm cài đặt

```
11. void InterchangeSort(int a[], int n)
12. {
13.     for(int i=0; i<=n-2; i++)
14.         for(int j=i+1; j<=n-1; j++)
15.             if(a[i]>a[j])
16.                 HoanVi(a[i],a[j]);
17. }
```

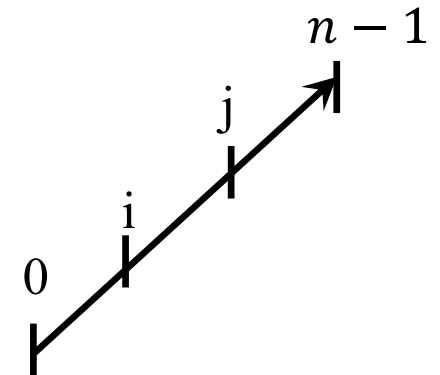


Hàm cài đặt chuẩn



— Hàm cài đặt

```
11. void InterchangeSort(int a[], int n)
12. {
13.     for(int i=0; i<=n-2; i++)
14.         for(int j=i+1; j<=n-1; j++)
15.             if(a[i]>a[j])
16.                 swap(a[i],a[j]);
17. }
```





CHẠY TỪNG BƯỚC THUẬT TOÁN

Chạy từng bước thuật toán



— Hãy sắp xếp mảng sau tăng dần:

| | | | | | |
|----|----|----|----|----|----|
| 24 | 45 | 23 | 13 | 43 | -1 |
|----|----|----|----|----|----|

— Thứ tự các bước khi sắp tăng dần mảng trên bằng thuật toán interchange sort.

Chạy từng bước thuật toán



— Bước 01: Xét phần tử đầu tiên.

| | | | | | |
|----|----|----|----|----|----|
| 24 | 45 | 23 | 13 | 43 | -1 |
|----|----|----|----|----|----|

| | | | | | |
|----|----|----|----|----|----|
| 24 | 45 | 23 | 13 | 43 | -1 |
|----|----|----|----|----|----|

| | | | | | |
|----|----|----|----|----|----|
| 23 | 45 | 24 | 13 | 43 | -1 |
|----|----|----|----|----|----|

| | | | | | |
|----|----|----|----|----|----|
| 13 | 45 | 24 | 23 | 43 | -1 |
|----|----|----|----|----|----|

| | | | | | |
|----|----|----|----|----|----|
| 13 | 45 | 24 | 23 | 43 | -1 |
|----|----|----|----|----|----|

| | | | | | |
|----|----|----|----|----|----|
| -1 | 45 | 24 | 23 | 43 | 13 |
|----|----|----|----|----|----|

| | | | | | |
|----|----|----|----|----|----|
| 23 | 45 | 24 | 13 | 43 | -1 |
|----|----|----|----|----|----|

| | | | | | |
|----|----|----|----|----|----|
| 13 | 45 | 24 | 23 | 43 | -1 |
|----|----|----|----|----|----|

| | | | | | |
|----|----|----|----|----|----|
| -1 | 45 | 24 | 23 | 43 | 13 |
|----|----|----|----|----|----|

Chạy từng bước thuật toán



— Bước 02: Xét phần tử thứ hai.

| | | | | | |
|----|----|----|----|----|----|
| -1 | 45 | 24 | 23 | 43 | 13 |
|----|----|----|----|----|----|

| | | | | | |
|----|----|----|----|----|----|
| -1 | 24 | 45 | 23 | 43 | 13 |
|----|----|----|----|----|----|

| | | | | | |
|----|----|----|----|----|----|
| -1 | 23 | 45 | 24 | 43 | 13 |
|----|----|----|----|----|----|

| | | | | | |
|----|----|----|----|----|----|
| -1 | 23 | 45 | 24 | 43 | 13 |
|----|----|----|----|----|----|

| | | | | | |
|----|----|----|----|----|----|
| -1 | 13 | 45 | 24 | 43 | 23 |
|----|----|----|----|----|----|

| | | | | | |
|----|----|----|----|----|----|
| -1 | 24 | 45 | 23 | 43 | 13 |
|----|----|----|----|----|----|

| | | | | | |
|----|----|----|----|----|----|
| -1 | 23 | 45 | 24 | 43 | 13 |
|----|----|----|----|----|----|

| | | | | | |
|----|----|----|----|----|----|
| -1 | 13 | 45 | 24 | 43 | 23 |
|----|----|----|----|----|----|

Chạy từng bước thuật toán



— Bước 03: Xét phần tử thứ ba.

| | | | | | |
|----|----|----|----|----|----|
| -1 | 13 | 45 | 24 | 43 | 23 |
|----|----|----|----|----|----|

| | | | | | |
|----|----|----|----|----|----|
| -1 | 13 | 24 | 45 | 43 | 23 |
|----|----|----|----|----|----|

| | | | | | |
|----|----|----|----|----|----|
| -1 | 13 | 24 | 45 | 43 | 23 |
|----|----|----|----|----|----|

| | | | | | |
|----|----|----|----|----|----|
| -1 | 13 | 23 | 45 | 43 | 24 |
|----|----|----|----|----|----|

| | | | | | |
|----|----|----|----|----|----|
| -1 | 13 | 24 | 45 | 43 | 23 |
|----|----|----|----|----|----|

| | | | | | |
|----|----|----|----|----|----|
| -1 | 13 | 23 | 45 | 43 | 24 |
|----|----|----|----|----|----|

Chạy từng bước thuật toán



— Bước 04: Xét phần tử thứ tư.

| | | | | | |
|----|----|----|----|----|----|
| -1 | 13 | 23 | 45 | 43 | 24 |
|----|----|----|----|----|----|

| | | | | | |
|----|----|----|----|----|----|
| -1 | 13 | 23 | 43 | 45 | 24 |
|----|----|----|----|----|----|

| | | | | | |
|----|----|----|----|----|----|
| -1 | 13 | 23 | 24 | 45 | 43 |
|----|----|----|----|----|----|

| | | | | | |
|----|----|----|----|----|----|
| -1 | 13 | 23 | 43 | 45 | 24 |
|----|----|----|----|----|----|

| | | | | | |
|----|----|----|----|----|----|
| -1 | 13 | 23 | 24 | 45 | 43 |
|----|----|----|----|----|----|

Chạy từng bước thuật toán



— Bước 05: Xét phần tử thứ năm.

| | | | | | |
|----|----|----|----|----|----|
| -1 | 13 | 23 | 24 | 45 | 43 |
|----|----|----|----|----|----|

| | | | | | |
|----|----|----|----|----|----|
| -1 | 13 | 23 | 24 | 43 | 45 |
|----|----|----|----|----|----|

| | | | | | |
|----|----|----|----|----|----|
| -1 | 13 | 23 | 24 | 43 | 45 |
|----|----|----|----|----|----|



Interchange sort và mảng một chiều

PROJECT G01 – DỰ ÁN G01

Interchange sort và mảng một chiều



— Viết chương trình thực hiện các yêu cầu sau:

- + Nhập mảng một chiều từ các tập tin: intdata01.inp; intdata02.inp; ...; intdata09.inp; intdata10.inp; intdata11.inp; intdata12.inp; intdata13.inp;
- + Sắp xếp mảng tăng dần bằng thuật toán interchange sort.
- + Xuất mảng sau khi sắp xếp ra các tập tin: intdata01.out; intdata02.out; ...; intdata09.out; intdata10.out; intdata11.out; intdata12.out; intdata13.out;

Interchange sort và mảng một chiều



- Định dạng tập tin `intdataxx.inp` và `intdataxx.out`
 - + Dòng đầu tiên: số phần tử của mảng (n).
 - + Dòng tiếp theo: lưu n số nguyên tương ứng với các giá trị trong mảng.

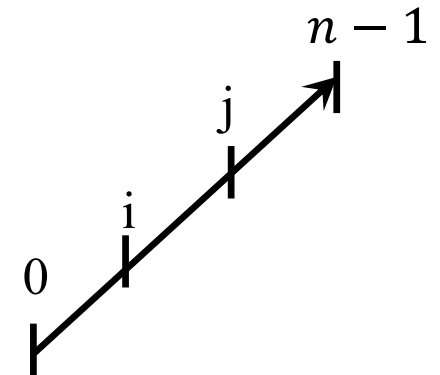
```
*intdata01.inp - Notepad
File Edit Format View Help
10
24 56 53 44 -54 6 63 -47 91 -99
```

Interchange sort và mảng một chiều



— Hàm cài đặt

```
11. void InterchangeSort(int a[], int n)
12. {
13.     for(int i=0; i<=n-2; i++)
14.         for(int j=i+1; j<=n-1; j++)
15.             if(a[i]>a[j])
16.                 swap(a[i], a[j]);
17. }
```



Interchange sort và mảng một chiều



```
11.int Nhap(int a[], int n, string filename)
```

```
12.{
```

A vertical line extends downwards from the opening curly brace of the function body.

Nhập mảng
một chiều số
nguyên từ
file

Interchange sort và mảng một chiều



```
11.int Nhap(int a[], int &n, string filename)
```

```
12.{
```

|

Nhập mảng
một chiều số
nguyên từ
file

Interchange sort và mảng một chiều



```
11.int Nhap(int a[], int &n, string filename)
```

```
12.{
```

```
13.    ifstream fi(filename);
```

Nhập mảng
một chiều số
nguyên từ
file

Interchange sort và mảng một chiều



```
11.int Nhap(int a[], int &n, string filename)
```

```
12.{
```

```
13.    ifstream fi(filename);
```

Ý1: fi là đối tượng thuộc lớp
ifstream.

Nhập mảng
một chiều số
nguyên từ
file

Interchange sort và mảng một chiều



```
11.int Nhap(int a[], int &n, string filename)
```

```
12.{
```

```
13.    ifstream fi(filename);
```

Ý2: Dòng lệnh số 13 khai báo đối tượng fi với đối số có tên filename và có kiểu string.

Nhập mảng
một chiều số
nguyên từ
file

Interchange sort và mảng một chiều



```
11. int Nhap(int a[], int &n, string filename)
12. {
13.     ifstream fi(filename);
```

Ý3: Khi chương trình thực hiện tới dòng lệnh số 13. Đối tượng `fi` gọi thực hiện phương thức thiết lập với tham số có kiểu `string`. Phương thức thiết lập sẽ mở tập tin có tên lưu trong biến `filename`.

Nhập mảng
một chiều số
nguyên từ
file

Interchange sort và mảng một chiều



```
11.int Nhap(int a[], int &n, string filename)
12.{
13.    ifstream fi(filename);
14.    if (fi.fail()==true)
15.        return 0;
16.    fi >> n;
17.    for(int i=0; i<n; i++)
18.        fi >> a[i];
19.    return 1;
20.}
```

Nhập mảng
một chiều số
nguyên từ
file

```
11.int Xuat(int a[],int n, string filename)
```

```
12.{
```

```
13.    ofstream fo(filename);
```

```
24.}
```



Xuất mảng
một chiều số
nguyên ra file

```
11.int Xuat(int a[],int n, string filename)
```

```
12.{
```

```
13.    ofstream fo(filename);
```

```
24.}
```



Xuất mảng
một chiều số
nguyên ra file


```
11.int Xuat(int a[],int n, string filename)
```

```
12.{
```

```
13.    ofstream fo(filename);
```

Ý1: fo là đối tượng thuộc lớp
ofstream.

```
24.}
```



Xuất mảng
một chiều số
nguyên ra file

```
11.int Xuat(int a[],int n, string filename)
```

```
12.{
```

```
13.    ofstream fo(filename);
```

Ý2: Dòng lệnh số 13 khai báo đối tượng fo với đối số có tên filename và có kiểu string.

```
24.}
```



Xuất mảng
một chiều số
nguyên ra file

```
11.int Xuat(int a[],int n, string filename)
```

```
12.{
```

```
13.    ofstream fo(filename);
```

Ý3: Khi chương trình thực hiện tới dòng lệnh số 13. Đối tượng fo gọi thực hiện phương thức thiết lập với tham số có kiểu string. Phương thức thiết lập sẽ mở tập tin có tên lưu trong biến filename.

```
24.}
```



Xuất mảng
một chiều số
nguyên ra file

```
11.int Xuat(int a[],int n, string filename)
12.{
13.    ofstream fo(filename);
14.    if (fo.fail()==true)
15.        return 0;
16.    fo << setw(5) << n << endl;
17.    int i = 0;
18.    while (i<n)
19.    {
20.        |    fo << setw(5) << a[i];
21.        |    i++;
22.    }
23.    return 1;
24.}
```



Xuất mảng
một chiều số
nguyên ra file

Interchange sort và mảng một chiều



```
11. int Xuat(int a[], int n, string filename)
12. {
13.     ofstream fo(filename);
14.     if (fo.fail() == true)
15.         return 0;
16.     fo << setw(5) << n << endl;
17.     for(int i = 0; i < n; i++)
18.         fo << setw(5) << a[i];
19.     return 1;
20. }
```

Xuất mảng
một chiều số
nguyên ra file



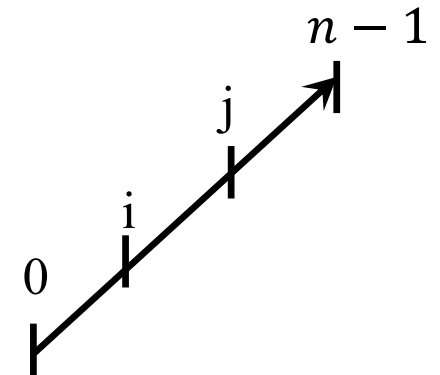
BIẾN THỂ CÀI ĐẶT 01

Hàm cài đặt chuẩn



— Hàm cài đặt

```
11. void InterchangeSort(int a[], int n)
12. {
13.     for(int i=0; i<=n-2; i++)
14.         for(int j=i+1; j<=n-1; j++)
15.             if(a[i]>a[j])
16.                 swap(a[i], a[j]);
17. }
```

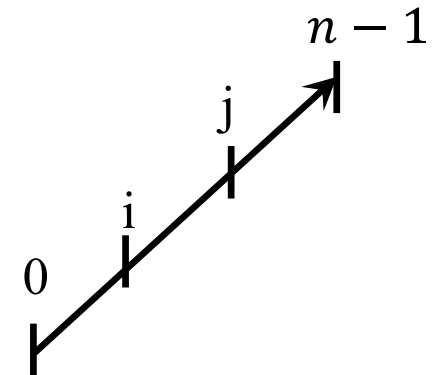


Biến thể cài đặt 01



— Bài toán: Định nghĩa hàm sắp mảng một chiều các số nguyên tăng dần bằng thuật toán Interchange sort.

```
11. void InterchangeSort01(int a[], int n)
12. {
13.     for(int i=0; i<=n-2; i++)
14.         for(int j=n-1; j>=i+1; j--)
15.             if(a[i]>a[j])
16.                 HoanVi(a[i], a[j]);
17. }
```





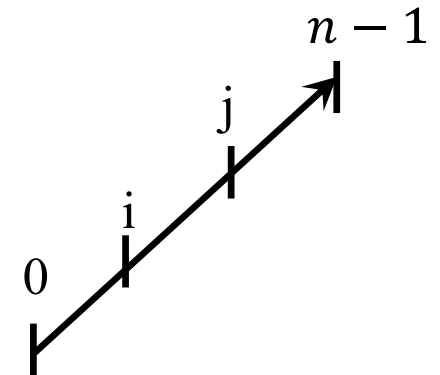
BIẾN THỂ CÀI ĐẶT 02

Hàm cài đặt chuẩn



— Hàm cài đặt

```
11. void InterchangeSort(int a[], int n)
12. {
13.     for(int i=0; i<=n-2; i++)
14.         for(int j=i+1; j<=n-1; j++)
15.             if(a[i]>a[j])
16.                 swap(a[i], a[j]);
17. }
```

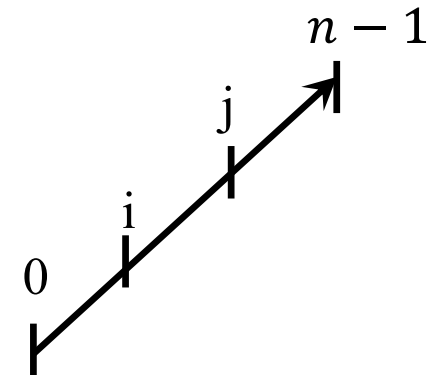


Biến thể cài đặt 02



— Bài toán: Định nghĩa hàm sắp mảng một chiều các số nguyên tăng dần bằng thuật toán Interchange sort.

```
11. void InterchangeSort02(int a[], int n)
12. {
13.     for(int i=n-1; i>=1; i--)
14.         for(int j=0; j<=i-1; j++)
15.             if(a[j]>a[i])
16.                 HoanVi(a[i], a[j]);
17. }
```





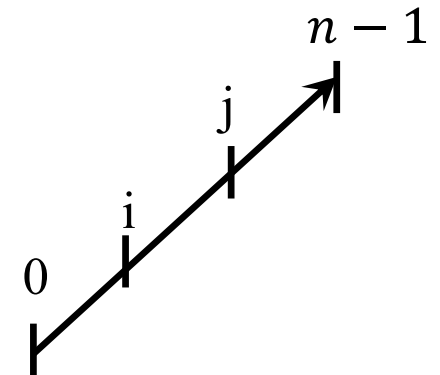
BIẾN THỂ CÀI ĐẶT 03

Biến thể cài đặt 03



— Bài toán: Định nghĩa hàm sắp mảng một chiều các số nguyên tăng dần bằng thuật toán Interchange sort.

```
11. void InterchangeSort03(int a[], int n)
12. {
13.     for(int i=n-1; i>=1; i--)
14.         for(int j=i-1; j>=0; j--)
15.             if(a[j]>a[i])
16.                 HoanVi(a[i], a[j]);
17. }
```





Interchange sort và mảng một chiều

PROJECT G02 – DỰ ÁN G02

Interchange sort và mảng một chiều



— Viết chương trình thực hiện các yêu cầu sau:

- + Nhập mảng một chiều từ các tập tin: intdata01.inp; intdata02.inp; ...; intdata09.inp; intdata10.inp; intdata11.inp; intdata12.inp; intdata13.inp;
- + Sắp xếp mảng tăng dần bằng thuật toán interchange sort với biến thể 1, 2, 3.
- + Xuất mảng sau khi sắp xếp ra các tập tin: intdata01.out; intdata02.out; ...; intdata09.out; intdata10.out; intdata11.out; intdata12.out; intdata13.out;

Interchange sort và mảng một chiều



- Định dạng tập tin `intdataxx.inp` và `intdataxx.out`
 - + Dòng đầu tiên: số phần tử của mảng (n).
 - + Dòng tiếp theo: lưu n số nguyên tương ứng với các giá trị trong mảng.

```
*intdata01.inp - Notepad
File Edit Format View Help
10
24 56 53 44 -54 6 63 -47 91 -99
```




Thuật toán interchange sort và mảng cấu trúc

MẢNG CẤU TRÚC

Interchange sort và mảng cấu trúc



- Bài toán: Định nghĩa hàm sắp mảng một chiều các phân số tăng dần bằng thuật toán Interchange sort.
- Khai báo kiểu dữ liệu biểu diễn phân số.

```
11.struct phanso
```

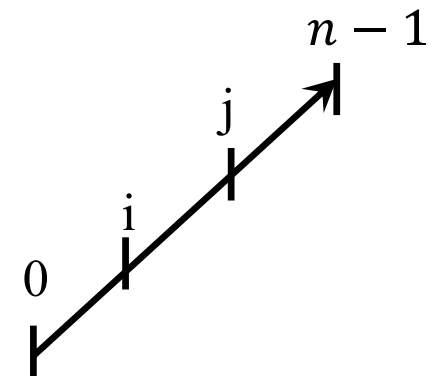
```
12.{
```

```
13. |    int tu;
```

```
14. |    int mau;
```

```
15.};
```

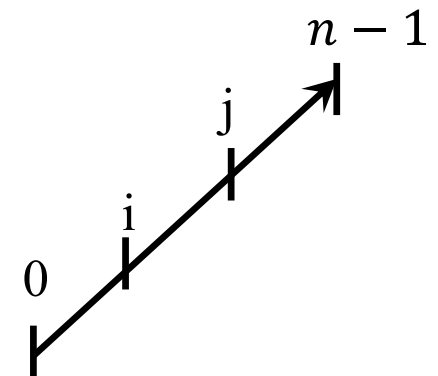
```
16.typedef struct phanso PHANSO;
```



Interchange sort và mảng cấu trúc



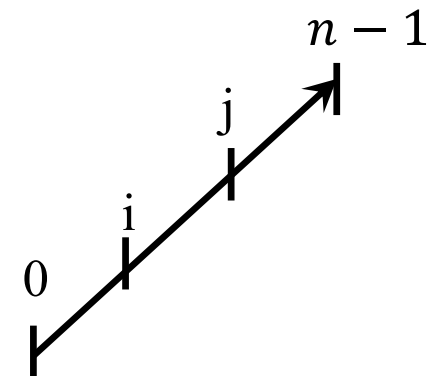
```
11. int SoSanh(PHANSO x, PHANSO y)
12. {
13.     float a = (float)x.tu/x.mau;
14.     float b = (float)y.tu/y.mau;
15.     if(a>b)
16.         return 1;
17.     if(a<b)
18.         return -1;
19.     return 0;
20. }
```



Interchange sort và mảng cấu trúc



```
11. void InterchangeSort(PHANSO a[], int n)
12. {
13.     for(int i=0; i<=n-2; i++)
14.         for(int j=i+1; j<=n-1; j++)
15.             if(SoSanh(a[i], a[j]) == 1)
16.             {
17.                 PHANSO temp = a[i];
18.                 a[i] = a[j];
19.                 a[j] = temp;
20.             }
21. }
```





PROJECT G03 – DỰ ÁN G03

Interchange sort và mảng cấu trúc



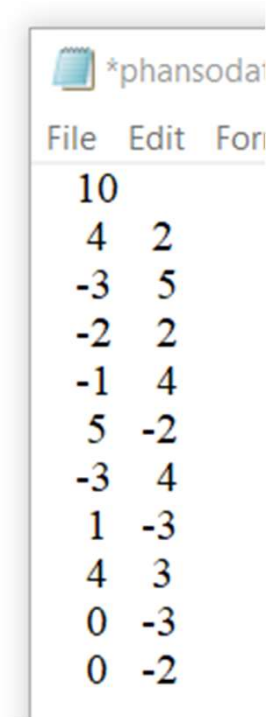
— Viết chương trình thực hiện các yêu cầu sau:

- + Nhập mảng một chiều từ các tập tin: phansodata01.inp; phansodata02.inp; ...; phansodata09.inp; phansodata10.inp; phansodata11.inp; phansodata12.inp; phansodata13.inp;
- + Sắp xếp mảng phân số tăng dần bằng thuật toán Interchange sort.
- + Xuất mảng sau khi sắp tăng ra các tập tin: phansodata01.out; phansodata02.out; ...; phansodata09.out; phansodata10.out; phansodata11.out; phansodata12.out; phansodata13.out;

Interchange sort và mảng cấu trúc



- Định dạng tập tin
 - + phansodataxx.inp,
 - + phansodataxx.out
- Dòng đầu tiên: số phần tử của mảng (n).
- n dòng tiếp theo: mỗi dòng lưu hai số nguyên tương ứng với phân số trong mảng.

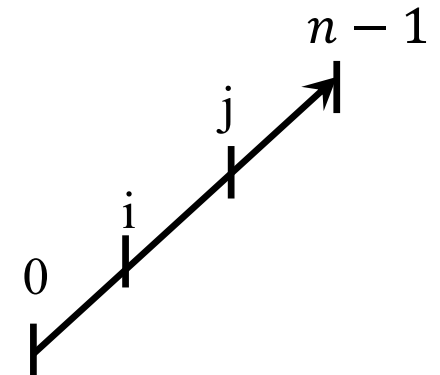


| | |
|----|----|
| 10 | |
| 4 | 2 |
| -3 | 5 |
| -2 | 2 |
| -1 | 4 |
| 5 | -2 |
| -3 | 4 |
| 1 | -3 |
| 4 | 3 |
| 0 | -3 |
| 0 | -2 |

Interchange sort và mảng cấu trúc



```
11. void InterchangeSort(PHANSO a[], int n)
12. {
13.     for(int i=0; i<=n-2; i++)
14.         for(int j=i+1; j<=n-1; j++)
15.             if(SoSanh(a[i], a[j]) == 1)
16.             {
17.                 PHANSO temp = a[i];
18.                 a[i] = a[j];
19.                 a[j] = temp;
20.             }
21. }
```



Interchange sort và mảng một chiều



```
11.int Nhap(PHANSO a[], int n, string filename)
```

```
12.{
```

|

Nhập mảng
một chiều
phân số từ
file

Interchange sort và mảng cấu trúc



```
11.int Nhap(PHANSO a[], int &n, string filename)
```

```
12.{
```

|

Nhập mảng
một chiều
phân số từ
file

Interchange sort và mảng cấu trúc



```
11. int Nhap(PHANSO a[], int &n, string filename)
```

```
12. {
```

```
13.     ifstream fi(filename);
```

Nhập mảng
một chiều
phân số từ
file

Interchange sort và mảng cấu trúc



```
11. int Nhap(PHANSO a[], int &n, string filename)
```

```
12. {
```

```
13.     ifstream fi(filename);
```

Ý1: fi là đối tượng thuộc lớp
ifstream.

Nhập mảng
một chiều
phân số từ
file

Interchange sort và mảng cấu trúc



```
11. int Nhap(PHANSO a[], int &n, string filename)
12. {
13.     ifstream fi(filename);
```

Ý2: Dòng lệnh số 13 khai báo đối tượng fi với đối số có tên filename và có kiểu string.

Nhập mảng
một chiều
phân số từ
file

Interchange sort và mảng cấu trúc



```
11. int Nhap(PHANSO a[], int &n, string filename)
12. {
13.     ifstream fi(filename);
```

Ý3: Khi chương trình thực hiện tới dòng lệnh số 13. Đối tượng `fi` gọi thực hiện phương thức thiết lập với tham số có kiểu `string`. Phương thức thiết lập sẽ mở tập tin có tên lưu trong biến `filename`.

Nhập mảng
một chiều
phân số từ
file

Interchange sort và mảng cấu trúc



```
11. int Nhap(PHANSO a[], int &n, string filename)
12. {
13.     ifstream fi(filename);
14.     if (fi.fail()==true)
15.         return 0;
16.     fi >> n;
17.     for(int i=0; i<n; i++)
18.         fi >> a[i].tu >> a[i].mau;
19.     return 1;
20. }
```

Nhập mảng
một chiều
phân số từ
file

```
11.int Xuat(PHANSO a[],int n, string filename)
```

```
12.{
```

```
13.    ofstream fo(filename);
```

```
24.}
```



Xuất mảng
một chiều
phân số ra
file


```
11.int Xuat(PHANSO a[],int n, string filename)
```

```
12.{
```

```
13.    ofstream fo(filename);
```

```
24.}
```



Xuất mảng
một chiều
phân số ra
file

```
11.int Xuat(PHANSO a[],int n, string filename)
```

```
12.{
```

```
13.    ofstream fo(filename);
```

Ý1: fo là đối tượng thuộc lớp
ofstream.

```
24.}
```



Xuất mảng
một chiều
phân số ra
file

```
11.int Xuat(PHANSO a[],int n, string filename)
```

```
12.{
```

```
13.    ofstream fo(filename);
```

Ý2: Dòng lệnh số 13 khai báo đối tượng fo với đối số có tên filename và có kiểu string.

```
24.}
```



Xuất mảng
một chiều
phân số ra
file

```
11.int Xuat(PHANSO a[],int n, string filename)
```

```
12.{
```

```
13.    ofstream fo(filename);
```

Ý3: Khi chương trình thực hiện tới dòng lệnh số 13. Đối tượng fo gọi thực hiện phương thức thiết lập với tham số có kiểu string. Phương thức thiết lập sẽ mở tập tin có tên lưu trong biến filename.

```
24.}
```



Xuất mảng
một chiều
phân số ra
file

```
11.int Xuat(PHANSO a[],int n, string filename)
12.{
13.    ofstream fo(filename);
14.    if (fo.fail()==true)
15.        return 0;
16.    fo << setw(5) << n << endl;
17.    int i = 0;
18.    while (i<n)
19.    {
20.        fo<<setw(5)<<a[i].tu<<setw(5)<<a[i].mau<<endl;
21.        i++;
22.    }
23.    return 1;
24.}
```



Xuất mảng một
chiều phân số
ra file

Interchange sort và mảng cấu trúc



```
11. int Xuat(PHANSO a[], int n, string filename)
12. {
13.     ofstream fo(filename);
14.     if (fo.fail() == true)
15.         return 0;
16.     fo << setw(5) << n << endl;
17.     for(int i = 0; i < n; i++)
18.         fo << setw(5) << a[i].tu << setw(5) << a[i].mau << endl;
19.     return 1;
20. }
```

Xuất mảng một
chiều phân số
ra file



Thuật toán interchange sort và ma trận

MA TRẬN

Interchange sort và ma trận



- Bài toán: Định nghĩa hàm sắp ma trận các số nguyên tăng dần bằng thuật toán Interchange sort.

| | 0 | 1 | 2 | 3 |
|---|----|----|----|----|
| 0 | 89 | 12 | 78 | 91 |
| 1 | 61 | 37 | 8 | 18 |
| 2 | 78 | 23 | 35 | 22 |

Ma trận trước khi sắp tăng

| | 0 | 1 | 2 | 3 |
|---|----|----|----|----|
| 0 | 8 | 12 | 18 | 22 |
| 1 | 23 | 35 | 37 | 61 |
| 2 | 78 | 78 | 89 | 91 |

Ma trận sau khi sắp tăng



Thuật toán interchange sort và ma trận

XỬ LÝ MA TRẬN NHƯ MẢNG MỘT CHIỀU

Ma trận và mảng một chiều



— Xử lý ma trận như mảng một chiều.

| | 0 | 1 | 2 | 3 |
|---|----|----|----|----|
| 0 | 89 | 12 | 78 | 91 |
| 1 | 61 | 37 | 8 | 18 |
| 2 | 78 | 23 | 35 | 22 |

Ma trận và mảng một chiều



— Xử lý ma trận như mảng một chiều.

| | 0 | 1 | 2 | 3 |
|---|----|----|----|----|
| 0 | 89 | 12 | 78 | 91 |
| 1 | 61 | 37 | 8 | 18 |
| 2 | 78 | 23 | 35 | 22 |

| | 0 | 1 | 2 | 3 |
|---|---|---|----|----|
| 0 | 0 | 1 | 2 | 3 |
| 1 | 4 | 5 | 6 | 7 |
| 2 | 8 | 9 | 10 | 11 |

Ma trận và mảng một chiều



— Xử lý ma trận như mảng một chiều.

| | 0 | 1 | 2 | 3 |
|---|----|----|----|----|
| 0 | 89 | 12 | 78 | 91 |
| 1 | 61 | 37 | 8 | 18 |
| 2 | 78 | 23 | 35 | 22 |

| 0 | 1 | 2 | 3 |
|---|---|---|---|
|---|---|---|---|

| | | | |
|----|----|----|----|
| 89 | 12 | 78 | 91 |
|----|----|----|----|

| | 0 | 1 | 2 | 3 |
|---|---|---|----|----|
| 0 | 0 | 1 | 2 | 3 |
| 1 | 4 | 5 | 6 | 7 |
| 2 | 8 | 9 | 10 | 11 |

Ma trận và mảng một chiều



— Xử lý ma trận như mảng một chiều.

| | 0 | 1 | 2 | 3 |
|---|----|----|----|----|
| 0 | 89 | 12 | 78 | 91 |
| 1 | 61 | 37 | 8 | 18 |
| 2 | 78 | 23 | 35 | 22 |

| | 0 | 1 | 2 | 3 |
|---|---|---|----|----|
| 0 | 0 | 1 | 2 | 3 |
| 1 | 4 | 5 | 6 | 7 |
| 2 | 8 | 9 | 10 | 11 |

| | | | | | | | |
|----|----|----|----|----|----|---|----|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 89 | 12 | 78 | 91 | 61 | 37 | 8 | 18 |

Ma trận và mảng một chiều



— Xử lý ma trận như mảng một chiều.

| | 0 | 1 | 2 | 3 |
|---|----|----|----|----|
| 0 | 89 | 12 | 78 | 91 |
| 1 | 61 | 37 | 8 | 18 |
| 2 | 78 | 23 | 35 | 22 |

| | 0 | 1 | 2 | 3 |
|---|---|---|----|----|
| 0 | 0 | 1 | 2 | 3 |
| 1 | 4 | 5 | 6 | 7 |
| 2 | 8 | 9 | 10 | 11 |

| | | | | | | | | | | | |
|----|----|----|----|----|----|---|----|----|----|----|----|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 89 | 12 | 78 | 91 | 61 | 37 | 8 | 18 | 78 | 23 | 35 | 22 |

Ma trận và mảng một chiều



— Xử lý ma trận như mảng một chiều.

vt=6

| | 0 | 1 | 2 | 3 |
|---|----|----|----|----|
| 0 | 89 | 12 | 78 | 91 |
| 1 | 61 | 37 | 8 | 18 |
| 2 | 78 | 23 | 35 | 22 |

| | 0 | 1 | 2 | 3 |
|---|---|---|----|----|
| 0 | 0 | 1 | 2 | 3 |
| 1 | 4 | 5 | 6 | 7 |
| 2 | 8 | 9 | 10 | 11 |

| | | | | | | | | | | | |
|----|----|----|----|----|----|---|----|----|----|----|----|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 89 | 12 | 78 | 91 | 61 | 37 | 8 | 18 | 78 | 23 | 35 | 22 |

Ma trận và mảng một chiều



— Xử lý ma trận như mảng một chiều.

vt=6

| | 0 | 1 | 2 | 3 |
|---|----|----|----|----|
| 0 | 89 | 12 | 78 | 91 |
| 1 | 61 | 37 | 8 | 18 |
| 2 | 78 | 23 | 35 | 22 |

| | 0 | 1 | 2 | 3 |
|---|---|---|----|----|
| 0 | 0 | 1 | 2 | 3 |
| 1 | 4 | 5 | 6 | 7 |
| 2 | 8 | 9 | 10 | 11 |

| | | | | | | | | | | | |
|----|----|----|----|----|----|---|----|----|----|----|----|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 89 | 12 | 78 | 91 | 61 | 37 | 8 | 18 | 78 | 23 | 35 | 22 |

Ma trận và mảng một chiều



— Xử lý ma trận như mảng một chiều.

vt=6

| | 0 | 1 | 2 | 3 |
|---|----|----|----|----|
| 0 | 89 | 12 | 78 | 91 |
| 1 | 61 | 37 | 8 | 18 |
| 2 | 78 | 23 | 35 | 22 |

| | 0 | 1 | 2 | 3 |
|---|---|---|----|----|
| 0 | 0 | 1 | 2 | 3 |
| 1 | 4 | 5 | 6 | 7 |
| 2 | 8 | 9 | 10 | 11 |

| | | | | | | | | | | | |
|----|----|----|----|----|----|---|----|----|----|----|----|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 89 | 12 | 78 | 91 | 61 | 37 | 8 | 18 | 78 | 23 | 35 | 22 |

Ma trận và mảng một chiều



— Xử lý ma trận như mảng một chiều.

vt=6

| | 0 | 1 | 2 | 3 |
|---|----|----|----|----|
| 0 | 89 | 12 | 78 | 91 |
| 1 | 61 | 37 | 8 | 18 |
| 2 | 78 | 23 | 35 | 22 |

| | 0 | 1 | 2 | 3 |
|---|---|---|----|----|
| 0 | 0 | 1 | 2 | 3 |
| 1 | 4 | 5 | 6 | 7 |
| 2 | 8 | 9 | 10 | 11 |

| | | | | | | | | | | | |
|----|----|----|----|----|----|---|----|----|----|----|----|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 89 | 12 | 78 | 91 | 61 | 37 | 8 | 18 | 78 | 23 | 35 | 22 |

Ma trận và mảng một chiều



— Xử lý ma trận như mảng một chiều.

vt=6

| | 0 | 1 | 2 | 3 |
|---|----|----|----|----|
| 0 | 89 | 12 | 78 | 91 |
| 1 | 61 | 37 | 8 | 18 |
| 2 | 78 | 23 | 35 | 22 |

| | 0 | 1 | 2 | 3 |
|---|---|---|----|----|
| 0 | 0 | 1 | 2 | 3 |
| 1 | 4 | 5 | 6 | 7 |
| 2 | 8 | 9 | 10 | 11 |

| | | | | | | | | | | | |
|----|----|----|----|----|----|---|----|----|----|----|----|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 89 | 12 | 78 | 91 | 61 | 37 | 8 | 18 | 78 | 23 | 35 | 22 |

Ma trận và mảng một chiều



— Xử lý ma trận như mảng một chiều.

vt=6

| | 0 | 1 | 2 | 3 |
|---|----|----|----|----|
| 0 | 89 | 12 | 78 | 91 |
| 1 | 61 | 37 | 8 | 18 |
| 2 | 78 | 23 | 35 | 22 |

| | 0 | 1 | 2 | 3 |
|---|---|---|----|----|
| 0 | 0 | 1 | 2 | 3 |
| 1 | 4 | 5 | 6 | 7 |
| 2 | 8 | 9 | 10 | 11 |

| | | | | | | | | | | | |
|----|----|----|----|----|----|---|----|----|----|----|----|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 89 | 12 | 78 | 91 | 61 | 37 | 8 | 18 | 78 | 23 | 35 | 22 |

Ma trận và mảng một chiều



— Xử lý ma trận như mảng một chiều.

$$vt=6$$

$$d=vt/n$$

| | 0 | 1 | 2 | 3 |
|---|----|----|----|----|
| 0 | 89 | 12 | 78 | 91 |
| 1 | 61 | 37 | 8 | 18 |
| 2 | 78 | 23 | 35 | 22 |

| | 0 | 1 | 2 | 3 |
|---|---|---|----|----|
| 0 | 0 | 1 | 2 | 3 |
| 1 | 4 | 5 | 6 | 7 |
| 2 | 8 | 9 | 10 | 11 |

| | | | | | | | | | | | |
|----|----|----|----|----|----|---|----|----|----|----|----|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 89 | 12 | 78 | 91 | 61 | 37 | 8 | 18 | 78 | 23 | 35 | 22 |

Ma trận và mảng một chiều



— Xử lý ma trận như mảng một chiều.

$$vt=6$$

$$d=vt/n$$

$$c=vt\%n$$

| | 0 | 1 | 2 | 3 |
|---|----|----|----|----|
| 0 | 89 | 12 | 78 | 91 |
| 1 | 61 | 37 | 8 | 18 |
| 2 | 78 | 23 | 35 | 22 |

| | 0 | 1 | 2 | 3 |
|---|---|---|----|----|
| 0 | 0 | 1 | 2 | 3 |
| 1 | 4 | 5 | 6 | 7 |
| 2 | 8 | 9 | 10 | 11 |

| | | | | | | | | | | | |
|----|----|----|----|----|----|---|----|----|----|----|----|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 89 | 12 | 78 | 91 | 61 | 37 | 8 | 18 | 78 | 23 | 35 | 22 |

Ma trận và mảng một chiều



— Xử lý ma trận như mảng một chiều.

vt=2

| | 0 | 1 | 2 | 3 |
|---|----|----|----|----|
| 0 | 89 | 12 | 78 | 91 |
| 1 | 61 | 37 | 8 | 18 |
| 2 | 78 | 23 | 35 | 22 |

| | 0 | 1 | 2 | 3 |
|---|---|---|----|----|
| 0 | 0 | 1 | 2 | 3 |
| 1 | 4 | 5 | 6 | 7 |
| 2 | 8 | 9 | 10 | 11 |

d=?

c=?

| | | | | | | | | | | | |
|----|----|----|----|----|----|---|----|----|----|----|----|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 89 | 12 | 78 | 91 | 61 | 37 | 8 | 18 | 78 | 23 | 35 | 22 |

Ma trận và mảng một chiều



— Xử lý ma trận như mảng một chiều.

vt=2

d=?

c=?

| | 0 | 1 | 2 | 3 |
|---|----|----|----|----|
| 0 | 89 | 12 | 78 | 91 |
| 1 | 61 | 37 | 8 | 18 |
| 2 | 78 | 23 | 35 | 22 |

| | 0 | 1 | 2 | 3 |
|---|---|---|----|----|
| 0 | 0 | 1 | 2 | 3 |
| 1 | 4 | 5 | 6 | 7 |
| 2 | 8 | 9 | 10 | 11 |

| | | | | | | | | | | | |
|----|----|----|----|----|----|---|----|----|----|----|----|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 89 | 12 | 78 | 91 | 61 | 37 | 8 | 18 | 78 | 23 | 35 | 22 |

Ma trận và mảng một chiều



— Xử lý ma trận như mảng một chiều.

vt=9

| | 0 | 1 | 2 | 3 |
|---|----|----|----|----|
| 0 | 89 | 12 | 78 | 91 |
| 1 | 61 | 37 | 8 | 18 |
| 2 | 78 | 23 | 35 | 22 |

| | 0 | 1 | 2 | 3 |
|---|---|---|----|----|
| 0 | 0 | 1 | 2 | 3 |
| 1 | 4 | 5 | 6 | 7 |
| 2 | 8 | 9 | 10 | 11 |

d=?

c=?

| | | | | | | | | | | | |
|----|----|----|----|----|----|---|----|----|----|----|----|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 89 | 12 | 78 | 91 | 61 | 37 | 8 | 18 | 78 | 23 | 35 | 22 |

Ma trận và mảng một chiều



— Xử lý ma trận như mảng một chiều.

vt=9

| | 0 | 1 | 2 | 3 |
|---|----|----|----|----|
| 0 | 89 | 12 | 78 | 91 |
| 1 | 61 | 37 | 8 | 18 |
| 2 | 78 | 23 | 35 | 22 |

| | 0 | 1 | 2 | 3 |
|---|---|---|----|----|
| 0 | 0 | 1 | 2 | 3 |
| 1 | 4 | 5 | 6 | 7 |
| 2 | 8 | 9 | 10 | 11 |

d=?

c=?

| | | | | | | | | | | | |
|----|----|----|----|----|----|---|----|----|----|----|----|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 89 | 12 | 78 | 91 | 61 | 37 | 8 | 18 | 78 | 23 | 35 | 22 |

Ma trận và mảng một chiều



— Xử lý ma trận như mảng một chiều.

vt=7

| | 0 | 1 | 2 | 3 |
|---|----|----|----|----|
| 0 | 89 | 12 | 78 | 91 |
| 1 | 61 | 37 | 8 | 18 |
| 2 | 78 | 23 | 35 | 22 |

| | 0 | 1 | 2 | 3 |
|---|---|---|----|----|
| 0 | 0 | 1 | 2 | 3 |
| 1 | 4 | 5 | 6 | 7 |
| 2 | 8 | 9 | 10 | 11 |

d=?

c=?

| | | | | | | | | | | | |
|----|----|----|----|----|----|---|----|----|----|----|----|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 89 | 12 | 78 | 91 | 61 | 37 | 8 | 18 | 78 | 23 | 35 | 22 |

Ma trận và mảng một chiều



— Xử lý ma trận như mảng một chiều.

vt=7

| | 0 | 1 | 2 | 3 |
|---|----|----|----|----|
| 0 | 89 | 12 | 78 | 91 |
| 1 | 61 | 37 | 8 | 18 |
| 2 | 78 | 23 | 35 | 22 |

| | 0 | 1 | 2 | 3 |
|---|---|---|----|----|
| 0 | 0 | 1 | 2 | 3 |
| 1 | 4 | 5 | 6 | 7 |
| 2 | 8 | 9 | 10 | 11 |

d=?

c=?

| | | | | | | | | | | | |
|----|----|----|----|----|----|---|----|----|----|----|----|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 89 | 12 | 78 | 91 | 61 | 37 | 8 | 18 | 78 | 23 | 35 | 22 |

Ma trận và mảng một chiều



— Xử lý ma trận như mảng một chiều.

vt=1

| | 0 | 1 | 2 | 3 |
|---|----|----|----|----|
| 0 | 89 | 12 | 78 | 91 |
| 1 | 61 | 37 | 8 | 18 |
| 2 | 78 | 23 | 35 | 22 |

| | 0 | 1 | 2 | 3 |
|---|---|---|----|----|
| 0 | 0 | 1 | 2 | 3 |
| 1 | 4 | 5 | 6 | 7 |
| 2 | 8 | 9 | 10 | 11 |

d=?

c=?

| | | | | | | | | | | | |
|----|----|----|----|----|----|---|----|----|----|----|----|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 89 | 12 | 78 | 91 | 61 | 37 | 8 | 18 | 78 | 23 | 35 | 22 |

Ma trận và mảng một chiều



— Xử lý ma trận như mảng một chiều.

vt=1

| | 0 | 1 | 2 | 3 |
|---|----|----|----|----|
| 0 | 89 | 12 | 78 | 91 |
| 1 | 61 | 37 | 8 | 18 |
| 2 | 78 | 23 | 35 | 22 |

| | 0 | 1 | 2 | 3 |
|---|---|---|----|----|
| 0 | 0 | 1 | 2 | 3 |
| 1 | 4 | 5 | 6 | 7 |
| 2 | 8 | 9 | 10 | 11 |

d=?

c=?

| | | | | | | | | | | | |
|----|----|----|----|----|----|---|----|----|----|----|----|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 89 | 12 | 78 | 91 | 61 | 37 | 8 | 18 | 78 | 23 | 35 | 22 |

Interchange sort và ma trận



- Bài toán: Định nghĩa hàm sắp ma trận các số nguyên tăng dần bằng thuật toán Interchange sort.

| | 0 | 1 | 2 | 3 |
|---|----|----|----|----|
| 0 | 89 | 12 | 78 | 91 |
| 1 | 61 | 37 | 8 | 18 |
| 2 | 78 | 23 | 35 | 22 |

Ma trận trước khi sắp tăng

| | 0 | 1 | 2 | 3 |
|---|----|----|----|----|
| 0 | 8 | 12 | 18 | 22 |
| 1 | 23 | 35 | 37 | 61 |
| 2 | 78 | 78 | 89 | 91 |

Ma trận sau khi sắp tăng

Interchange sort và ma trận



— Định nghĩa hàm

```
11. void InterchangeSort(int a[][100], int m, int n)
12. {
13.     for(int i=0; i<=m*n-2; i++)
14.         for (int j=i+1; j<=m*n-1; j++)
15.             if(a[i] > a[j])
16.                 swap(a[i], a[j]);
17. }
```


Interchange sort và ma trận



— Định nghĩa hàm

```
11. void InterchangeSort(int a[][100], int m, int n)
12. {
13.     for(int i=0; i<=m*n-2; i++)
14.         for (int j=i+1; j<=m*n-1; j++)
15.             if(a[i/n] > a[j/n] )
16.                 swap(a[i/n] ,a[j/n] );
17. }
```

Interchange sort và ma trận



— Định nghĩa hàm

```
11. void InterchangeSort(int a[][100], int m, int n)
12. {
13.     for(int i=0; i<=m*n-2; i++)
14.         for (int j=i+1; j<=m*n-1; j++)
15.             if(a[i/n][ ]>a[j/n][ ])
16.                 swap(a[i/n][ ],a[j/n][ ]);
17. }
```

Interchange sort và ma trận



— Định nghĩa hàm

```
11. void InterchangeSort(int a[][100], int m, int n)
12. {
13.     for(int i=0; i<=m*n-2; i++)
14.         for (int j=i+1; j<=m*n-1; j++)
15.             if(a[i/n][i%n]>a[j/n][j%n])
16.                 swap(a[i/n][i%n],a[j/n][j%n]);
17. }
```



Interchange sort và ma trận

PROJECT G04 – DỰ ÁN G04

Interchange sort và ma trận



— Viết chương trình thực hiện các yêu cầu sau:

- + Nhập ma trận các số nguyên từ các tập tin: intmatran01.inp; intmatran02.inp; ...; intmatran09.inp; intmatran10.inp; intmatran11.inp; intmatran12.inp; intmatran13.inp;
- + Sắp xếp ma trận tăng dần bằng thuật toán interchange sort.
- + Xuất ma trận sau khi sắp xếp ra các tập tin: intmatran01.out; intmatran02.out; ...; intmatran09.out; intmatran10.out; intmatran11.out; intmatran12.out; intmatran13.out;

Interchange sort và ma trận



- Định dạng tập tin
 - + `intmatranxx.inp` và
 - + `intmatranxx.out`
- Dòng đầu tiên: lưu hai số nguyên tương ứng với số hàng ma trận (m) và số cột ma trận (n).
- m dòng tiếp theo: mỗi dòng lưu n số nguyên tương ứng với các giá trị trong ma trận.

*intmatrandata01.inp - Notepad

File Edit Format View Help

```
8 10
-27 63 -7 -49 78 -27 92 -71 71 59
14 20 32 -81 -17 -12 49 -38 17 78
-71 77 33 -62 49 39 47 -23 -2 -20
89 -89 -39 0 3 97 67 -98 -32 -33
67 82 20 -9 4 -3 -1 -87 -54 61
-68 -6 -86 52 9 -80 84 -99 5 18
-54 -19 18 100 -73 -40 -58 -80 -12 -96
-43 44 32 -85 -11 32 77 33 -75 -32
```

Interchange sort và ma trận



— Định nghĩa hàm

```
11. void InterchangeSort(int a[][100], int m, int n)
12. {
13.     for(int i=0; i<=m*n-2; i++)
14.         for (int j=i+1; j<=m*n-1; j++)
15.             if(a[i/n][i%n]>a[j/n][j%n])
16.                 swap(a[i/n][i%n],a[j/n][j%n]);
17. }
```

Interchange sort và ma trận



```
11. int Nhap(int a[][100], int& m, int& n, string filename)
```

```
12. {
```

```
13. |
```

Nhập ma trận
số nguyên từ
file

Interchange sort và ma trận



```
11.int Nhap(int a[][100],int& m,int& n,string filename)
```

```
12.{
```

```
13.    ifstream fi(filename);
```

Nhập ma trận
số nguyên từ
file

Interchange sort và ma trận



```
11.int Nhap(int a[][100],int& m,int& n,string filename)
```

```
12.{
```

```
13.    ifstream fi(filename);
```

Ý1: fi là đối tượng thuộc lớp
ifstream.

Nhập ma trận
số nguyên từ
file

Interchange sort và ma trận



```
11.int Nhap(int a[][100],int& m,int& n,string filename)
12.{
13.    ifstream fi(filename);
```

Ý2: Dòng lệnh số 13 khai báo đối tượng fi với đối số có tên filename và có kiểu string.

Nhập ma trận
số nguyên từ
file

Interchange sort và ma trận



```
11.int Nhap(int a[][100],int& m,int& n,string filename)
12.{
13.    ifstream fi(filename);
```

Ý3: Khi chương trình thực hiện tới dòng lệnh số 13. Đối tượng fi gọi thực hiện phương thức thiết lập với tham số có kiểu string. Phương thức thiết lập sẽ mở tập tin có tên lưu trong biến filename.

Nhập ma trận
số nguyên từ
file

Interchange sort và ma trận



```
11.int Nhap(int a[][100],int& m,int& n,string filename)
12.{
13.    ifstream fi(filename);
14.    if (fi.fail()==true)
15.        return 0;
16.    fi >> m >> n;
17.    for(int i=0; i<m; i++)
18.        for(int j=0; j<n; j++)
19.            fi >> a[i][j];
20.    return 1;
21.}
```

Nhập ma trận
số nguyên từ
file

```
11.int Xuat(int a[][100],int m,int n,string filename)
```

```
12.{
```

```
13.    ofstream fo(filename);
```

```
24.}
```



Xuất ma trận
số nguyên ra
file

```
11.int Xuat(int a[][100],int m,int n,string filename)
```

```
12.{
```

```
13.    ofstream fo(filename);
```

```
24.}
```



Xuất ma trận
số nguyên ra
file

```
11.int Xuat(int a[][100],int m,int n,string filename)
```

```
12.{
```

```
13.    ofstream fo(filename);
```

Ý1: fo là đối tượng thuộc lớp
ofstream.

```
24.}
```



Xuất ma trận
số nguyên ra
file


```
11.int Xuat(int a[][100],int m,int n,string filename)
```

```
12.{
```

```
13.    ofstream fo(filename);
```

Ý2: Dòng lệnh số 13 khai báo đối tượng fo với đối số có tên filename và có kiểu string.

```
24.}
```



Xuất ma trận
số nguyên ra
file

```
11.int Xuat(int a[][100],int m,int n,string filename)
12.{
13.    ofstream fo(filename);
```



Ý3: Khi chương trình thực hiện tới dòng lệnh số 13. Đối tượng fo gọi thực hiện phương thức thiết lập với tham số có kiểu string. Phương thức thiết lập sẽ mở tập tin có tên lưu trong biến filename.

```
24.}
```

Xuất ma trận
số nguyên ra
file

```
11.int Xuat(int a[][100],int m,int n,string filename)
12.{
13.    ofstream fo(filename);
14.    if (fo.fail()==true)
15.        return 0;
16.    fo << setw(5) << m << n << endl;
17.    for(int i = 0; i<m; i++)
18.    {
19.        for(int j = 0; j<n; j++)
20.            fo << setw(5) << a[i][j];
21.        fo << endl;
22.    }
23.    return 1;
24.}
```



Xuất ma trận
số nguyên ra
file

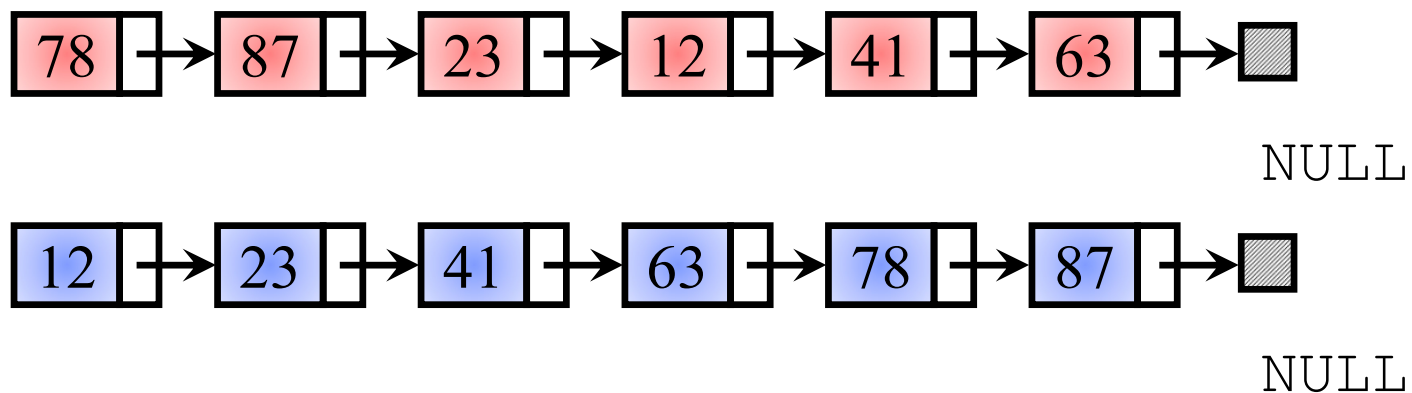


Thuật toán interchange sort và dslk đơn
DANH SÁCH LIÊN KẾT ĐƠN

Interchange sort và dslk đơn



- Bài toán: Định nghĩa hàm sắp xếp danh sách liên kết đơn các số nguyên tăng dần bằng thuật toán Interchange sort.



Interchange sort và dslk đơn



```
11. void InterchangeSort(LIST& l)
12. {
13.     for (NODE*p=l.pHead; p->pNext!=NULL; p=p->pNext)
14.         for (NODE*q=p->pNext; q!=NULL; q=q->pNext)
15.             if (p->info > q->info)
16.                 swap(p->info, q->info);
17. }
```



Thuật toán interchange sort và dslk đơn

PROJECT G05 – DỰ ÁN G05

Thuật toán interchange sort và dslk đơn



— Viết chương trình thực hiện các yêu cầu sau:

- + Nhập dslk đơn các số nguyên từ các tập tin: intdata01.inp; intdata02.inp; ...; intdata09.inp; intdata10.inp; intdata11.inp; intdata12.inp; intdata13.inp;
- + Sắp xếp dslk đơn các số nguyên tăng dần bằng thuật toán interchange sort.
- + Xuất dslk đơn các số nguyên sau khi sắp xếp ra các tập tin: intdata01.out; intdata02.out; ...; intdata09.out; intdata10.out; intdata11.out; intdata12.out; intdata13.out;

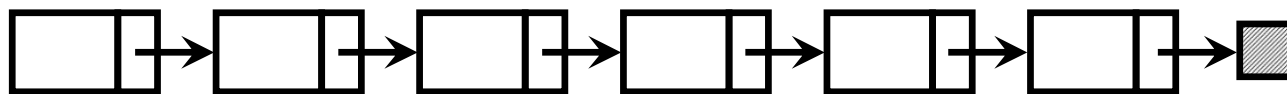
Thuật toán interchange sort và dslk đơn



- Định dạng tập tin `intdataxx.inp` và `intdataxx.out`
 - + Dòng đầu tiên: số phần tử của dslk đơn các số nguyên (n).
 - + Dòng tiếp theo: lưu n số nguyên tương ứng với các giá trị trong dslk đơn các số nguyên.

```
*intdata01.inp - Notepad
File Edit Format View Help
10
24 56 53 44 -54 6 63 -47 91 -99
```

Interchange sort và dslk đơn

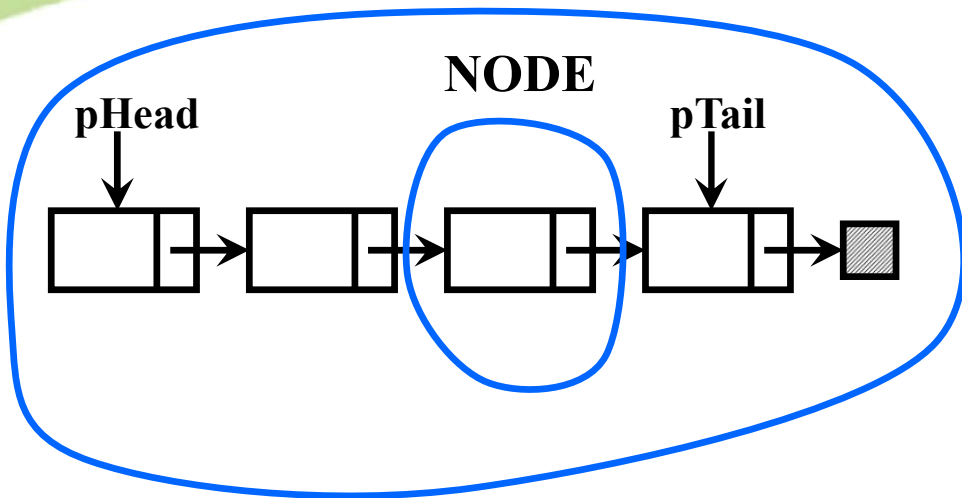


NULL

Cấu trúc dữ liệu danh
sách liên kết đơn

LIST

NODE

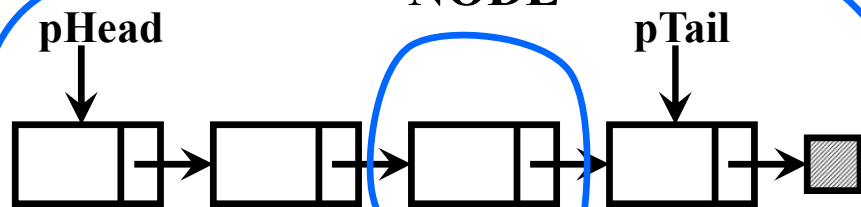


Cấu trúc dữ liệu danh sách liên kết đơn

```
11.struct node
12.{
13.|    KDL info;
14.|    struct node* pNext;
15.};
16.typedef struct node NODE;
17.struct list
18.{
19.|    NODE* pHead;
20.|    NODE* pTail;
21.};
22.typedef struct list LIST;
```

LIST

NODE



CTDL dslk đơn các
số nguyên

```
11.struct node
12.{
13.|    int info;
14.|    struct node* pNext;
15.};
16.typedef struct node NODE;
17.struct list
18.{
19.|    NODE* pHead;
20.|    NODE* pTail;
21.};
22.typedef struct list LIST;
```

Interchange sort và dslk đơn



- Khái niệm: Khởi tạo danh sách liên kết đơn là tạo ra danh sách rỗng không chứa node nào hết.
- Định nghĩa hàm khởi tạo danh sách liên kết đơn.

```
11. void Init(LIST& l)
12. {
13. |     l.pHead = NULL;
14. |     l.pTail = NULL;
15. }
```

Khởi tạo danh sách
liên kết đơn

Interchange sort và dslk đơn



— Khái niệm: Kiểm tra danh sách liên kết đơn rỗng là hàm trả về giá trị 1 khi danh sách rỗng. Trong tình huống danh sách không rỗng thì hàm sẽ trả về giá trị 0.

```
11. int IsEmpty(LIST l)
12. {
13.     if(l.pHead==NULL)
14.         return 1;
15.     return 0;
16. }
```

Kiểm tra danh sách
liên kết đơn rỗng

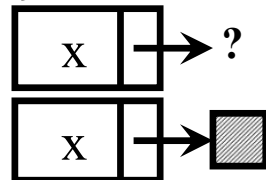
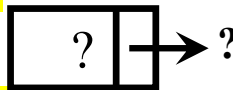
Interchange sort và dslk đơn



- **Khái niệm:** Tạo node cho danh sách liên kết đơn là xin cấp phát bộ nhớ có kích thước bằng với kích thước của kiểu dữ liệu **NODE** để chứa thông tin đã được biết trước.

Tạo node cho danh sách liên kết đơn

```
11. NODE* GetNode(KDL x)
12. {
13.     NODE* p=new NODE;
14.     if(p==NULL)
15.         return NULL;
16.     p->info = x;
17.     p->pNext=NULL;
18.     return p;
19. }
```



Interchange sort và dslk đơn



— Phân tích câu lệnh dòng 11

11. **NODE* GetNode(KDL x)**

— Tên hàm tạo node cho dslk đơn là **GetNode**.

— Có một tham số đầu vào, tên tham số là x, tham số là tham trị.

— KDL trả về của hàm **GetNode** là con trỏ kiểu cấu trúc **NODE**.

— Về mặt bản chất hàm **GetNode** sẽ trả về một địa chỉ ô nhớ.

— Phân tích câu lệnh dòng 18

18. **return p;**

— Kết thúc lời gọi hàm và trả về địa chỉ ô nhớ đang được lưu trong biến con trỏ **p**.

— Ý nghĩa của địa chỉ ô nhớ đang lưu biến con trỏ **p** xem ở slide ngay sau.

Interchange sort và dslk đơn



Phân tích dòng lệnh 13.

13. `NODE *p=new NODE;`

- `p` là một biến con trỏ kiểu cấu trúc `NODE`.
- Miền giá trị của biến con trỏ `p` là địa chỉ ô nhớ.
- `new NODE` là xin cấp phát động một vùng nhớ có kích thước bằng kích thước của kiểu dữ liệu `NODE`.

- Nếu việc cấp phát thành công OS sẽ trả về địa chỉ ô nhớ đầu tiên của vùng nhớ được cấp phát, địa chỉ ô nhớ này được gán cho biến con trỏ `p`.
- Nếu việc cấp phát thất bại, OS sẽ trả về một địa chỉ đặc biệt là địa chỉ `NULL`, địa chỉ `NULL` này sẽ được gán cho biến con trỏ `p`.
- Như vậy, thông thường sau câu lệnh thứ 13, biến con trỏ `p` sẽ giữ địa chỉ ô nhớ đầu tiên của vùng nhớ có kích thước bằng kích thước của KDL `NODE`.

Interchange sort và dslk đơn



- **Khái niệm:** Tạo node cho danh sách liên kết đơn là xin cấp phát bộ nhớ có kích thước bằng với kích thước của kiểu dữ liệu **NODE** để chứa thông tin đã được biết trước.

Tạo node cho danh sách liên kết đơn

```
11. NODE* GetNode(KDL x)
12. {
13.     NODE *p=new NODE;
14.     if(p==NULL)
15.         return NULL;
16.     p->info = x;
17.     p->pNext = NULL;
18.     return p;
19. }
```

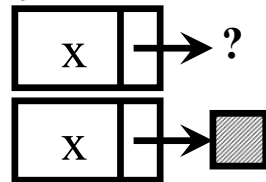
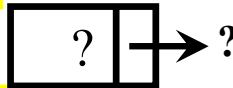
Interchange sort và dslk đơn



- Khái niệm: Tạo node cho danh sách liên kết đơn là xin cấp phát bộ nhớ có kích thước bằng với kích thước của kiểu dữ liệu **NODE** để chứa thông tin đã được biết trước.

Tạo node cho dslk đơn các số nguyên

```
11. NODE* GetNode(int x)
12. {
13.     NODE *p=new NODE;
14.     if(p==NULL)
15.         return NULL;
16.     p->info = x;
17.     p->pNext=NULL;
18.     return p;
19. }
```

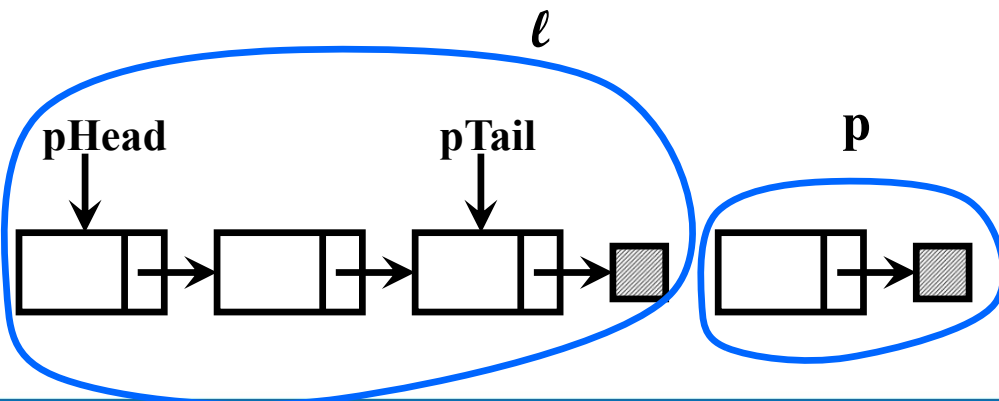


Interchange sort và dslk đơn



— Khái niệm: Thêm một node vào cuối danh sách liên kết đơn là gắn node đó vào cuối danh sách.

— Hình vẽ



```
11. void AddTail(LIST&l, NODE*p)
12. {
13.     if(l.pHead==NULL)
14.         l.pHead=l.pTail=p;
15.     else
16.     {
17.         |   l.pTail->pNext=p;
18.         |   l.pTail = p;
19.     }
20. }
```

Thêm vào cuối dslk đơn

Interchange sort và ds lk đơn



```
11.int Nhap(LIST&l, string filename)
```

```
12.{
```

```
13.    ifstream fi(filename);
```

Nhập danh
sách liên
kết đơn từ
file

Interchange sort và dslk đơn



```
11.int Nhap(LIST&l, string filename)
```

```
12.{
```

```
13.    ifstream fi(filename);
```

Ý1: fi là đối tượng thuộc lớp
ifstream.

Nhập danh
sách liên
kết đơn từ
file

Interchange sort và dslk đơn



```
11.int Nhap(LIST&l, string filename)
```

```
12.{
```

```
13.    ifstream fi(filename);
```

Ý2: Dòng lệnh số 13 khai báo đối tượng fi với đối số có tên filename và có kiểu string.

Nhập danh
sách liên
kết đơn từ
file

Interchange sort và dslk đơn



```
11.int Nhap(LIST&l, string filename)
```

```
12.{
```

```
13.    ifstream fi(filename);
```

Ý3: Khi chương trình thực hiện tới dòng lệnh số 13. Đối tượng fi gọi thực hiện phương thức thiết lập với tham số có kiểu string. Phương thức thiết lập sẽ mở tập tin có tên lưu trong biến filename.

Nhập danh
sách liên
kết đơn từ
file

Interchange sort và dslk đơn



```
11. int Nhap(LIST&l, string filename)
12. {
13.     ifstream fi(filename);
14.     if (fi.fail() == true)
15.         return 0;
16.     int n;
17.     int x;
18.     fi >> n;
19.     Init(l);
20.     ...
```

Nhập danh
sách liên
kết đơn từ
file

Interchange sort và dslk đơn



```
11. | ...
12. | for (int i = 1; i <= n; i++)
13. | {
14. |     fi >> x;
15. |     NODE* p = GetNode(x);
16. |     if (p != NULL)
17. |         AddTail(l, p);
18. | }
19. | return 1;
20. | }
```

Nhập danh
sách liên
kết đơn từ
file

Interchange sort và dslk đơn



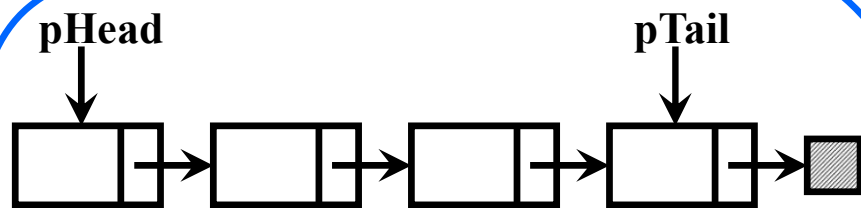
```
11. void InterchangeSort(LIST& l)
12. {
13.     for (NODE*p=l.pHead; p->pNext!=NULL; p=p->pNext)
14.         for (NODE*q=p->pNext; q!=NULL; q=q->pNext)
15.             if (p->info > q->info)
16.                 swap(p->info, q->info);
17. }
```

Sắp xếp danh sách
liên kết đơn tăng dần

Interchange sort và dslk đơn



LIST



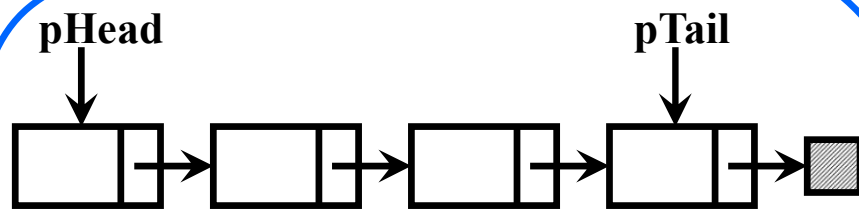
Xuất danh sách liên
kết đơn

```
11. void Xuat(LIST l)
12. {
13.     NODE* p = l.pHead;
14.     while (p != NULL)
15.     {
16.         cout<<p->info;
17.         p = p->pNext;
18.     }
19. }
```

Interchange sort và dslk đơn



LIST



Đếm node trong danh sách liên kết đơn

```
11. int DemNode(LIST l)
12. {
13.     int dem = 0;
14.     NODE* p = l.pHead;
15.     while (p != NULL)
16.     {
17.         dem++;
18.         p = p->pNext;
19.     }
20.     return dem;
21. }
```

```
11.int Xuat(LIST l, string filename)
```

```
12.{
```

```
13.    ofstream fo(filename);
```

```
24.}
```



Xuất danh
sách liên
kết đơn ra
file

```
11.int Xuat(LIST l, string filename)
```

```
12.{
```

```
13.    ofstream fo(filename);
```

Ý1: fo là đối tượng thuộc lớp
ofstream.

```
24.}
```



Xuất danh
sách liên
kết đơn ra
file

```
11.int Xuat(LIST l, string filename)
```

```
12.{
```

```
13.    ofstream fo(filename);
```

Ý2: Dòng lệnh số 13 khai báo đối tượng fo với đối số có tên filename và có kiểu string.

```
24.}
```



Xuất danh
sách liên
kết đơn ra
file


```
11.int Xuat(LIST l, string filename)
```

```
12.{
```

```
13.    ofstream fo(filename);
```

Ý3: Khi chương trình thực hiện tới dòng lệnh số 13. Đối tượng fo gọi thực hiện phương thức thiết lập với tham số có kiểu string. Phương thức thiết lập sẽ mở tập tin có tên lưu trong biến filename.

```
24.}
```



Xuất danh sách liên kết đơn ra file

```
11.int Xuat(LIST l, string filename)
12.{
13.    ofstream fo(filename);
14.    if (fo.fail()==true)
15.        return 0;
16.    fo << setw(5) << DemNode(1) << endl;
17.    NODE* p = l.pHead;
18.    while (p != NULL)
19.    {
20.        fo << setw(5) << p->info;
21.        p = p->pNext;
22.    }
23.    return 1;
24.}
```



Xuất danh
sách liên
kết đơn ra
file

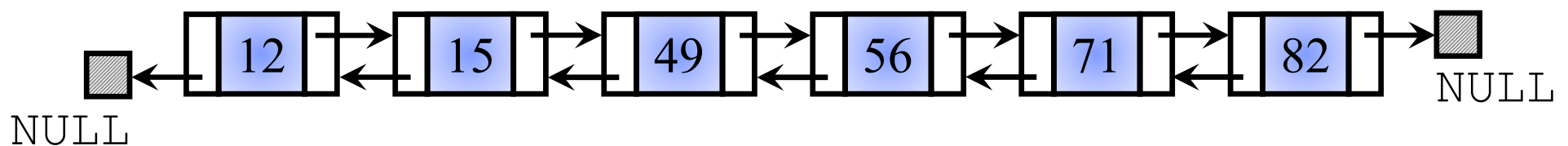
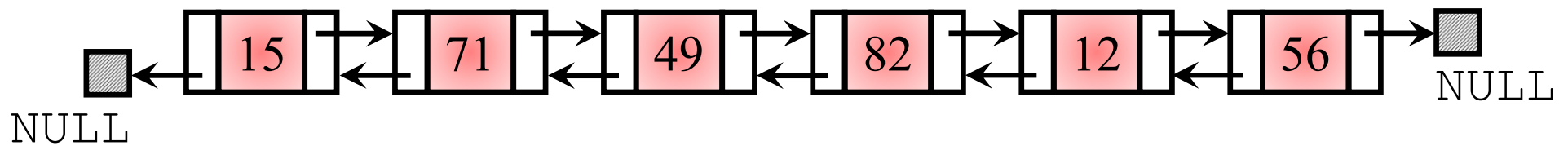


Thuật toán interchange sort và dslk kép
DANH SÁCH LIÊN KẾT KÉP

Interchange sort và dslk kép



- Bài toán: Định nghĩa hàm sắp xếp danh sách liên kết kép các số nguyên tăng dần bằng thuật toán Interchange sort.



Interchange sort và dslk kép



```
11. void InterchangeSort(LIST& l)
12. {
13.     for (NODE*p=l.pHead; p->pNext!=NULL; p=p->pNext)
14.         for (NODE*q=p->pNext; q!=NULL; q=q->pNext)
15.             if (p->info > q->info)
16.                 swap(p->info, q->info);
17. }
```

Sắp xếp danh sách
liên kết kép tăng dần



PROJECT G06 – DỰ ÁN G06

Dự án A50



— Viết chương trình thực hiện các yêu cầu sau:

- + Nhập dslk kép các số nguyên từ các tập tin: intdata01.inp; intdata02.inp; ...; intdata09.inp; intdata10.inp; intdata11.inp; intdata12.inp; intdata13.inp;
- + Sắp xếp dslk kép các số nguyên tăng dần bằng thuật toán interchange sort.
- + Xuất dslk kép các số nguyên sau khi sắp xếp ra các tập tin: intdata01.out; intdata02.out; ...; intdata09.out; intdata10.out; intdata11.out; intdata12.out; intdata13.out;

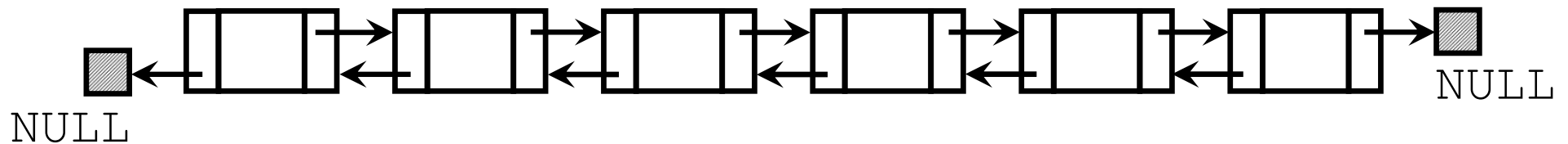
Interchange sort và dslk kép



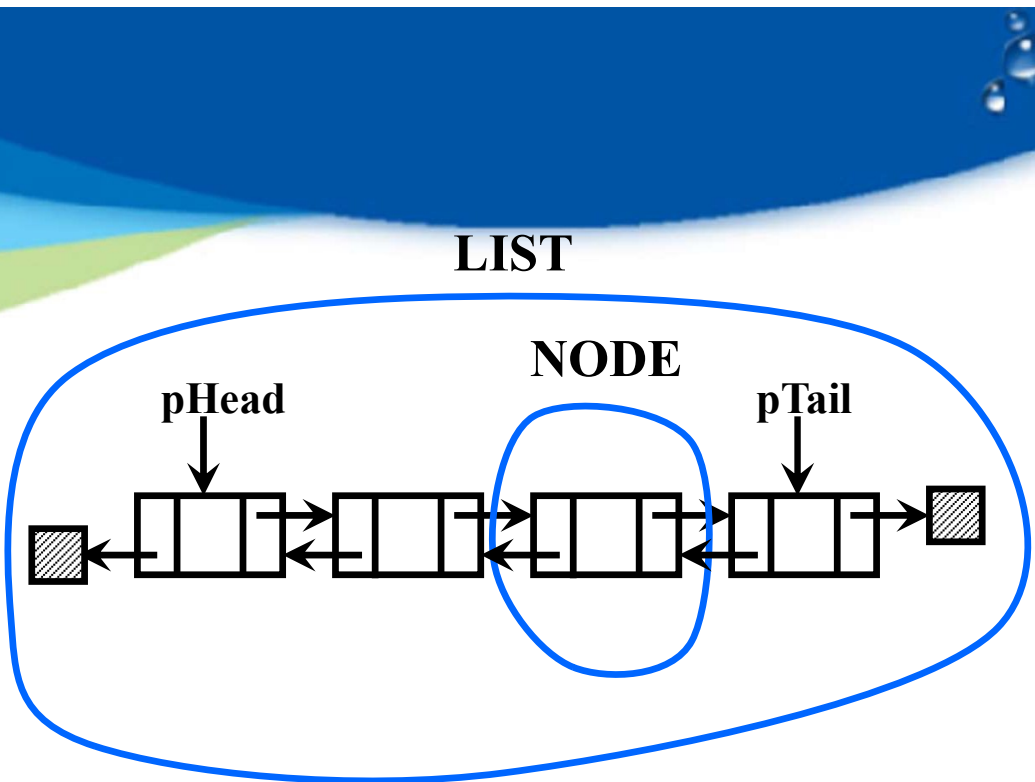
- Định dạng tập tin `intdataxx.inp` và `intdataxx.out`
 - + Dòng đầu tiên: số phần tử của dslk kép các số nguyên (n).
 - + Dòng tiếp theo: lưu n số nguyên tương ứng với các giá trị trong dslk kép các số nguyên.

```
*intdata01.inp - Notepad
File Edit Format View Help
10
24 56 53 44 -54 6 63 -47 91 -99
```


Interchange sort và dslk kép



Cấu trúc dữ liệu danh
sách liên kết kép

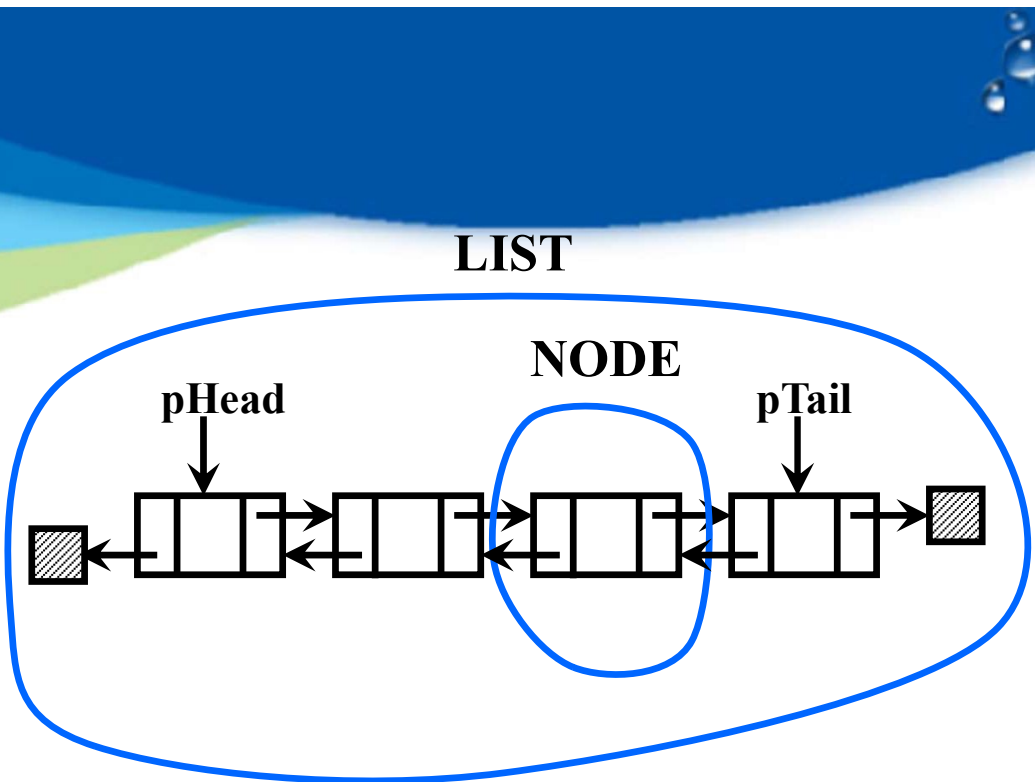


Cấu trúc dữ liệu danh sách liên kết kép

```

11.struct node
12.{
13.    KDL info;
14.    struct node* pNext;
15.    struct node* pPrev;
16.};
17.typedef struct node NODE;
18.struct list
19.{
20.    NODE* pHead;
21.    NODE* pTail;
22.};
23.typedef struct list LIST;

```



CTDL dslk kép các
số nguyên

```

11.struct node
12.{
13.|   int info;
14.|   struct node* pNext;
15.|   struct node* pPrev;
16.};
17.typedef struct node NODE;
18.struct list
19.{
20.|   NODE* pHead;
21.|   NODE* pTail;
22.};
23.typedef struct list LIST;

```

Interchange sort và dslk kép



- Khái niệm: Khởi tạo danh sách liên kết kép là tạo ra danh sách rỗng không chứa node nào hết.
- Định nghĩa hàm khởi tạo danh sách liên kết kép.

```
11. void Init(LIST& l)
12. {
13. |     l.pHead = NULL;
14. |     l.pTail = NULL;
15. }
```

Khởi tạo danh sách
liên kết kép

Interchange sort và dslk kép



— Khái niệm: Kiểm tra danh sách liên kết kép rỗng là hàm trả về giá trị 1 khi danh sách rỗng. Trong tình huống danh sách không rỗng thì hàm sẽ trả về giá trị 0.

```
11. int IsEmpty(LIST l)
12. {
13.     if(l.pHead==NULL)
14.         return 1;
15.     return 0;
16. }
```

Kiểm tra danh sách
liên kết kép rỗng

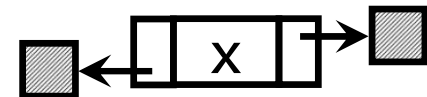
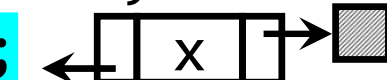
Interchange sort và dslk kép



- **Khái niệm:** Tạo node cho danh sách liên kết kép là xin cấp phát bộ nhớ có kích thước bằng với kích thước của kiểu dữ liệu **NODE** để chứa thông tin đã được biết trước.

Tạo node cho danh sách liên kết kép

```
11. NODE* GetNode(KDL x)
12. {
13.     NODE* p=new NODE;
14.     if(p==NULL)
15.         return NULL;
16.     p->info = x;
17.     p->pNext=NULL;
18.     p->pPrev=NULL;
19.     return p;
20. }
```



Interchange sort và dslk kép



— Phân tích câu lệnh dòng 11

11. **NODE* GetNode(KDL x)**

— Tên hàm tạo node cho dslk kép là **GetNode**.

— Có một tham số đầu vào, tên tham số là x, tham số là tham trị.

— KDL trả về của hàm **GetNode** là con trỏ kiểu cấu trúc **NODE**.

— Về mặt bản chất hàm **GetNode** sẽ trả về một địa chỉ ô nhớ.

— Phân tích câu lệnh dòng 19

19. **return p;**

— Kết thúc lời gọi hàm và trả về địa chỉ ô nhớ đang được lưu trong biến con trỏ **p**.

— Ý nghĩa của địa chỉ ô nhớ đang lưu biến con trỏ **p** xem ở slide ngay sau.

Interchange sort và dslk kép



Phân tích dòng lệnh 13.

13. `NODE *p=new NODE;`

- `p` là một biến con trỏ kiểu cấu trúc `NODE`.
- Miền giá trị của biến con trỏ `p` là địa chỉ ô nhớ.
- `new NODE` là xin cấp phát động một vùng nhớ có kích thước bằng kích thước của kiểu dữ liệu `NODE`.

- Nếu việc cấp phát thành công OS sẽ trả về địa chỉ ô nhớ đầu tiên của vùng nhớ được cấp phát, địa chỉ ô nhớ này được gán cho biến con trỏ `p`.
- Nếu việc cấp phát thất bại, OS sẽ trả về một địa chỉ đặc biệt là địa chỉ `NULL`, địa chỉ `NULL` này sẽ được gán cho biến con trỏ `p`.
- Như vậy, thông thường sau câu lệnh thứ 13, biến con trỏ `p` sẽ giữ địa chỉ ô nhớ đầu tiên của vùng nhớ có kích thước bằng kích thước của KDL `NODE`.

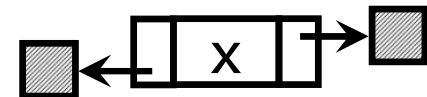
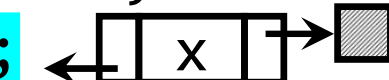
Interchange sort và dslk kép



- **Khái niệm:** Tạo node cho danh sách liên kết kép là xin cấp phát bộ nhớ có kích thước bằng với kích thước của kiểu dữ liệu **NODE** để chứa thông tin đã được biết trước.

Tạo node cho danh sách liên kết kép

```
11. NODE* GetNode(KDL x)
12. {
13.     NODE* p=new NODE;
14.     if(p==NULL)
15.         return NULL;
16.     p->info = x;
17.     p->pNext=NULL;
18.     p->pPrev=NULL;
19.     return p;
20. }
```



Interchange sort và dslk kép



- Khái niệm: Tạo node cho danh sách liên kết kép là xin cấp phát bộ nhớ có kích thước bằng với kích thước của kiểu dữ liệu NODE để chứa thông tin đã được biết trước.

Tạo node cho dslk kép các số nguyên

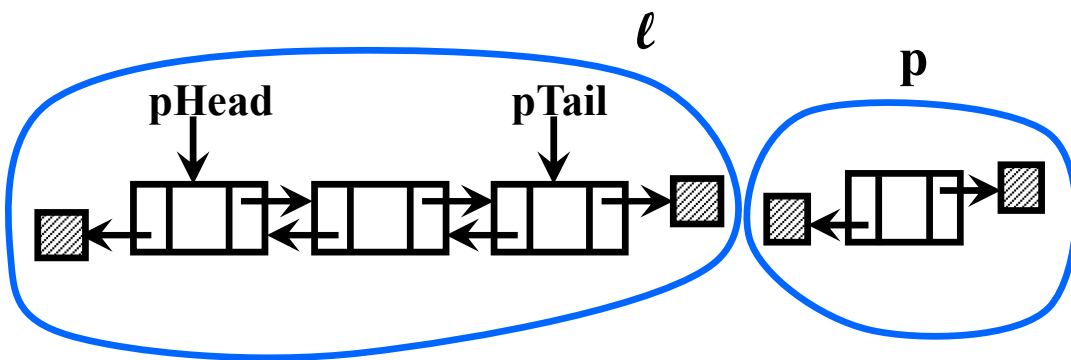
```
11. NODE* GetNode(int x)
12. {
13.     NODE* p=new NODE;
14.     if(p==NULL)
15.         return NULL;
16.     p->info = x;
17.     p->pNext=NULL;
18.     p->pPrev=NULL;
19.     return p;
20. }
```

Interchange sort và dslk kép



— Khái niệm: Thêm một node vào cuối danh sách liên kết kép là gắn node đó vào cuối danh sách.

— Hình vẽ



Thêm vào cuối dslk kép

```
11. void AddTail(LIST&l, NODE*p)
12. {
13.     if(l.pHead==NULL)
14.         l.pHead=l.pTail=p;
15.     else
16.     {
17.         l.pTail->pNext=p;
18.         p->pPrev=l.pTail;
19.         l.pTail = p;
20.     }
21. }
```

Interchange sort và dslk kép



```
11.int Nhap(LIST&l, string filename)
```

```
12.{
```

```
13.    ifstream fi(filename);
```

Nhập danh
sách liên
kết kép từ
file

Interchange sort và dslk kép



```
11.int Nhap(LIST& l, string filename)
```

```
12.{
```

```
13.    ifstream fi(filename);
```

Ý1: fi là đối tượng thuộc lớp
ifstream.

Nhập danh
sách liên
kết kép từ
file

Interchange sort và dslk kép



```
11.int Nhap(LIST& l, string filename)
```

```
12.{
```

```
13.    ifstream fi(filename);
```

Ý2: Dòng lệnh số 13 khai báo đối tượng fi với đối số có tên filename và có kiểu string.

Nhập danh sách liên kết kép từ file

Interchange sort và dslk kép



```
11.int Nhap(LIST& l, string filename)
```

```
12.{
```

```
13.    ifstream fi(filename);
```

Ý3: Khi chương trình thực hiện tới dòng lệnh số 13. Đối tượng fi gọi thực hiện phương thức thiết lập với tham số có kiểu string. Phương thức thiết lập sẽ mở tập tin có tên lưu trong biến filename.

Nhập danh sách liên kết kép từ file

Interchange sort và dslk kép



```
11. int Nhap(LIST& l, string filename)
12. {
13.     ifstream fi(filename);
14.     if (fi.fail()==true)
15.         return 0;
16.     int n;
17.     int x;
18.     fi >> n;
19.     Init(l);
20.     ...
```

Nhập danh
sách liên
kết kép từ
file

Interchange sort và dslk kép



```
11. | ...
12. | for (int i = 1; i <= n; i++)
13. | {
14. |     fi >> x;
15. |     NODE* p = GetNode(x);
16. |     if (p != NULL)
17. |         AddTail(1, p);
18. | }
19. | return 1;
20. | }
```

Nhập danh
sách liên
kết kép từ
file

Interchange sort và dslk kép



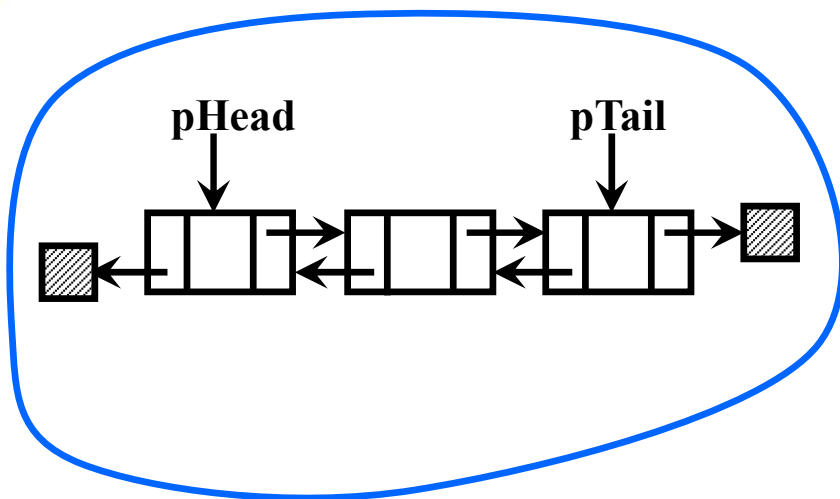
```
11. void InterchangeSort(LIST& l)
12. {
13.     for (NODE*p=l.pHead; p->pNext!=NULL; p=p->pNext)
14.         for (NODE*q=p->pNext; q!=NULL; q=q->pNext)
15.             if (p->info > q->info)
16.                 swap(p->info, q->info);
17. }
```

Sắp xếp danh sách
liên kết kép tăng dần

Interchange sort và dslk kép



ℓ



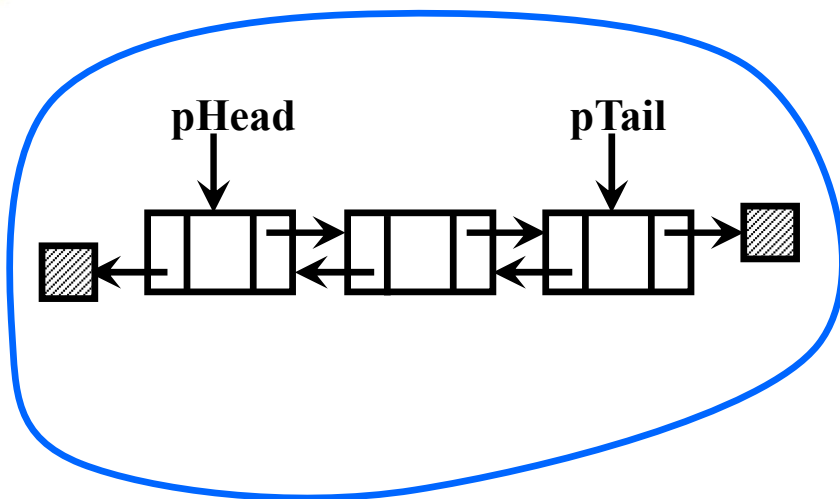
Xuất danh sách liên
kết kép ra màn hình

```
11. void Xuat(LIST l)
12. {
13.     NODE* p = l.pHead;
14.     while (p != NULL)
15.     {
16.         cout<<p->info;
17.         p = p->pNext;
18.     }
19. }
```

Interchange sort và dslk kép



ℓ



Đếm node trong danh sách liên kết kép

```
11. int DemNode(LIST l)
12. {
13.     int dem = 0;
14.     NODE* p = l.pHead;
15.     while (p != NULL)
16.     {
17.         dem++;
18.         p = p->pNext;
19.     }
20.     return dem;
21. }
```

```
11.int Xuat(LIST l, string filename)
```

```
12.{
```

```
13.    ofstream fo(filename);
```

```
24.}
```



Xuất danh
sách liên
kết kép ra
file

```
11.int Xuat(LIST l, string filename)
```

```
12.{
```

```
13.    ofstream fo(filename);
```

Ý1: fo là đối tượng thuộc lớp
ofstream.

```
24.}
```



Xuất danh
sách liên
kết kép ra
file

```
11.int Xuat(LIST l, string filename)
```

```
12.{
```

```
13.    ofstream fo(filename);
```

Ý2: Dòng lệnh số 13 khai báo đối tượng fo với đối số có tên filename và có kiểu string.

```
24.}
```



Xuất danh
sách liên
kết kép ra
file

```
11.int Xuat(LIST l, string filename)
```

```
12.{
```

```
13.    ofstream fo(filename);
```

Ý3: Khi chương trình thực hiện tới dòng lệnh số 13. Đối tượng fo gọi thực hiện phương thức thiết lập với tham số có kiểu string. Phương thức thiết lập sẽ mở tập tin có tên lưu trong biến filename.

```
24.}
```



Xuất danh
sách liên
kết kép ra
file



```
11.int Xuat(LIST l, string filename)
12.{
13.    ofstream fo(filename);
14.    if (fo.fail()==true)
15.        return 0;
16.    fo << setw(5) << DemNode(1) << endl;
17.    NODE* p = l.pHead;
18.    while (p != NULL)
19.    {
20.        fo << setw(5) << p->info;
21.        p = p->pNext;
22.    }
23.    return 1;
24.}
```

Xuất danh
sách liên
kết kép ra
file



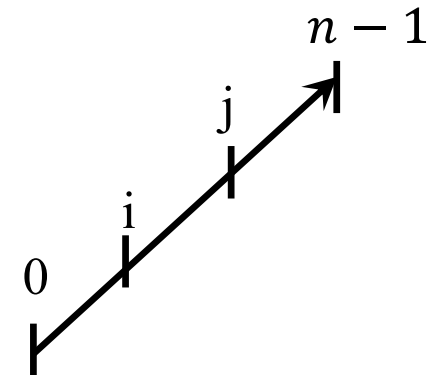
ĐỘ PHỨC TẠP CỦA THUẬT TOÁN

Độ phức tạp của thuật toán



— Hãy đánh giá độ phức tạp của thuật toán interchange sort dựa trên hàm cài đặt chuẩn.

```
11. void InterchangeSort(int a[], int n)
12. {
13.     for(int i=0; i<=n-2; i++)
14.         for(int j=i+1; j<=n-1; j++)
15.             if(a[i]>a[j])
16.                 HoanVi(a[i], a[j]);
17. }
```



Độ phức tạp của thuật toán





Thuật toán interchange sort

ĐẶC ĐIỂM – ĐIỂM MẠNH – ĐIỂM YẾU

Đặc điểm – điểm mạnh – điểm yếu



— Đặc điểm thuật toán Interchange sort:

- + Độ phức tạp về thời gian (time complexity): $O(n^2)$.
- + Độ phức tạp về bộ nhớ (space complexity): $O(1)$.
- + Trường hợp xấu nhất (worst case): $O(n^2)$.
- + Trường hợp trung bình (average case): $O(n^2)$.
- + Trường hợp tốt nhất (best case): $O(n^2)$.
- + Ổn định.

Đặc điểm – điểm mạnh – điểm yếu



— Điểm mạnh:

- + Thuật toán rõ ràng, dễ hiểu.
- + Thuật toán dễ cài đặt.
- + Không yêu cầu dung lượng bộ nhớ lớn.



Đặc điểm – điểm mạnh – điểm yếu



— Điểm yếu:

- + Thời gian thực hiện thuật toán lâu.
- + Không nhận biết mảng đã được sắp xếp.



Cảm ơn quý vị đã lắng nghe

Nhóm tác giả

Hồ Thái Ngọc

ThS. Võ Duy Nguyên

TS. Nguyễn Tấn Trần Minh Khang



PROJECT A01 – DỰ ÁN A01

Dự án A01



— Viết chương trình thực hiện các yêu cầu sau:

- + Nhập mảng một chiều từ các tập tin: intdata01.inp; intdata02.inp; ...; intdata09.inp; intdata10.inp; intdata11.inp; intdata12.inp; intdata13.inp;
- + Xuất các cặp giá trị trong mảng ra các tập tin: intdata01.out; intdata02.out; ...; intdata09.out; intdata10.out; intdata11.out; intdata12.out; intdata13.out;

Dự án A01



+ Định dạng tập tin `intdataxx.inp`

- Dòng đầu tiên: số phần tử của mảng (n).
- Dòng tiếp theo: lưu n số nguyên tương ứng với các giá trị trong mảng.

```
*intdata01.inp - Notepad
File Edit Format View Help
10
-91 -65 7 49 24 41 64 -12 -28 15|
```

Dự án A01



+ Định dạng tập tin
intdataxx.out

- Dòng đầu tiên: số lượng cặp giá trị trong mảng (k).
- k dòng tiếp theo: mỗi dòng lưu hai số nguyên tương ứng với hai giá trị trong cặp.

```
intdata01.out - Notepad
File Edit Format View Help
45
-91 -65
-91 7
-91 49
-91 24
-91 41
-91 64
```



PROJECT A02 – DỰ ÁN A02

Dự án A02



— Viết chương trình thực hiện các yêu cầu sau:

- + Nhập mảng một chiều từ các tập tin: intdata01.inp; intdata02.inp; ...; intdata09.inp; intdata10.inp; intdata11.inp; intdata12.inp; intdata13.inp;
- + Xuất các cặp giá trị nghịch thế (tăng) trong mảng ra các tập tin: intdata01.out; intdata02.out; ...; intdata09.out; intdata10.out; intdata11.out; intdata12.out; intdata13.out;

Dự án A02



+ Định dạng tập tin `intdataxx.inp`

- Dòng đầu tiên: số phần tử của mảng (n).
- Dòng tiếp theo: lưu n số nguyên tương ứng với các giá trị trong mảng.

```
*intdata01.inp - Notepad
File Edit Format View Help
10
-91 -65 7 49 24 41 64 -12 -28 15|
```


Dự án A02



+ Định dạng tập tin

intdataxx.out

- Dòng đầu tiên: số lượng cặp giá trị nghịch thế trong mảng (k).
- k dòng tiếp theo: mỗi dòng lưu hai số nguyên tương ứng với hai giá trị trong cặp nghịch thế.

+ Định dạng tập tin

intdataxx.out

- Dòng đầu tiên: số lượng cặp giá trị nghịch thế trong mảng (k).
- k dòng tiếp theo: mỗi dòng lưu hai số nguyên tương ứng với hai giá trị trong cặp nghịch thế.

Dự án A02



```
11. int DemSoCapNghichThe(int a[],int n)
12. {
13.     int dem = 0;
14.     for(int i=0; i<=n-2; i++)
15.         for(int j=i+1; j<=n-1;j++)
16.             if(a[i]>a[j])
17.                 dem++;
18.     return dem;
19. }
```



Interchange sort và mảng một chiều

PROJECT A10 – DỰ ÁN A10

Dự án A11



— Viết chương trình thực hiện các yêu cầu sau:

- + Nhập mảng một chiều từ các tập tin: intdata01.inp; intdata02.inp; ...; intdata09.inp; intdata10.inp; intdata11.inp; intdata12.inp; intdata13.inp;
- + Sắp xếp mảng tăng dần bằng thuật toán interchange sort.
- + Xuất mảng sau khi sắp xếp ra các tập tin: intdata01.out; intdata02.out; ...; intdata09.out; intdata10.out; intdata11.out; intdata12.out; intdata13.out;

Dự án A11



+ Định dạng tập tin `intdataxx.inp` và `intdataxx.out`

- Dòng đầu tiên: số phần tử của mảng (n).
- Dòng tiếp theo: lưu n số nguyên tương ứng với các giá trị trong mảng.

```
*intdata01.inp - Notepad
File Edit Format View Help
10
-91 -65 7 49 24 41 64 -12 -28 15|
```



PROJECT A11 – DỰ ÁN A11

Dự án A11



— Viết chương trình thực hiện các yêu cầu sau:

- + Nhập mảng một chiều từ các tập tin: intdata01.inp; intdata02.inp; ...; intdata09.inp; intdata10.inp; intdata11.inp; intdata12.inp; intdata13.inp;
- + Sắp xếp mảng tăng dần bằng thuật toán interchange sort với biến thể 01.
- + Xuất mảng sau khi sắp xếp ra các tập tin: intdata01.out; intdata02.out; ...; intdata09.out; intdata10.out; intdata11.out; intdata12.out; intdata13.out;

Dự án A11



+ Định dạng tập tin `intdataxx.inp` và `intdataxx.out`

- Dòng đầu tiên: số phần tử của mảng (n).
- Dòng tiếp theo: lưu n số nguyên tương ứng với các giá trị trong mảng.

```
*intdata01.inp - Notepad
File Edit Format View Help
10
-91 -65 7 49 24 41 64 -12 -28 15|
```




PROJECT A12 – DỰ ÁN A12

Dự án A12



— Viết chương trình thực hiện các yêu cầu sau:

- + Nhập mảng một chiều từ các tập tin: intdata01.inp; intdata02.inp; ...; intdata09.inp; intdata10.inp; intdata11.inp; intdata12.inp; intdata13.inp;
- + Sắp xếp mảng tăng dần bằng thuật toán interchange sort với biến thể 02.
- + Xuất mảng sau khi sắp xếp ra các tập tin: intdata01.out; intdata02.out; ...; intdata09.out; intdata10.out; intdata11.out; intdata12.out; intdata13.out;

Dự án A12



+ Định dạng tập tin `intdataxx.inp` và `intdataxx.out`

- Dòng đầu tiên: số phần tử của mảng (n).
- Dòng tiếp theo: lưu n số nguyên tương ứng với các giá trị trong mảng.

```
*intdata01.inp - Notepad
File Edit Format View Help
10
-91 -65 7 49 24 41 64 -12 -28 15|
```



PROJECT A13 – DỰ ÁN A13

Dự án A13



— Viết chương trình thực hiện các yêu cầu sau:

- + Nhập mảng một chiều từ các tập tin: intdata01.inp; intdata02.inp; ...; intdata09.inp; intdata10.inp; intdata11.inp; intdata12.inp; intdata13.inp;
- + Sắp xếp mảng tăng dần bằng thuật toán interchange sort với biến thể 03.
- + Xuất mảng sau khi sắp xếp ra các tập tin: intdata01.out; intdata02.out; ...; intdata09.out; intdata10.out; intdata11.out; intdata12.out; intdata13.out;

Dự án A13



+ Định dạng tập tin `intdataxx.inp` và `intdataxx.out`

- Dòng đầu tiên: số phần tử của mảng (n).
- Dòng tiếp theo: lưu n số nguyên tương ứng với các giá trị trong mảng.

```
*intdata01.inp - Notepad
File Edit Format View Help
10
-91 -65 7 49 24 41 64 -12 -28 15|
```



Cảm ơn quý vị đã lắng nghe

Nhóm tác giả

Hồ Thái Ngọc

ThS. Võ Duy Nguyên

TS. Nguyễn Tấn Trần Minh Khang