

# PROGRAMMING TECHNIQUES

Phan Tân Quốc, Lai Đình Hải, Trịnh Tân Đạt

Email: quocpt@sgu.edu.vn

Website: <https://sites.google.com/site/phantanquoc>

Classroom:

1

## Tài liệu tham khảo

*Giáo trình Kỹ thuật lập trình*, 277 trang, NXB ĐHQG TPHCM, ISBN: 978-604-73-4633-2, 2016, Phan Tân Quốc (đồng tác giả)

2

## Đánh giá

- Điểm thi kết thúc HP: 50% (lý thuyết, 90 phút, tự luận, không sử dụng tài liệu, trường tổ chức thi).
  - Điểm quá trình: 50%
    - Điểm lý thuyết:
      - ✓ Chuyên cần, thảo luận: Điểm +
      - ✓ Kiểm tra lý thuyết lần 1: 15%
      - ✓ Kiểm tra lý thuyết lần 2: 10%
      - ✓ (SV có thể làm bài tiểu luận thay thế điểm lý thuyết 25%)
    - Điểm thực hành:
      - ✓ Chuyên cần, thảo luận: Điểm +
      - ✓ Kiểm tra thực hành 1: 15%
      - ✓ Kiểm tra thực hành 2: 10%

3

## Nội dung

Chương 1. Kỹ thuật lập trình file văn bản (4LT+4TH)

Chương 2. Kỹ thuật tối ưu hóa thuật toán (4LT+4TH)

Chương 3. Kỹ thuật lập trình đê qui (4LT+4TH)

Chương 4. Kỹ thuật lập trình sử dụng con trỏ (4LT+4TH) + kiểm tra lần 1

Chương 5. Kỹ thuật lập trình với chuỗi ký tự-file với chuỗi ký tự (4LT+4TH)

Chương 6. Một số kỹ thuật lập trình nâng cao (4LT+4TH)

kiểm tra lần 2 & Ôn tập thi học kỳ & Báo cáo tiểu luận (6IT+6TH)

Ghi chú: +Nội dung làm tiểu luân (nếu có) nằm trong chương 6

+SV xem thêm thông tin trong classroom của GV

4

# KỸ THUẬT LẬP TRÌNH VỚI FILE VĂN BẢN

5

## NỘI DUNG

- Giới thiệu
- Một số hàm thao tác với file dữ liệu văn bản
- File văn bản với dữ liệu số không có cấu trúc
- File văn bản với dữ liệu số có cấu trúc mảng 1 chiều
- File văn bản với dữ liệu số có cấu trúc mảng 2 chiều  
(File văn bản với dữ liệu chuỗi sẽ được đề cập sau bài KTLT chuỗi)

6

## GIỚI THIỆU

- Chương này trình bày các thao tác cần thiết nhất đối với các loại file văn bản để giải quyết vấn đề lưu trữ dữ liệu bền vững trên đĩa từ.
- Chương này hữu ích cho các vấn đề bài toán cần xử lý dữ liệu vào/ra từ file; hữu ích cho các học phần tiếp theo như: cấu trúc dữ liệu, hệ điều hành, cơ sở trí tuệ nhân tạo,...

7

## KHAI BÁO CON TRỎ FILE

- Khai báo các con trỏ quản lý file như sau:  
`FILE * <filepointer>;`
- Trong đó filepointer là tên biến con trỏ kiểu FILE; và từ lúc này, mọi thao tác tác động lên con trỏ file này thực chất là sẽ tác động lên file dữ liệu tương ứng.

8

# MỘT SỐ HÀM

**Mở tập tin**

```
FILE *fopen (const char * filename, const char *mode);
```

**Đóng tập tin**

```
void fclose (FILE * filepointer);
```

**Ghi dữ liệu vào tập tin**

```
void fprintf (FILE * filepointer,const char formattext, varname);
```

**Đọc dữ liệu từ tập tin**

```
void fscanf (FILE *filepointer, const char formattext, &varname);
```

9

## Ví dụ 1

Cho file văn bản so.inp có cấu trúc chỉ một dòng duy nhất chứa 3 số nguyên; các số cách nhau ít nhất một khoảng trắng.

Hãy tính tổng và trung bình cộng của 3 số trên;

Kết quả ghi ra file ketqua.out

-Dòng đầu ghi giá trị tổng

-Dòng thứ hai ghi giá trị trung bình của 3 số đó.

SO.INP	KETQUA.INP
3 4 6	13
	4.33

10

## Ví dụ 2

Cho file văn bản dayso.inp có cấu trúc như sau:

- Dòng đầu ghi số nguyên dương n;
- Trong các dòng tiếp theo; ghi đủ n số; các số cách nhau một khoảng trắng.

Hãy thực hiện các công việc sau:

- a.Tính tổng các phần tử
- b.Tìm giá trị lớn nhất
- c.Đếm số lượng số nguyên tố.

11

## Ví dụ 2 (...)

d.Đếm số lượng số vừa nguyên tố vừa đối xứng.

e.Tính tổng các chữ số của tất cả các số.

Kết quả ghi ra file dayso.out; mỗi dòng ghi kết quả mỗi câu trên

12

## Test

dayso.inp	dayso.out
10	Cau a: 501
3 7 27	Cau b: 131
4 2 5 131	Cau c: 6
100 101	Cau d: 6
121	Cau e: 42

13

## Lưu ý

Nếu không biết trước kích thước của file thì sử dụng hàm sau để duyệt qua các phần tử của file

Hàm feof(FILE \* filepointer)

```
while (!feof(f))  
{  
...  
}
```

14

### Ví dụ 3

Cho file văn bản bangso.inp có cấu trúc như sau:

- Dòng đầu ghi hai số nguyên dương m và n;
- Trong m dòng tiếp; mỗi dòng ghi n số; các số cách nhau ít nhất một khoảng trắng.

Hãy thực hiện các công việc sau:

- a.Tính tổng các phần tử
- b.Đếm số lượng số nguyên tố
- c.Tính tổng các phần tử trên mỗi dòng

15

d.Tính tổng các phần tử trên mỗi cột

e.Đếm số lượng số chẵn, số lượng số lẻ của bảng.

Kết quả ghi ra file bangso.out; mỗi dòng ghi kết quả mỗi câu trên.

16

## Test

Bangso.inp

4	6					
3	7	5	1	9	3	
0	0	7	9	8	7	
6	2	100	11	200	4	
8	3	7	13	17	29	

Bangso.out

Cau a: 459	
Cau b: 13	
Cau c: 28 31 323 77	
Cau d: 17 12 119 34 234 43	
Cau e: 9 15	

17

## Ví dụ 4

Cho file NUM.INP có cấu trúc như sau:

- Dòng đầu chứa số nguyên dương  $n$  (với  $n \leq 30000$ ).
- Trong các dòng tiếp theo chứa  $n$  số nguyên dương; các số có giá trị  $< 1$  triệu.

Sau đây, khi nói đến file NUM.INP được hiểu là  $n$  số trong file không bao gồm số  $n$  ở dòng đầu tiên.

Hãy viết một chương trình hoàn chỉnh thực hiện các công việc sau đây:

1. Đếm xem file NUM.INP có bao nhiêu cặp số nguyên tố cùng nhau ?  
( $x,y$ ) và ( $y,x$ ) được tính là một cặp;
2. Giá trị nào trong file NUM.INP xuất hiện nhiều lần nhất; đếm số lần xuất hiện tương ứng.
3. Đếm xem file NUM.INP chứa bao nhiêu giá trị khác nhau ?

18

4. Đếm xem file NUM.INP có bao nhiêu số nguyên tố đối xứng ? (số nguyên tố đối xứng là một số nguyên tố bằng trung bình cộng của hai số nguyên tố liền trước và liền sau nó (lưu ý định nghĩa trong bài này không quan tâm đến việc 2 số nguyên tố kề với nó có thuộc dãy hay không))
5. Chuyển đổi mỗi số  $a_i$  về số nguyên tố nhỏ nhất lớn hơn hoặc bằng nó. Hãy tính tổng các số của file NUM.INP sau khi đã chuyển đổi.
6. Tìm một dãy con liên tiếp tăng dài nhất. Xuất chiều dài của dãy tìm được.
7. Tìm cặp số nguyên tố liên tiếp  $(x,y)$  nhỏ hơn hoặc bằng  $n$  sao cho khoảng cách giữa  $x$  và  $y$  là lớn nhất. Xuất khoảng cách lớn nhất đó; khoảng cách bằng  $|x-y|$ ; nếu không có tồn tại cặp số thỏa mãn xuất giá trị 0.

19

### Dữ liệu nhập/xuất từ file

NUM.INP

16

1023 127 4000 12 7 29 3000 10 23 29 29 10 5 3000 31 9

NUM.OUT

Cau 1: ???

Cau 2:

Cau 3:

Cau 4:

Cau 5:

Cau 6:

Cau 7:

20

## Bài tập

### BT1.

Cho tập tin mang.inp chứa các số nguyên mỗi số cách nhau ít nhất một khoảng trắng. Hãy viết chương trình hoàn chỉnh đọc các số nguyên trong mang.inp; sau đó thực hiện các yêu cầu sau:

- Tính tổng các phần tử của mảng.
- Sắp xếp theo các số theo thứ tự không giảm ( $\leq$ )

Kết quả ghi vào tập tin mang.out

**mang.inp**

5 4 1 2 3 1

**mang.out**

16

1 1 2 3 4 5

21

### BT2.

Cho dãy  $n$  số nguyên dương  $a_1, a_2, \dots, a_n$  ( $n \leq 30000, a_i \leq 10^6$ ).

Hãy viết chương trình hoàn chỉnh thực hiện các công việc sau:

- Tìm giá trị lớn nhất, giá trị lớn thứ nhì, giá trị lớn thứ ba trong dãy.
- Tìm dãy con liên tiếp không giảm dài nhất. Xuất chiều dài dãy con tìm được.

Dữ liệu vào được cho từ file songuyen.inp; trong đó

+Dòng đầu ghi số  $n$ .

+Trong các dòng tiếp theo ghi  $n$  số; các số cách nhau ít nhất một khoảng trắng.

22

### BT2 (...)

Kết quả ghi vào file ketqua.out gồm 2 dòng; dòng đầu ghi 3 giá trị tương ứng với câu a, dòng thứ hai ghi một giá trị tương ứng với yêu cầu của câu b.

Ví dụ:

songuyen.inp

8

5 3 1 6 9 6 9 9

ketqua.out

9 6 5

3

23

### BT3

Cho tập tin input.txt chứa một chuỗi ký tự gồm các ký tự chữ cái thường và các ký tự khoảng trắng.

Viết chương trình hoàn chỉnh đọc chuỗi ký tự và cho biết tần suất xuất hiện các ký tự có trong tập tin input.txt (không kể ký tự khoảng trắng).

Kết quả được ghi trong tập tin output.txt theo mô tả như ở ví dụ sau:

input.txt

chao mung cac sinh vien khoa cong nghe thong tin

output.txt

a 3 c 4 e 2 g 4 h 5 i 3 k 1 m 1 n 7 o 4 s 1 t 2 u 1 v 1

24

## BT4

- a.Tạo một file chứa  $n$  số nguyên không âm đôi một khác nhau.
- b.Tạo một file kytu.txt chứa  $n$  ký tự ngẫu nhiên (chỉ chứa chữ cái thường), các ký tự cách nhau ít nhất một khoảng cách; yêu cầu mỗi dòng chứa tối đa 1000 ký tự.
- c.Tạo file chứa  $n$  chuỗi ký tự, mỗi chuỗi trên một dòng, các từ cách nhau ít nhất một khoảng trắng, chiều dài mỗi từ tối đa 7 ký tự; yêu cầu mỗi dòng chứa tối đa 1000 ký tự.

(SV xem cấu trúc file input/output ở ví dụ trong slide tiếp theo. Nếu chọn bài này làm bài đánh giá điểm quá trình thì SV chỉ cần làm 2 trong 3 ý trên; mỗi ý nên viết thành file cpp riêng)

25

BT4a:

file input dat ten la sokhac.inp, vi du:  
sokhac.inp  
6  
file output dat ten la sokhac.txt,vi du:  
sokhac.txt  
6  
4 12 5 2 19 3

BT4b:

file input dat ten la kytu.inp, vi du:  
kytu.inp  
8  
file output dat ten la kytu.txt,vi du:  
8  
a a b e s a s d  
BT4c:  
file input dat ten la chuoi.inp, vi du:  
chuoi.inp  
4  
file output dat ten la chuoi.txt, vi du:  
4  
sdfg fsf fsga sdgg dgasgf fgdgad sdsfg fdhdshf  
fgsfdh  
fg sdfa sdadagd dffdas gdg dsfa agg dg d  
ef wegw fdfee wffege

26

# KỸ THUẬT TỐI ƯU HÓA THUẬT TOÁN

1

## Nội dung

- Thuật toán và độ phức tạp
- Chương trình máy tính mô tả một thuật toán phụ thuộc vào những yếu tố nào ?
- Tối ưu hóa chương trình
- Các phương pháp giải bài cần rèn luyện

2

## THUẬT TOÁN

Thuật toán giải quyết một bài toán đặt ra là một thủ tục xác định bao gồm một dãy hữu hạn các bước cần thực hiện để thu được đầu ra cho một đầu vào cho trước của bài toán.

3

## Độ phức tạp tính toán của thuật toán

- **Định nghĩa:**

Cách đánh giá thời gian thực hiện thuật toán độc lập với máy tính và các yếu tố liên quan đến máy tính như vậy sẽ dẫn tới khái niệm về “cấp độ lớn của thời gian thực hiện thuật toán” hay còn gọi là :*“Độ phức tạp tính toán của thuật toán”*.

- Đánh giá độ phức tạp của thuật toán là đánh giá lượng tài nguyên các loại mà thuật toán đó đòi hỏi sử dụng.
  - Thời gian tính
  - Bộ nhớ

4

## Độ phức tạp tính toán của thuật toán (...)

- Nếu thời gian thực hiện một thuật toán là  $T(n)=cn^2$  với  $c$  là hằng số thì ta nói: Độ phức tạp tính toán của thuật toán này có cấp là  $n^2$  và ta ký hiệu  $T(n)=O(n^2)$  (ký hiệu là chữ **O** lớn).
- Một hàm  $f(n)$  được xác định là  $O(g(n))$  viết là  $f(n)= O(g(n))$  và được gọi là có cấp  $g(n)$  nếu tồn tại một hằng số  $c$  và  $n_0$  sao cho  $f(n) \leq c.g(n)$  khi  $n \geq n_0$ ; nghĩa là  $f(n)$  bị chặn trên bởi  $c.g(n)$  với mọi giá trị của  $n$  từ một thời điểm nào đó.

5

## Xác định độ phức tạp tính toán

- Để xác định độ phức tạp tính toán của một thuật toán bất kỳ có thể dẫn tới những bài toán phức tạp.
- Tuy nhiên trong thực tế, đối với một số thuật toán ta cũng có thể phân tích được bằng một số quy tắc đơn giản sau:
  - Quy tắc cộng
  - Quy tắc nhân

6

## Quy tắc cộng

Giả sử  $T_1(n)$  và  $T_2(n)$  là thời gian thực hiện của hai đoạn chương trình  $P_1$  và  $P_2$  mà  $T_1(n)=O(f(n))$ ;  $T_2(n)=O(g(n))$  thì thời gian thực hiện  $P_1$  rồi  $P_2$  tiếp theo sẽ là  $T_1(n)+T_2(n)=O(\max(f(n),g(n)))$

Ví dụ: Đoạn lệnh sau có độ phức tạp  $O(n)$

for ( $i=1;i\leq n;i++$ )

<công việc>

7

## Quy tắc nhân

Giả sử  $T_1(n)$  và  $T_2(n)$  là thời gian thực hiện của hai đoạn chương trình  $P_1$  và  $P_2$  mà  $T_1(n)=O(f(n))$ ;  $T_2(n)=O(g(n))$  thì thời gian thực hiện  $P_1$  và  $P_2$  lồng nhau sẽ là  $T_1(n)\times T_2(n)=O(\max(f(n)\times g(n)))$ .

Ví dụ: Đoạn lệnh sau có độ phức tạp  $O(m.n)$

for ( $i=1;i\leq m;i++$ )

for ( $j=1;j\leq n;j++$ )

<công việc>

8

## Một số lớp thuật toán

- $O(1)$ ,
- $O(\log n)$ ,
- $O(n)$ ,
- $O(n \log n)$ ,
- $O(n^2)$ ,
- $O(n^3)$ ,
- $O(n^k)$ ,
- $O(a^n)$ .

9

## CHƯƠNG TRÌNH MÁY TÍNH MÔ TẢ MỘT THUẬT TOÁN PHỤ THUỘC VÀO NHỮNG YẾU TỐ NÀO ?

- Kích thước của dữ liệu đầu vào,
- Cấu trúc dữ liệu chọn cài đặt,
- Kỹ thuật lập trình,
- Môi trường lập trình,

10

## TỐI ƯU HÓA CHƯƠNG TRÌNH

- Chiến lược tối ưu hóa cấu trúc
- Chiến lược giải quyết bài toán qua nhiều giai đoạn độc lập (áp dụng quy tắc cộng)
- Quyết định chọn lựa tối ưu hóa về mặt thời gian hay tối ưu hóa về mặt bộ nhớ ?
- Sử dụng chiến lược thiết kế thuật toán hợp lý.

11

## CÁC PHƯƠNG PHÁP GIẢI BÀI CẦN RÈN LUYỆN

- Kỹ năng đọc đề
- Kỹ năng phân loại bài toán
- Kỹ năng phân tích thuật toán
- Kỹ năng làm chủ ngôn ngữ lập trình
- Kỹ năng gõ nhanh
- Kỹ năng test chương trình

12

## Kỹ năng đọc đề

- Kiến thức cơ sở của bài toán
- Các dữ kiện và yêu cầu của bài toán
- Mô tả khuôn dạng dữ liệu vào và ra
- Các hạn chế cho các test của bài toán

## Kỹ năng phân loại bài toán

- Nhanh chóng phân loại các dạng bài toán (Duyệt toàn bộ, chia để trị, tham lam, quy hoạch động, đồ thị, toán học, xử lý xâu, hình học tính toán,...)
- Có thể là kết hợp của nhiều dạng khác nhau

13

## Kỹ năng phân tích thuật toán

- Lời giải bài toán phải đủ nhanh và không sử dụng quá nhiều bộ nhớ
- Lời giải nên càng đơn giản càng tốt
- Sử dụng phương pháp phân tích thuật toán để xác định xem lời giải đưa ra có thỏa mãn giới hạn thời gian và bộ nhớ hay không
- Thông thường tính:  $10^8$  phép tính trong 1 giây
- Luôn nhớ hãy sử dụng phương pháp đơn giản nhất thỏa mãn giới hạn thời gian.

## Kỹ năng làm chủ ngôn ngữ lập trình

- Hãy thành thạo ngôn ngữ lập trình như lòng bàn tay bao gồm cả các thư viện có sẵn
- Nếu đã có sẵn trong thư viện chuẩn thì không cần phải lập trình lại: cài đặt nhanh, không có bug
- Hạn chế: khả năng tùy biến của thư viện không linh hoạt. Rất nhiều phần kỹ thuật xử lý thuật toán không thể gọi trực tiếp thư viện mà phải tùy biến đi.

14

## Kỹ năng gõ nhanh

- Hãy trở thành thợ gõ nhanh và chính xác
- Đừng để gõ lời giải là hạn chế cho việc giải bài
- Khi đánh máy không nhìn bàn phím, mắt nhìn vào màn hình kiểm tra luôn tính đúng đắn của việc gõ, trong lúc đó đầu vẫn có thể suy nghĩ song song các vấn đề tiếp theo.

## Kỹ năng test chương trình

- Phải test để chắc chắn kết quả bài giải là đúng và thỏa mãn giới hạn thời gian.
- Hoặc ít ra là biết lời giải sai nhưng không hiểu tại sao
- Cố gắng phản biện bài giải bằng cách tìm ra phản ví dụ (một dữ liệu vào mà bài giải trả kết quả ra sai, hoặc mất quá nhiều thời gian để tìm ra kết quả).
- Test các biến và dữ liệu lớn
- Viết một thuật toán trực tiếp đơn giản để kiểm tra kết quả chương trình của mình có đúng không với những trường hợp kích thước đầu vào là bé.

15

## Ví dụ 1

Cho dãy  $n$  số nguyên  $\{a\}$  ( $n \leq 1$  triệu). Dãy con liên tiếp là dãy mà thành phần của nó là các thành phần liên tiếp nhau trong  $\{a\}$ , ta gọi tổng của dãy con là tổng tất cả các thành phần của nó. Tìm tổng lớn nhất trong tất cả các tổng của các dãy con của  $\{a\}$

Ví dụ:

$n=7$

4 -5 6 -4 2 3 -7

Kết quả

7

16

## Ví dụ 2

Cho dãy n số nguyên và một số nguyên k với  $1 \leq k < n$  và  $(n \leq 10^9)$ . Hãy chuyển k phần tử đầu dãy về cuối dãy  
Yêu cầu không dung mảng trung gian.

Ví dụ:

$n=9; k=3$

3 6 5 1 2 7 6 9 8

Kết quả:

1 2 7 6 9 8 3 6 5

17

## Ví dụ 3

Cho dãy n số nguyên  $a_1, a_2, \dots, a_n$  ( $n \leq 10^6$ ).

Hãy tìm 3 số sao cho tích của chúng là lớn nhất (xuất tích lớn nhất tìm được).

18

**BT1.**

Cho dãy gồm  $n$  số nguyên  $a_1, a_2, \dots, a_n$  ( $n \leq 10^6$ ). Tìm một dãy con liên tiếp tăng dài nhất. Xuất chiều dài của dãy tìm được.

Ví dụ

8

4 5 10 2 3 9 5

Kết quả:

4

**BT2.**

$n$  được gọi là số nguyên tố đối xứng nếu  $n$  bằng trung bình cộng của 2 số nguyên tố liên kề với nó. Ví dụ 5 là số nguyên tố đối xứng.

a. Hỏi từ 1 đến 1 tỷ có bao nhiêu số nguyên tố ?

b. Hỏi từ 1 đến 1 tỷ có bao nhiêu số nguyên tố đối xứng ?

19

**BT3.**

Nhập vào  $n$  số nguyên  $a_1, a_2, \dots, a_n$  ( $n \leq 30000, |a_i| \leq 10000$ ). Số  $a_p$  ( $1 \leq p \leq n$ ) được gọi là một số trung bình cộng trong dãy nếu tồn tại 3 chỉ số  $i, j, k$  ( $1 \leq i, j, k \leq n$ ) đôi một khác nhau, sao cho  $a_p = (a_i + a_j + a_k)/3$ . Hãy tìm số lượng các số trung bình cộng trong dãy ( $p$  không nhất thiết phải khác các số  $i, j, k$ ).

Ví dụ

$n=5$

2 9 5 7 10

kết quả là 1

20

**BT4.**

Tìm các số  $p, q < 1$  triệu sao cho tổng các ước số không kể chính nó của  $p$  bằng  $q$  và tổng các ước số thực sự của  $q$  bằng  $p$  (ví dụ 6 có các ước 1, 2, 3).

**BT5.**

Cho hai số nguyên lớn  $a$  và  $b$ ; trong đó  $a$  có  $m$  chữ số và  $b$  có  $n$  chữ số. Viết chương trình thực hiện các phép cộng, trừ hai số nguyên lớn.

21

**BT6.**

Cho dãy gồm  $n$  số nguyên  $a_1, a_2, \dots, a_n$  ( $n \leq 1$  triệu) và số  $M$ . Đếm xem trong dãy có bao nhiêu cặp số có tổng bằng  $M$ . Xuất số lượng cặp số tìm được (hạn chế: Giả thiết các số trong dãy  $a_i$  đôi một khác nhau).

**BT7.**

Cho dãy  $n$  số nguyên dương ( $n \leq 10^6$ ; các số có giá trị nhỏ hơn  $10^6$ ). Hỏi dãy trên có bao nhiêu số có giá trị đôi một khác nhau?

22

# KỸ THUẬT LẬP TRÌNH ĐỆ QUI

1

## Nội dung

- Hàm đệ qui
- Phân loại đệ quy
- Khử đệ qui
- Chia để trị
- Ví dụ & Bài tập

2

## HÀM ĐỆ QUY

- Một hàm được gọi là có tính đệ quy nếu trong bản thân hàm đó có lệnh gọi lại chính nó một cách trực tiếp hay gián tiếp.
- Một chương trình được gọi là có đệ quy nếu nó có chứa ít nhất một hàm đệ quy (gọi tắt là chương trình đệ quy).

3

## Sơ đồ hàm đệ quy

Một hàm đệ quy (recursive function) gồm 2 bước:

- **Bước cơ sở** (hay còn gọi là bước dừng):  
Mô tả cấp độ giải được của bài toán.
- **Bước đệ quy:**  
Bước gọi lại chính nó nhưng với cấp độ thấp hơn.

4

## Hàm đệ qui hoạt động như thế nào ?

- Khi một hàm gọi đệ quy đến chính nó, chương trình sẽ tạo ra một tập các biến cục bộ hoàn toàn độc lập với các tập biến cục bộ đã được tạo ra trong các lần gọi trước đó.
- Có bao nhiêu lần gọi tới hàm đệ quy thì cũng có bấy nhiêu lần thoát ra khỏi hàm.
- Cứ mỗi lần thoát ra khỏi hàm thì một tập các biến cục bộ sẽ được giải phóng.
- Sự tương ứng giữa các lần gọi tới hàm và thoát ra khỏi hàm được thực hiện theo thứ tự ngược, nghĩa là lần ra đầu tiên ứng với lần vào cuối cùng và lần ra khỏi hàm cuối cùng ứng với lần đầu tiên gọi tới hàm (cơ chế vào sau ra trước).

5

## PHÂN LOẠI ĐỆ QUY

- Đệ quy tuyến tính,
  - Đệ quy nhị phân,
  - Đệ quy hỗ tương,
  - Đệ quy phi tuyến
- ❖ (việc phân loại này chỉ mang tính hình thức)

6

## Đệ quy tuyến tính

Một hàm được gọi là đệ quy tuyến tính (đệ quy đơn) nếu một lần gọi hàm nó chỉ phát sinh tối đa một lời gọi đệ quy.

7

### Ví dụ 1.

Giai thừa được định nghĩa theo kiểu quy nạp như sau:  $n! = n*(n-1)!$

Viết chương trình tính  $n!$ , với  $n$  là một số nguyên không âm.

```
1. long giaithua(int n)
2. {
3.     if (n==0 || n==1)
4.         return 1;
5.     else
6.         return n*giaithua(n-1);
}
```

8

## Đệ quy nhị phân

Một hàm được gọi là đệ quy nhị phân nếu mỗi lần gọi hàm nó phát sinh một số ít lời gọi đệ quy.

9

### Ví dụ 2.

Viết chương trình tính số hạng thứ n của dãy fibonacci  $f_n$  xác định theo công thức đệ quy sau:

$$\begin{aligned}f_1 &= 1, \\f_2 &= 1, \\f_n &= f_{n-1} + f_{n-2}, \quad n \geq 3.\end{aligned}$$

```
1. int f(int n)
2. {
3.     if (n==1 || n==2)    return 1;
4.     else                  return f(n-1)+f(n-2);
}
```

10

### Ví dụ 3.

Cho một dãy số được định nghĩa theo công thức quy nạp sau (với  $n$  là số nguyên  $\geq 1$ )

$$f(1) = 1; f(2) = 2; f(3) = 3.$$

$$f(n+3) = 2f(n+2) + f(n+1) - 3f(n)$$

Viết chương trình tính  $f(n)$ .

1. long f(int n)

2. {

3.     if ( $n==1 \mid \mid n==2 \mid \mid n==3$ ) return n;

4.     return  $2*f(n-1) + f(n-2) - 3*f(n-3);$

}

11

### Đệ quy hỗ tương

Hai hàm P,Q được gọi là đệ quy hỗ tương nếu hàm P có lời gọi đến hàm Q và ngược lại một cách trực tiếp hay gián tiếp.

### Ví dụ 4.

Viết chương trình tính số hạng thứ  $n$  của hai dãy sau:

$$x_0 = 1, y_0 = 0,$$

$$x_n = x_{n-1} + y_{n-1} \text{ với mọi } n > 0,$$

$$y_n = 3*x_{n-1} + 2*y_{n-1} \text{ với mọi } n > 0.$$

12

```
1. int tinhxn(int n)
2. {
3.     if (n==0) return 1;
4.     return tinhxn(n-1)+tinhyn(n-1);
5. }
6. int tinhyn(int n)
7. {
8.     if (n==0) return 0;
9.     return 3*tinhxn(n-1)+2*tinhyn(n-1);
}
```

13

## Đệ quy phi tuyến

- Một hàm được gọi là đệ quy phi tuyến (đệ quy phức) nếu mỗi lần gọi hàm thì nó phát sinh ra khoảng n lần gọi đệ quy.
- Thông thường với loại này, lời gọi đệ quy được đặt trong một hoặc nhiều vòng lặp.

14

Ví dụ 5.

Dãy  $A_n$  được cho như sau :

$$A_1=1$$

$$A_n=n(A_1+A_2+\dots+A_{n-1})$$

Viết hàm tính  $A_n$  có sử dụng đệ quy

15

```
1. int An(int n)
2. {
3.     if (n==1) return 1 ;
4.     int s=0 ;
5.     for (int i=1;i<n ;i++)
6.         s=s+An(i) ;
7.     return s*n ;
8. }
```

16

## KHỬ ĐỆ QUY

- Không nên lạm dụng đệ quy và nếu một bài toán không quá khó để tìm được một lời giải không đệ quy thì nên chọn cách giải không đệ quy.
- Lời giải không đệ quy được gọi là khử đệ quy.
- Ví dụ việc tính  $n!$ , tính số hạng thứ  $n$  của dãy Fibonacci bằng khử đệ qui là công việc đơn giản.

17

## CHIA ĐỀ TRỊ

- Phân bài toán cần giải thành các bài toán con. Các bài toán con lại được tiếp tục phân thành các bài toán con nhỏ hơn, cứ thế tiếp tục cho tới khi ta nhận được các bài toán con hoặc đã có thuật toán hoặc là có thể dễ ràng đưa ra thuật toán.
- Kết hợp các nghiệm của các bài toán con để nhận được nghiệm của bài toán con lớn hơn, để cuối cùng nhận được nghiệm của bài toán cần giải.
- Thông thường các bài toán con nhận được trong quá trình phân chia là cùng dạng với bài toán ban đầu, chỉ có cỡ của chúng là nhỏ hơn.

18

### SƠ ĐỒ CHIA ĐỀ TRỊ (...)

```
procedure DivideConquer(A,x); // tìm nghiệm x của bài toán A
begin
if (A đủ nhỏ) then Solve(A)
else
begin
Phân rã A thành các bài toán con A1, A2, ..., Am;
for i:=1 to m do DivideConquer(Ai, xi);
Kết hợp các nghiệm xi (i=1,2,...,m) của các bài toán con Ai để nhận được nghiệm
của bài toán A;
end;
end;
```

Trong thủ tục trên, Solve(*A*) là thuật toán bài toán *A* trong trường hợp *A* có cỡ đủ nhỏ.

19

### VD: Tính số hạng thứ *n* của dãy fibonacci

$$\begin{aligned}f_1 &= 1; \\f_2 &= 1 \\f_n &= f_{n-1} + f_{n-2}\end{aligned}$$

20

### VD: Tính $n!$

$$n! = n \times (n-1)!$$

21

### VD6. CHUYỂN K PHẦN TỬ ĐẦU DÃY VỀ CUỐI DÃY

INPUT:

$$n=7; k=3$$

6 5 1 4 8 7 6

OUTPUT:

4 8 7 6 6 5 1

22

## Chuyển k phần tử đầu dãy về cuối dãy

```
void swap(int &x, int &y)
{
    int tmp=x;
    x=y;
    y=tmp;
}

void swaparray(int a[], int u, int v)
{
    while (u<v)
        swap(a[u++], a[v--]);
}
void process(int a[], int n, int k)
{
    swaparray(a,1,k);
    swaparray(a,k+1,n);
    swaparray(a,1,n);
}
```

23

## VD7. TÍNH $a^n$

Cho  $a$  là số thực,  $n$  là số nguyên không âm. Tính  $a^n$

```
function Power(x, n: Integer): Integer;
var
    i: Integer;
begin
    Result := 1;
    for i := 1 to n do Result := Result * x;
end;
```

```
function Power(x, n: Integer): Integer;
begin
    if n = 0 then Result := 1
    else Result := x * Power(x, n - 1);
end;
```

24

$$x^n = \begin{cases} 1, & \text{nếu } n = 0 \\ (x^{n/2})^2, & \text{nếu } n > 0 \text{ và } n \text{ chẵn} \\ (x^{\lfloor n/2 \rfloor})^2 \times x, & \text{nếu } n > 0 \text{ và } n \text{ lẻ} \end{cases}$$

```

function Power(x, n: Integer): Integer;
begin
  if n = 0 then Result := 1
  else
    begin
      Result := Power(x, n div 2);
      Result := Result * Result;
      if n mod 2 = 1 then Result := Result * x;
    end;
end;

```

25

```

// phiên bản C/C++
double xn(double x, int n)
{
  if (n>0)
    if (n%2==0)
    {
      double t=xn(x,n/2);
      return t*t;
    }
    else
    {
      double t=xn(x,(n-1)/2);
      return t*t*x;
    }
  return 1;
}

```

26

## VD8.BÀI TOÁN THÁP HÀ NỘI

Giả sử có 3 cọc A, B, C. Ban đầu tại A đặt một số đĩa với thứ tự đĩa trên nhỏ đĩa dưới to.

Yêu cầu của bài toán là chuyển toàn bộ số đĩa trên sang cọc C, trong quá trình chuyển được phép sử dụng cọc C, mỗi lần chuyển đúng 01 đĩa và luôn bảo đảm nguyên tắc đĩa nhỏ nằm trên đĩa lớn trong suốt quá trình chuyển, đồng thời số lần di chuyển đĩa là ít nhất.

27

```
#include <iostream.h>
void move(int disk, char start, char finish,
          char spare);
int count=0;
int main()
{
    int disk;
    cout<<"Input number of disk: ";
    cin>>disk;
    move(disk, 'A', 'C', 'B');
    cout<<"\nTotal number of moves:
    "<<count<<endl;
}
```

```
void move(int disk, char start, char finish, char spare)
{
    if(disk == 1)
    {
        cout<<"\nMove disk from "<<start<< " to "<<
        finish;
        count++;
        return;
    }
    else{
        move(disk-1, start, spare, finish);
        move(1, start, finish, spare);
        move(disk-1, spare, finish, start);
    }
}
```

28

## KẾT LUẬN

- Chia để trị là một phương pháp thiết kế giải thuật cho các bài toán mang bản chất đệ quy (Chia để trị không nhất thiết phải sử dụng đệ qui),
- Nếu một bài toán mà số bài toán con quá lớn thì việc áp dụng chia để trị sẽ không hiệu quả,
- Chia để trị là một phương pháp thiết kế giải thuật hiệu quả,...

29

Phụ lục về bài toán tìm kiếm và bài toán sắp xếp

### Bài toán tìm kiếm

Cho dãy  $n$  số nguyên  $a_0, a_1, \dots, a_{n-1}$  và một số nguyên  $x$ . Hãy tìm xem  $x$  có thuộc vào dãy số trên hay không ? Nếu tìm được ở vị trí thứ  $i$  thì xuất kết quả là  $i$ , ngược lại nếu không tìm thấy thì xuất kết quả là  $-1$  (chú ý dãy bắt đầu từ chỉ số 0, nếu dãy có nhiều số bằng  $x$  thì xuất vị trí  $i$  nhỏ nhất).

30

## Một số thuật toán tìm kiếm

- Tìm kiếm tuyến tính
- Tìm kiếm nhị phân (yêu cầu dãy phải có thứ tự)

31

### Tìm kiếm tuyến tính

- Bắt đầu từ phần tử thứ nhất  $a[0]$ , ta lần lượt so sánh  $x$  với các giá trị  $a[i]$ .
- Nếu có  $a[i]$  bằng  $x$  thì  $i$  chính là kết quả cần tìm và kết thúc giải thuật.
- Nếu trong dãy không có số  $a[i]$  nào bằng  $x$  thì xuất kết quả là  $-1$  và cũng kết thúc giải thuật.

32

```
int LinearSearch ( int a[], int n, int x )
{
    int i = 0;
    while ( i<n && a[i]!=x)
        i++;
    if( i==n)
        return -1; // tìm hết nhưng không có x
    return i; //   tìm thấy a[i] là phần tử có khóa x
}
```

❖ Giải thuật tìm kiếm tuyến tính có độ phức tạp tính toán là  $O(n)$ .

33

### Tìm kiếm tuyến tính (cải tiến)

Hiệu quả của giải thuật được nâng cao bằng cách đặt thêm phần tử cầm canh (sentinel) ở cuối mảng ( $a[n]=x$ ) để bảo đảm rằng trong dãy  $a[i]$  lúc này luôn có phần tử bằng  $x$  và vòng lặp while luôn kết thúc. Do đó không cần kiểm tra điều kiện ( $i<n$ ) nữa.

34

```

int LinearSearch ( int a[], int n, int x )
{
    int i = 0; // mảng gồm n phần tử từ a[0]...a[n-1]
    a[n] = x; // thêm phần tử thứ n+1
    while (a[i]!=x)
        i++;
    if( i==n)
        return -1; // tìm hết nhưng không có x
    return i; // tìm thấy x tại vị trí i
}

```

35

### Tìm kiếm nhị phân

Giả sử dãy tìm kiếm hiện hành bao gồm các phần tử  $a_{left}, \dots, a_{right}$ .

Gọi  $midle=(left+right)/2$

Nhận xét rằng nếu  $x > a[i]$  thì  $x$  chỉ có thể xuất hiện bên phải  $a[i]$  - nghĩa là trong đoạn  $[a_{middle+1}, a_{right}]$  của dãy, ngược lại nếu  $x < a[i]$  thì  $x$  chỉ có thể xuất hiện bên trái  $a[i]$  - trong đoạn  $[a_{left}, a_{middle-1}]$  của dãy, nhờ vậy giải thuật sẽ thu gọn phạm vi tìm kiếm một cách đáng kể. Mỗi lần so sánh loại được một nửa thông tin không có ích.

❖ Giải thuật tìm kiếm nhị phân có độ phức tạp tính toán là  $O(\log_2 n)$

36

### Cài đặt đệ quy

```
int BinarySearch_Recursive(int a[],int n,int x,int left,int right)
{ if (left>right) return -1;
  int mid=(left+right)/2;
  if (x==a[mid]) return mid;
  if (x<a[mid])
    return BinarySearch_Recursive(a,n,x,left,mid-1);
  return BinarySearch_Recursive(a,n,x,mid+1,right);
}
```

37

### Cài đặt không đệ quy

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

38

## Bài toán sắp xếp

- **Input:**

Cho dãy  $n$  số  $a_0, a_1, a_2, \dots, a_{n-1}$ .

- **Output:**

- Dãy  $n$  số  $a'_0, a'_1, a'_2, \dots, a'_{n-1}$  sao cho  $(a'_0, a'_1, a'_2, \dots, a'_{n-1})$  là một hoán vị của  $(a_0, a_1, a_2, \dots, a_{n-1})$  thỏa điều kiện  $a'_0 \leq a'_1 \leq a'_2 \leq \dots \leq a'_{n-1}$ .

39

## ỨNG DỤNG

- Ứng dụng quan trọng trong khoa học và kỹ thuật,
- Quản trị cơ sở dữ liệu,
- Trong các máy tìm kiếm,
- Sắp xếp là một công đoạn quan trọng hỗ trợ giải quyết một vấn đề tin học hiệu quả,...
- D.Knuth: “*40% thời gian hoạt động của máy tính là dành cho việc sắp xếp*”.

40

## THUẬT TOÁN SẮP XẾP CHỌN (SELECTION SORT)

- Trong trường hợp trung bình, thuật toán Selection sort cần  $n^2/2$  lần so sánh và  $n$  lần đổi chỗ;
- Trong trường hợp xấu nhất, Selection sort cần  $n^2/2$  lần so sánh và  $n-1$  lần đổi chỗ;
- Thuật toán Selection sort có độ phức tạp thời gian tuyến tính đối với các mẫu tin có kích thước lớn và khóa của các mẫu tin có kích thước nhỏ;

41

```
void exch(int &x, int &y)
{
    int temp=x;x=y;y=temp;
}
```

42

```

void Selectionsort(int a[],int n)
{
for(int i = 0;i <n-1; i++)
{
    int min = i;
    for ( int j = i+1; j <n; j++)
        if (a[j] < a[min])
            min = j;
    exch(a[min], a[i]); // nghia la ai la phan tu nho nhat o buoc
    chon thu i
}
}

```

43

## THUẬT TOÁN SẮP XẾP NHANH (QUICK SORT)

- Thuật toán Quick sort do C.A.R. Hoare đề xuất vào năm 1960;
- Đây là thuật toán sắp xếp trên thực tế chạy nhanh nhất hiện nay trong số các thuật toán cỡ  $O(n \log n)$ ;
- Là thuật toán được ứng dụng nhiều trong khoa học và kỹ thuật;
- Thời gian tính trung bình của thuật toán Quick sort cỡ  $O(n \log n)$  và trong trường hợp xấu nhất cỡ  $O(n^2)$ .

44

## QUICK SORT (...)

Cải tiến dành cho thuật toán Quick sort:

- **Khử đệ qui** (-> làm việc với các dãy có kích thước lớn hơn)
- **Tập tin có kích thước nhỏ** (chẳng hạn khi  $| - r \leq q$ ;  $q$  thường được đề nghị là 25) thì nên sử dụng thuật toán Insertion sort hoặc thuật toán Selection sort để sắp xếp dãy (->rút ngắn được thời gian tính khoảng 20%)
- **Chọn phần tử làm mốc phân hoạch tốt hơn** (->rút ngắn được thời gian tính khoảng 5%)

45

```
void quicksort(int a[],int l,int r)
{
    int x=a[(l+r)/2];
    int i=l;
    int j=r;
    do {
        while (a[i]<x) i++;
        while (a[j]>x) j--;
        if (i<=j)
            exch(a[i++],a[j--]);
    }
    while (i<j);
    if (l<j) quicksort(a,l,j);
    if (i<r) quicksort(a,i,r);
}
```

46

**BT1.**

a.Viết hàm tìm ước số chung lớn nhất của hai số tự nhiên  $a, b$ .

b.Cho số nguyên dương  $n$ , hãy viết các hàm thực hiện các công việc sau:  
In các chữ số của  $n$  theo chiều ngược lại; tìm chữ số lớn nhất của số  $n$ ; tính tổng các chữ số của số  $n$ ; đếm số lượng các chữ số của số  $n$ .

c.Viết hàm chuyển đổi một số  $n$  trong hệ đếm thập phân thành số trong hệ đếm cơ số  $b$ .

d.Cho số nguyên dương  $n$ .  
Tính tổng  $s = 1 + 1 \times 2 + 1 \times 2 \times 3 + \dots + 1 \times 2 \times 3 \times \dots \times n$ .

e.Cho số nguyên dương  $n$ .  
Tính tổng  $S = 1 \times 2 + 2 \times 3 \times 4 + \dots + n \times (n+1) \times \dots \times (2n)$ .

47

**BT2.**

Cho hai dãy số  $\{x_n\}, \{y_n\}$ ; trong đó  $x_0 = 1; y_0 = 2$ ; và nếu  $n \geq 1$  thì  $\{x_n\}, \{y_n\}$  được cho theo quy luật sau:

$$x_n = \frac{x_{n-1}}{3} + \frac{y_{n-1}}{2} + 27; \quad y_n = \frac{x_{n-1}}{5} + \frac{y_{n-1}}{7} + 11$$

- a. Hãy tính  $x_3$  và  $y_3$
- b. Viết hàm tính  $x_n, y_n$  bằng cách sử dụng đệ qui.
- c. Viết hàm tính  $x_n, y_n$  bằng cách không sử dụng đệ qui và cũng không sử dụng biến mảng. Yêu cầu thuật toán có độ phức tạp là  $O(n)$ .

**BT3.**

Cho  $f_1=1; f_2=1$ ; và  $f_n=f_{n-1}+f_{n-2}$  với mọi  $n > 2$ .

Hãy tính:

$$S_n = \frac{1}{1+f_1} + \frac{2}{1+f_2} + \dots + \frac{n}{1+f_n} \text{ (với } n \text{ là số nguyên dương).}$$

- a. Hãy tính  $S_5$
- b. Viết hàm tính  $S_n$  bằng cách sử dụng đệ qui ( $f_n$  cũng được viết đệ qui).
- c. Viết hàm tính  $S_n$  bằng cách không sử dụng đệ qui và cũng không dùng biến mảng ( $f_n$  cũng được viết không sử dụng đệ qui và cũng không dùng biến mảng).

48

#### BT4.

Dãy số  $x_n$  được định nghĩa như sau:

$$x_0 = 1; x_1 = 1;$$

$$x_n = nx_0 + (n-1)x_1 + (n-2)x_2 + (n-3)x_3 + \dots + x_{n-1}, \forall n > 1.$$

a. Tính  $x_7$ .

b. Viết hàm tính giá trị của  $x_n$ .

#### BT5.

Cho dãy  $x_n$  được định nghĩa như sau:

$$x_1 = 1;$$

$$x_n = n(x_1 + x_2 + x_3 + \dots + x_{n-1}), \forall n > 1.$$

a. Tính  $x_6$  theo công thức trên.

b. Hãy trình bày thuật toán tính  $x_n$ .

c. Hãy cho biết độ phức tạp của thuật toán đã đề xuất ở câu b.

49

#### BT6.

Bài toán xếp Hậu: Liệt kê tất cả các cách xếp  $n$  quân Hậu trên bàn cờ  $n \times n$  sao cho chúng không khống chế lẫn nhau.

#### BT7. (chia để trị)

Cho dãy  $n$  số nguyên. Tìm giá trị lớn nhất của dãy số.

#### BT8. (chia để trị)

Cho mảng  $n$  số nguyên, hãy tìm  $\text{Diff}(a[1..n]) = a_j - a_i$  đạt giá trị lớn nhất mà  $1 \leq i \leq j \leq n$ . Ví dụ mảng có 6 số 4, 2, 5, 8, 1, 7 thì kết quả là 6.

50

# KỸ THUẬT LẬP TRÌNH CON TRỎ

1

## Nội dung

1. Địa chỉ và con trỏ
2. Sử dụng con trỏ trong các biểu thức
3. Thao tác trên con trỏ
4. Con trỏ với mảng một chiều
5. Con trỏ với mảng hai chiều
6. Tổ chức dữ liệu dạng danh sách liên kết

2

## Địa chỉ và con trỏ

- Địa chỉ ô nhớ: Mỗi biến chiếm một số ô nhớ, địa chỉ ô nhớ được biểu thị ở hệ đếm 16
- Con trỏ: là một biến, trỏ đến địa chỉ của một ô nhớ
- Khai báo con trỏ: <kiểu dữ liệu> \*p
- Phép lấy địa chỉ của một biến con trỏ p: &p
- Phép toán lấy giá trị tại một địa chỉ trỏ bởi p: \*p

3

## Sử dụng con trỏ trong các biểu thức

Nếu p là con trỏ trỏ tới biến x thì các cách viết x và \*p là tương đương (đồng nhất) nhau.

4

## Thao tác trên con trỏ

- Cấp phát động bộ nhớ: new
- Giải phóng khôi nhớ đã được cấp phát: delete

5

## Con trỏ với mảng 1 chiều

+ Các cách viết sau đây là tương đương:

a,              &a[0];  
a+i,            &a[i];  
\*(a+i),        a[i];

6

## Con trỏ với mảng 2 chiều

- Phần tử ở dòng  $i$  cột  $j$  của mảng  $a$  phải được truy xuất bởi  $*(a + n*i + j)$

7

## Tổ chức dữ liệu dạng danh sách liên kết

- Với cách tổ chức dữ liệu như mảng một chiều hoặc mảng hai chiều thì mối liên kết giữa các phần tử là một đặc trưng quan trọng của cấu trúc dữ liệu mảng; nó thuận tiện cho nhiều bài toán trong thực tế; nhưng cũng bất lợi trong việc xử lý nhiều bài toán khác; chẳng hạn như các bài toán chèn/xóa phần tử trong dãy,...

8

## Tổ chức dữ liệu dạng danh sách (...)

- Mục này trình bày cách tổ chức dữ liệu linh hoạt hơn, đó là tổ chức dữ liệu dạng danh sách liên kết; mỗi phần tử của danh sách được trỏ bởi một con trỏ; các khồi nhớ được cấp cho một danh sách không nhất thiết phải liên tục nhau.
- Một phần tử của danh sách có hai thành phần:
  - Phần thứ nhất lưu trữ các thông tin về bản thân phần tử,
  - Phần thứ hai lưu trữ địa chỉ của phần tử kế tiếp trong danh sách, hoặc lưu trữ giá trị NULL nếu là phần tử cuối danh sách.

9

## Ví dụ 1

Hãy cho biết kết quả của chương trình sau:

```
1. #include <iostream.h>
2. int main()
3. {
4.     int ar[]={12,10,11,18,13,6},*p;
5.     p=&ar[0];
6.     printf("%d ",*p);
7.     p=&ar[3];
8.     printf("%d ",*p);
9.     *p=*p+100;
10.    printf("%d ",ar[3]);
11.    p=p-2;
12.    printf("%d ",*p);
13. }
```

10

## Ví dụ 2 (với mảng 1 chiều)

Cho dãy n số nguyên. Hãy thực hiện các công việc sau:

- a.Tìm giá trị lớn nhất của dãy.
- b.Sắp xếp các phần tử của dãy theo thứ tự tăng dần.

11

```
1. #include <alloc.h>
2. #include <iostream.h>
3. void nhap(int *&a, int &n);
4. void xuat(int *a, int n);
5. int giatrilonnhat(int *a, int n);
6. void swap(int *a, int *b);
7. void sapxep(int *a, int n);
8. int main()
9. {int *a,n;
10. nhap(a,n);
11. cout<<giatrilonnhat(a,n)<<endl;
12. sapxep(a,n);
13. xuat(a,n);
14. delete a;
15. }
```

12

```
16. void nhap(int *&a, int &n)
17. {
18.     cout<<"Nhập số phần tử của mảng :"; cin>>n;
19.     a=new int [n];
20.     for (int i=0;i<n;i++)
21.         cin>>*(a+i);
22. }
23. void xuat(int *a, int n)
24. {
25.     for (int i=0;i<n;i++)
26.         cout<<*(a+i)<<" ";
27. }
```

13

```
28. int giatrilonnhat(int *a, int n)
29. {
30.     int max=*a;
31.     for (int i=1;i<n;i++)
32.         if (* (a+i) > max)
33.             max=*(a+i);
34.     return max;
35. }
```

14

```

36. void swap(int *a, int *b)
37. {
38.     int t=*a;
39.     *a=*b;
40.     *b=t;
41. }
42. void sapxep(int *a, int n)
43. {
44.     for (int i=0;i<n-1;i++)
45.         for (int j=i+1;j<n;j++)
46.             if (* (a+i)>* (a+j))
47.                 swap(a+i,a+j);
48. }
```

15

```

1. #include <alloc.h>
2. #include <iostream.h>
3. void nhap(int *&a, int &n);
4. void noimang1(int *a, int *b, int *&c,int n, int m);
5. int *noimang2(int *a, int *b, int *c,int n, int m);
6. void xuat(int *a, int n);
7. int main()
8. {
9.     int *a,*b,*c,n,m;
10.    nhap(a,n);
11.    nhap(b,m);
12.    noimang1(a,b,c,n,m);
13.    int *d=noimang2(a,b,c,n,m);
14.    xuat(c,n+m);
15.    xuat(d,n+m);
16.    delete a;
17.    delete b;
18.    delete c;
19.    delete d;
20. }
```

16

```
21. void nhap(int *&a, int &n)
22. {
23.     cout<<"Nhập số phần tử của mảng :"; cin>>n;
24.     a=new int [n];
25.     for (int i=0;i<n;i++)
26.         cin>>*(a+i);
27. }
28. void xuat(int *a, int n)
29. {
30.     for (int i=0;i<n;i++)
31.         cout<<*(a+i)<<" ";
32.     cout<<endl;
33. }
```

17

```
//Cách 1: Trả về biến con trả c
34. void noimang1(int *a, int *b, int *&c,int n, int m)
35. {c=new int [n+m];
36. int t=0;
37. for (int i=0;i<n;i++)           c[t++]=a[i];
38. for (int i=0;i<m;i++)           c[t++]=b[i];
39. }
```

18

```
//Cách 2: Trả về hàm kiểu con trả
40. int *noimang2(int *a, int *b, int *c,int n, int m)
41. {
42.     c=new int [n+m];
43.     int t=0;
44.     for (int i=0;i<n;i++)
45.         c[t++]=a[i];
46.     for (int i=0;i<m;i++)
47.         c[t++]=b[i];
48.     return c;
49. }
```

19

### Ví dụ 3 (với mảng 2 chiều)

Viết chương trình nhập mảng hai chiều a có m dòng n cột, các phần tử là các số nguyên và một số nguyên x.

- a.Hãy đếm xem trong mảng có bao nhiêu số bằng x.
- b.Cho biết vị trí của các phần tử bằng x.

20

```
1. #include<iostream.h>
2. #include<alloc.h>
3. void nhap(int *&a, int &m, int &n, int &x);
4. void demsobangx(int *a, int m, int n, int x);
5. int main()
6. {
7.     int *a,m,n,x;
8.     / nhap(a,m,n);
9.     demsobangx(a,m,n,x);
10.    delete a;
11. }
```

21

```
12. void nhap(int *&a, int &m, int &n, int &x);
13. {
14.     cout<<"Nhập vào số dòng:";      cin>>m;
15.     cout<<"Nhập vào số cột:";      cin>>n;
16.     a=new int [m*n];
17.     for (int i=0;i<m;i++)
18.         for (int j=0;j<n;j++)
19.             cin>>*(a+n*i+j);
20.     cout<<"Nhập vào giá trị x:";  cin>>x;
21. }
```

22

```
22. void demsobangx(int *a, int m, int n, int x)
23. {
24.     int d=0;
25.     for (int i=0;i<m;i++)
26.         for (int j=0;j<n;j++)
27.             if (x==*(a+n*i+j))
28.             {
29.                 d++;
30.                 cout<<i<<" "<<j<<endl;
31.             }
32.     cout<<"So lan xuat hien cua x la : "<<d;
33. }
```

23

#### Ví dụ 4 (tổ chức theo kiểu liên kết)

Nhập vào một danh sách các số nguyên, xuất các số nguyên đó lên màn hình.

24

```
1. #include <iostream.h>
2. struct node
3. {
4.     int info;
5.     struct node *next;
6. };
```

25

```
7. structlist
8. {
9.     node *head,*tail;
10. };
11. void themnut(list &l,node *p);
12. void nhapdanh sach(list &l);
13. void xuatdanh sach(list l);
14. list l;
15. int main()
16. {
17.     nhapdanh sach(l);
18.     xuatdanh sach(l);
19. }
```

26

```
20. void themnut(list &l,node *p)
21. {
22.     if (l.head==NULL)
23.     {
24.         l.tail=p;
25.         l.head=p;
26.     }
27.     else
28.     {
29.         l.tail->next=p;
30.         l.tail=p;
31.     }
32. }
```

27

```
33. void nhapdanh sach(list &l)
34. {
35.     int x,n;
36.     cout<<"Danh sach co bao nhieu phan tu : ";cin>>n;
37.     for (int i=1;i<=n;i++)
38.     {    cin>>x;
39.         node *p=new node;p->info=x;    p->next=NULL;
40.         themnut(l,p);
41.     }
42. }
```

28

```
43. void xuatdanh sach(list l)
44. {
45.     cout<<endl;
46.     node *p = l.head;
47.     while (p!=NULL)
48.     {      cout<<p->info<<" ";
49.         p=p->next;
50.     }
51. }
```

29

## BT1.

Hãy cho biết kết quả của chương trình sau:

```
#include <iostream.h>
int main()
{ int ar[]={2,0,4,8,3,-4},*p;
  p=&ar[0];
  printf("%d ",*p);
  p=&ar[3];
  printf("%d ",*p);
  *p=*p+200;
  printf("%d ",ar[3]);
  p=p-2;
  printf("%d ",*p);
}
```

30

BT2. Sử dụng cấp phát động viết chương trình hoàn chỉnh tìm giá trị lớn nhất của mảng một chiều gồm  $n$  phần tử; trong đó các phần tử là các số nguyên.

BT3. Sử dụng cấp phát động viết chương trình hoàn chỉnh tìm giá trị lớn nhất của mảng hai chiều  $m$  dòng  $n$  cột; trong đó các phần tử là các số nguyên.

31

#### BT4.

a. Cho danh sách liên kết  $I$ , mỗi phần tử là một số nguyên. Hãy tính tổng các phần tử của  $I$ .

b. Cho danh sách liên kết  $I$ , mỗi phần tử là một số nguyên. Hãy kiểm tra xem danh sách  $I$  có chứa giá trị  $x$  không? Nếu có trả về 1, nếu không có trả về 0.

c. Cho danh sách liên kết  $I$ , mỗi phần tử là một số nguyên. Hãy sắp xếp các phần tử theo chiều tăng dần.

32

### BT5

Dùng danh sách liên kết đơn để biểu diễn một đa thức  $P(a_i, n, x)$ . Viết chương trình thực hiện các công việc sau:

- a.Nhập một đa thức
- b.Xuất một đa thức
- c.Tính giá trị của một đa thức tại điểm  $x_0$
- d.Cộng hai đa thức, xuất đa thức kết quả lên màn hình

# KỸ THUẬT LẬP TRÌNH KÝ TỰ VÀ CHUỖI

1

## NỘI DUNG

- KÝ TỰ VÀ CHUỖI KÝ TỰ
- CÁC HÀM XỬ LÝ CHUỖI
- CÁC HÀM XỬ LÝ KÝ TỰ

2

## KÝ TỰ VÀ CHUỖI KÝ TỰ

- Ký tự được đặt giữa hai dấu nháy đơn
- Chuỗi ký tự được đặt trong dấu nháy kép
- Chuỗi ký tự là một dãy các ký tự liên tiếp kết thúc bằng ký tự NULL('\'0')

3

## string.h

- `char *gets(char * str);`
- `char *puts (char * str);`
- `unsigned *strlen(const char *str);`
- `char* strcpy(char *dest, const char * src);`
- `int strcmp(char *s1, char *s2);`

4

- `char * strrev(char *st);`
- `strlwr(char *s);`
- `strupr(char *s);`
- `strcat(char *s1, char *s2);`
- `strchr(char *s,char c);`
- `strrchr(char *s,char c);`
- `strstr(char *s1, char *s2);`
- `void flushall();`

5

## ctype.h

- `void putchar (char c);`
- `void getchar (char c);`
- `int isalnum(int c);`
- `int isalpha(int c);`
- `int isdigit(int c);`

6

- int islower (int c);
- isupper(int c);
- int isspace(int c);
- int tolower(int c);
- int toupper(int c);
- int toascii(int c);

7

## Lưu ý

- Với chuỗi  $s$ , cú pháp  $s + k$  sẽ trả về chuỗi tính từ ký tự thứ  $k$  trở về cuối của chuỗi  $s$ .
- Với chuỗi  $s$ , cú pháp  $s[k] = '0'$  trả về chuỗi  $s$  trong đó  $s$  được kết thúc ở vị trí  $k$ .

8

## NHẬP/XUẤT KÝ TỰ

- **Hàm getchar();//**Dùng để đọc một ký tự từ bàn phím và trả về ký tự đó.
- **Hàm getch();//**Có cùng chức năng như hàm getchar(), nhưng getch() nhập ký tự mà không hiện ra màn hình, trong khi getchar() nhập ký tự và ký tự đó có hiện ra màn hình.
- **Hàm putchar();//** Xuất một ký tự ra màn hình.

9

## Mảng chuỗi

```
#include<iostream.h>
#define maxn 1000
int main()
{
    char s[1000][256];
    for (int i=0;i<3;i++)
        gets(s[i]);
    for (int i=0;i<3;i++)
        cout<<s[i]<<" ";
    return 0;
}
```

10

## Ví dụ 1.

- Nhập vào 2 chuỗi và sau đó xuất 2 chuỗi vừa nhập lên màn hình.

```
1. #include<iostream.h>
2. #include<string.h>
3. #include<alloc.h>
4. int main()
5. {
6.     char *s1;
7.     s1=new char[1000];
8.     gets(s1);
9.     char *s2;
10.    s2=new char[1000];
11.    gets(s2);
12.    cout<<s1<<endl; // có thể sử dụng puts(s1);
13.    cout<<s2<<endl; // có thể sử dụng puts(s2);
14.    delete s1;
15.    delete s2;
16. }
```

11

## Ví dụ 2.

Nhập từ bàn phím một chuỗi s; giả thiết thêm rằng chuỗi không có khoảng trắng dư thừa ở đầu và cuối chuỗi và giữa các từ có duy nhất một khoảng trắng. Hãy viết các hàm thực hiện các công việc sau (các công việc là độc lập với nhau).

- Tách một từ bên phải (bên trái) của chuỗi s.
- Đưa các ký tự đầu của mỗi từ thành chữ hoa, còn các ký tự khác thành chữ thường.
- Đếm xem chuỗi s có bao nhiêu từ,
- Đếm xem chuỗi s có bao nhiêu từ bắt đầu bằng ký tự nguyên âm (các ký tự a, u, i, o, e là các nguyên âm).

12

- e. Đếm số ký tự của mỗi từ của chuỗi s.
- f. Sắp xếp các ký tự của chuỗi s theo chiều tăng (theo mã ASCII), trong đó các ký tự khoảng trắng giữ nguyên vị trí.
- g. Cho biết lần số xuất hiện của các chữ cái trong chuỗi s (không kể ký tự trống).
- h. Đếm số lần xuất hiện chuỗi y trong chuỗi s.

13

```
18. char *tachtutrai(char *s)
19. {
20.     return strrev(strrchr(strrev(s), ' ') + 1);
21. }
22. char *tachtuphai(char *s)
23. {
24.     return strrchr(s, ' ') + 1;
25. }
```

14

```
26. char *chuanhoatu(char *s)
27. {
28.     strlwr(s);
29.     for (int i=0;i<strlen(s);i++)
30.         if (s[i]==' ' && s[i+1]!=' ')
31.             s[i+1]=toupper(s[i+1]);
32.     if (s[0]!=' ')
33.         s[0]=toupper(s[0]);
34.     return s;
35. }
```

15

```
36. int demsotu(char *s)
37. {
38.     int l=strlen(s),d=1;
39.     for (int i=0;i<l;i++)
40.         if (s[i]==' ' && s[i+1]!=' ')
41.             d++;
42.     return d;
43. }
```

16

```

44. int demtubatdaunguyenam(char s[])
45. {
46.     int l=strlen(s),d=0;
47.     for (int i=0;i<l;i++)
48.         if (s[i]==' ' && (s[i+1]=='a' || s[i+1]=='u' ||
s[i+1]=='i' ||
49.             s[i+1]=='o' || s[i+1]=='e'))
50.             d++;
// Kiểm tra ký tự đầu tiên
51.     if (s[0]=='a' || s[0]=='u' || s[0]=='i' || s[0]=='o' ||
s[0]=='e')
52.         d++;
53.     return d;
54. }

```

17

```

55. void demkytucuamoitu(char *s)
56. {
57.     int sokytumoitu=0;
58.     for (int i=0;i<strlen(s);i++)
59.         if (s[i]!=' ') sokytumoitu++;
60.     else
61.     {
62.         cout<<sokytumoitu<<" ";
63.         sokytumoitu=0;
64.     }
65.     cout<<sokytumoitu;
66. }

```

18

```
67. void sapxepkytutang(char *s)
68. {
69.     for(int i=0;i<strlen(s)-1;i++)
70.         for(int j=i+1;j<strlen(s);j++)
71.             if(* (s+i)>* (s+j) && *(s+i) !=' ' && *(s+j) !=' ')
72.             {
73.                 char temp=* (s+i);
74.                 *(s+i)=*(s+j);
75.                 *(s+j)=temp;
76.             }
77. }
```

19

```
78. void tansocackytu( char *s)
79. {
80.     int l=strlen(s);
81.     int d[255];
82.     for (int i=0;i<l;i++)
83.         d[s[i]]=0;
84.     for (i=0;i<l;i++)
85.         d[s[i]]++;
86.     for (i=0;i<l;i++)
87.         if (d[s[i]]!=0)
88.         {
89.             cout<<s[i]<<"    xuat    hien    "<<d[s[i]]<<"    lan"
90.             <<endl;
91.         }
92. }
```

20

```

93. int demchuoiicon(char *s, char *y)
94. {
95.     int d=0;
96.     /    while (strstr(s,y) !=NULL)
97.     {
98.         d=d+1;
99.         strcpy(s,strstr(s,y)+1);
100.    }
101. return d;
102. }

```

21

### Ví dụ 3.

Nhập từ bàn phím một chuỗi; giả thiết chuỗi không có khoảng trắng dư thừa ở đầu và cuối chuỗi và giữa các từ có duy nhất một khoảng trắng. Hãy viết các hàm thực hiện các công việc sau (các công việc là độc lập với nhau).

- Tìm từ có k ký tự, với k=1..7.
- Tạo các chuỗi con được ghép từ từ đầu tiên và từ cuối cùng của mỗi chuỗi, giữa hai từ này có một khoảng trắng.
- Đào ngược chuỗi. Ví dụ chuỗi nhập vào là “truong dai hoc sai gon” thì kết quả sẽ là “gon sai hoc dai truong”.

22

- d. Đảo ngược các ký tự trong mỗi từ. Ví dụ chuỗi nhập vào là “truong dai hoc sai gon” thì kết quả sẽ là “gnourt iad coh ias nog”.
- e. Sắp xếp các từ theo chiều tăng. Ví dụ chuỗi nhập vào là “truong dai hoc sai gon” thì kết quả sẽ là “dai gon hoc sai truong”.
- f. Tìm một từ dài nhất của chuỗi. Ví dụ chuỗi nhập vào là “truong dai hoc sai gon” thì kết quả sẽ là “truong”.

23

```

18. int demsotukkytu(char *s, int k)
19. {
20.     int demkytu=0,demtukkytu=0;
21.     for (int i=0;i<strlen(s);i++)
22.         if (s[i]!=' ') demkytu++;
23.     else
24.         {
25.             if (demkytu==k)
26.                 demtukkytu++;
27.             demkytu=0;
28.         }
29.     return demkytu==k?demtukkytu+1:demtukkytu;//??
30. }
```

24

```
31. void demtuloaik(char *s)
32. {
33.     for (int i=1;i<=7;i++)
34.         cout<<demsootukkytu(s,i)<<" ";
35. }
```

25

```
36. char *gheptu(char *s)
37. {
38.     char stemp[256];
39.     strcpy(stemp,s);
40.     int k=strchr(s,' ')-s;
41.     s[k]='\0';
42.     return strcat(s,strrchr(stemp,' '));
43. }
```

26

```

44. char *daonguocchuoi(char *s)
45. {
46.     strcat(strrev(s), " ");
47.     int i=0;
48.     while (i<strlen(s))
49.     {
50.         int j=i;
51.         while (s[j]!=' ')j++;
52.         {
53.             int u=i,v=j-1;
54.             while (u<v) // dao doan u,v cua chuoi
55.             {
56.                 char ch=s[u];
57.                 s[u]=s[v];
58.                 s[v]=ch;
59.                 u++;
60.                 v--;
61.             }
62.         }
63.         i=j+1;
64.     }
65.     s[strlen(s)-1]='\0';
66.     return s;
67. }
```

27

```

68. char *daonguoctu(char *s)
69. {
70.     strcat(s, " ");
71.     int i=0;
72.     while (i<strlen(s))
73.     {
74.         int j=i;
75.         while (s[j]!=' ')j++;
76.         {
77.             int u=i,v=j-1;
78.             while (u<v) // dao doan u,v cua chuoi
79.             {
80.                 char ch=s[u];
81.                 s[u]=s[v];
82.                 s[v]=ch;
83.                 u++;
84.                 v--;
85.             }
86.         }
87.         i=j+1;
88.     }
89.     s[strlen(s)-1]='\0';
90.     return s;
91. }
```

28

```
92. char *tutrai(char *s)
93. {
94.     int k=strchr(s, ' ') - s;
95.     s[k]='\0';
96.     return s;
97. //if (s[k-1]==10)    s[k-1]='\0';
98. }
```

29

```
99. int demtu(char *s)
100. {
101.     s=strcat(s, " ");
102.     int d=0;
103.     for (int i=0;i<strlen(s);i++)
104.         if (s[i]==' ') d++;
105.     return d;
106. }
```

30

```

107. char *sapxep(char *s)
108. {
109.     char *stemp;
110.     stemp=new char[256];
111.     int d=0;
112.     int sotu=demtu(s);
113.     char *strsub[100];
114.     while (d<sotu)
115.     {
116.         strcpy(stemp,s);
117.         strsub[d]=tutrai(s);
118.         strcpy(s,stemp);
119.         s=s+strlen(tutrai(s))+1;
120.         d++;
121.     }
122.     for (int i=0;i<sotu-1;i++)
123.     for (int j=i+1;j<sotu;j++)
124.     if (strcmp(strsub[i],strsub[j])>0)
125.     {
126.         stemp=strsub[i];
127.         strsub[i]=strsub[j];
128.         strsub[j]=stemp;
129.     }
130.     strcpy(s,"");
131.     for (int i=0;i<sotu;i++)
132.     {
133.         strcat(s,strsub[i]);
134.         strcat(s," ");
135.     }
136.     s[strlen(s)-1]='\0';
137.     return s;// cung co the viet tra ve void neu ...
138. }

```

31

```

139. char *tudainhat(char *s)
140. {
141.     strcat(s," ");
142.     char *skq;
143.     skq=new char [256];
144.     int i=0,max=0;
145.     while (i<strlen(s))
146.     {
147.         int j=i;
148.         while (s[j]!=' ')j++;
149.         if (j-i>max)
150.         {
151.             max=j-i+1;
152.             int u=0;
153.             for (int k=i;k<=j;k++)
154.                 skq[u++]=s[k];
155.             }
156.             i=j+1;
157.         }
158.         skq[max-1]='\0';
159.         return skq;
160. }

```

32

# FILE VĂN BẢN VỚI DỮ LIỆU KÝ TỰ VÀ CHUỖI

33

## HÀM XỬ LÝ KÝ TỰ

### **Hàm fputc(int ch, FILE \*filepointer) ;**

- ch là một giá trị nguyên không dấu, filepointer là con trỏ file.
- Hàm này ghi lên file filepointer một ký tự có mã bằng  $m=ch\%255$ .

### **Hàm fgetc(FILE \*filepointer)**

- Hàm đọc một ký tự từ file filepointer. Nếu thành công hàm cho mã đọc được (có giá trị từ 0 đến 255).

34

## Ví dụ 4

Đọc từng ký tự trong file có tên là filechar.inp và xuất ra màn hình.

```
1. #include <stdio.h>
2. int main()
3. {
4.     FILE *f;
5.     char ch;
6.     f=fopen("filechar.inp","rt");
7.     while (!feof(f))
8.     {
9.         ch=fgetc(f);
10.        putchar(ch);
11.    }
12.    fclose(f);
13. }
```

35

## HÀM XỬ LÝ CHUỖI KÝ TỰ

**int fputs(const char\*s, FILE \* filepointer);**

- s là con trỏ trả về đến địa chỉ đầu của một chuỗi ký tự kết thúc bằng '\0'.
- Lệnh này có tác dụng ghi chuỗi s lên file filepointer.

36

## **char \*fgets(char \*s, int n, FILE \*filepointe)**

- s là con trỏ kiểu char trỏ tới một vùng nhớ đủ lớn để chứa chuỗi ký tự đọc từ file.
- n là số nguyên xác định độ dài cực đại của dãy cần đọc.
- Lệnh này có tác dụng đọc một dãy ký tự từ file filepointe chứa vào vùng nhớ s.

37

## **Ghi chú khi sử dụng lệnh fgets**

- Việc đọc sẽ kết thúc khi:
  - Hoặc đã đọc n-1 ký tự;
  - Hoặc gặp dấu xuống dòng (mã 10); khi đó mã 10 được đưa vào chuỗi kết quả hoặc gặp dấu kết thúc file ()�.
- File có nhiều chuỗi:
  - Mỗi chuỗi có 1 ký tự cuối dòng;
  - Chuỗi cuối file chỉ có ký tự kết thúc file mà không có ký tự kết thúc dòng.

38

## Ví dụ 5

Đọc một file chuoi.inp chứa mỗi chuỗi trên một dòng.

```
1. #include <stdio.h>
2. int main()
3. { char *s;
4. FILE *f;
5. s=new char [80];
6. f=fopen("chuoi.inp","rt");
7. while (!feof(f))
8. {
9.     fgets(s,80,f);
10.    puts(s);
11. }
12. fclose(f);
13. }
```

39

## Ví dụ 6

- VTC nhập n chuỗi và ghi vào file "str.txt"

```
#include<iostream.h>
int main()
{
    char s[256],n;
    cout<<"Nhập số chuỗi n = "; cin>>n;
    FILE *f=fopen("str.txt","wt");
    for (int i=0;i<n;i++)
    {
        fflush(stdin);
        gets(s);
        fputs(s,f);fputs("\n",f);
    }
    fclose(f);
    return 0;
}
```

40

## Ví dụ 7

Đọc file “str.txt” mỗi dòng chứa một chuỗi; thực hiện các yêu cầu sau và ghi kết quả vào “str.out”

- a.Tách từ bên phải của mỗi chuỗi
- b.Đếm số lượng ký tự khác khoảng trắng của mỗi chuỗi
- c.Đếm số lượng ký tự của mỗi từ trong mỗi chuỗi

41

```
#include<iostream.h>
char *tachtuphai(char []);
int demsokytu(char *s);
void demkytucuamoitu(char *s);
FILE *fi=fopen("str.txt","rt");
FILE *fo=fopen("str.out","wt");
```

```
int main()
{
    char s[256],n;
    while (!feof(fi))
    {
        fgets(s,256,fi);
        fputs(tachtuphai(s),fo);
    }
    fprintf(fo,"\n");
```

42

```
fi=fopen("str.txt","rt");
while (!feof(fi))
{
    fgets(s,256,fi);
    int d=demsokytu(s)-1;
    if (feof(fi)) d++;
    fprintf(fo,"%d\n",d);
}
fprintf(fo,"\n");
```

43

```
fi=fopen("str.txt","rt");
while (!feof(fi))
{
    fgets(s,256,fi);
    demkytucuamoitu(s);
}
fclose(fi);
fclose(fo);
return 0;
}
```

44

```
char *tachtuphai(char *s)
{
    return strrchr(s, ' ') + 1;
}
```

45

```
int demsokytu(char *s)// không kể ký tự khoảng trắng
{
    int d=0;
    for (int i=0;i<strlen(s);i++)
        if (s[i]!=' ') d++;
    return d;
}
```

46

```

void demkytucuamoitu(char *s)
{
    fprintf(fo, "\n");
    int sokytumoitu=0;
    for (int i=0;i<strlen(s);i++)
        if (s[i]!=' ') sokytumoitu++;
        else
    {
        fprintf(fo, "%d ",sokytumoitu);
        sokytumoitu=0;
    }
    if (feof(fi)) sokytumoitu++;
    fprintf(fo,"%d",sokytumoitu-1);
}

```

47

## BT1.

Cho một chuỗi S bao gồm các chữ cái thường từ a đến z và ký tự khoảng trắng. Hãy viết các hàm thực hiện các công việc sau:

- Đếm xem trong chuỗi có bao nhiêu ký tự nguyên âm ? (a i o u e là các ký tự nguyên âm).
- Kiểm tra xem chuỗi đó có đối xứng hay không? Nếu có thì trả về 1, nếu không thì trả về 0. Ví dụ: abcba hay abccba là các chuỗi đối xứng.

48

## BT2.

Cho chuỗi s chứa các ký tự là chữ hoa, chữ thường và khoảng trắng; ta gọi một từ là dãy các ký tự liên tiếp nhau không chứa khoảng trắng. Giả thiết đầu và cuối chuỗi không có khoảng trắng và giữa các từ chỉ có duy nhất một khoảng trắng.

Hãy thực hiện các công việc sau:

- a.Đếm xem chuỗi s có bao nhiêu ký tự nguyên âm, bao nhiêu ký tự phụ âm ? (a i o e u là các nguyên âm).
- b.Đếm xem chuỗi s có bao nhiêu từ ?

49

## BT3.

Cho chuỗi s chứa các ký tự là chữ hoa, chữ thường và khoảng trắng; ta gọi một từ là dãy các ký tự liên tiếp nhau không chứa khoảng trắng. Giả thiết đầu và cuối chuỗi không có khoảng trắng và giữa các từ chỉ có duy nhất một khoảng trắng.

Viết các hàm thực hiện các công việc sau:

- a.Đếm xem chuỗi s có bao nhiêu từ ?
- b.Đếm số ký tự của mỗi từ trong chuỗi s.

50

## BT4.

Nhập vào một chuỗi  $s$  chứa các ký tự là chữ cái thường và ký tự khoảng trắng. Giả thiết đầu chuỗi  $s$ , cuối chuỗi  $s$  không có khoảng trắng và giữa các từ có đúng một khoảng trắng.

Hãy viết các hàm thực hiện các công việc sau:

- a.Đếm số từ có  $k$  ký tự trong chuỗi  $s$ .
- b.Tìm chuỗi con gồm  $p$  từ bên trái của chuỗi  $s$  (giả sử chuỗi có nhiều hơn  $p$  từ).

Ví dụ:  $s="ky thuat lap trinh"$  và  $k=5$ ,  $p=2$  thì kết quả câu a là 2 và kết quả câu b là chuỗi “ky thuat”.

51

## BT5.

Nhập vào một chuỗi  $s$  chứa các ký tự là chữ cái thường và ký tự khoảng trắng. Giả thiết đầu chuỗi  $s$ , cuối chuỗi  $s$  không có khoảng trắng và giữa các từ có đúng một khoảng trắng.

Hãy viết các hàm thực hiện các công việc sau:

- a.Đếm xem mỗi từ của chuỗi có bao nhiêu ký tự ?
- b.Tìm từ dài nhất của chuỗi  $s$  có bao nhiêu ký tự ?

Ví dụ:  $s="ky thuat lap trinh"$  thì kết quả câu a là 2 5 3 5 và kết quả câu b là 5.

52

## BT6.

Cho file str1.txt chứa 1 chuỗi s; giả sử s không có khoảng trắng đầu chuỗi, không có khoảng trắng cuối chuỗi, và giữa các từ có đúng 1 khoảng trắng.

Hỏi chuỗi s có bao nhiêu từ ? Từ dài nhất có bao nhiêu ký tự ?

str1.txt

ky thuat lap trinh

str1.out

4 5

53

## BT7.

Cho file str2.txt chứa nhiều chuỗi s; mỗi chuỗi trên một dòng; chiều dài mỗi chuỗi tối đa 256 ký tự. Giả sử các chuỗi không có khoảng trắng đầu chuỗi, không có khoảng trắng cuối chuỗi, và giữa các từ có đúng 1 khoảng trắng.

Hỏi chuỗi file str2.txt có bao nhiêu từ ? Từ dài nhất có bao nhiêu ký tự ?

str2.txt

ky thuat lap trinh

ngon ngu lap trinh python

str2.out

9 6

54

# DYNAMIC PROGRAMMING

## References:

- David Pisinger, **Algorithms for Knapsack problems**, Denmark, pp.1-200, 1995.
- Robert Sedgewick, Kevin Wayne, “**Algorithms**”, Fourth edition, Addison-Wesley, 2011.

1

## KHÁI NIỆM CƠ SỞ

- Quy hoạch động (QHĐ) là công cụ hiệu quả nhất để thiết kế các thuật toán giải bài toán tối ưu;
- Thuật ngữ “Quy hoạch” ý muốn nói đến quá trình đưa/chuyển bài toán ban đầu về một dạng nào đó để giải;
- Ví dụ: Tìm số hạng thứ n của dãy fibonacci

2

## KHÓ KHĂN KHI SỬ DỤNG QHĐ

- Không phải lúc nào sự kết hợp lời giải của các bài toán con cũng cho ra lời giải của bài toán con cỡ lớn hơn;
- Số lượng bài toán con cần giải quyết và lưu trữ đáp án có thể rất lớn, không thể chấp nhận được; và khi đó thuật toán QHĐ sẽ không hiệu quả.

3

## SƠ ĐỒ QUY HOẠCH ĐỘNG

Thuật toán QHĐ gồm ba giai đoạn sau:

- Phân rã;
- Ghi nhận lời giải;
- Tổng hợp lời giải;

4

## PHÂN RÃ

- Chia bài toán cần giải thành những bài toán con nhỏ hơn có cùng dạng với bài toán ban đầu sao cho bài toán con kích thước nhỏ nhất có thể giải một cách trực tiếp;
- Bản thân bài toán xuất phát có thể coi là bài toán con có kích thước lớn nhất trong họ các bài toán con này.

5

## GHI NHẬN LỜI GIẢI

- Lưu trữ lời giải của các bài toán con vào các bảng (thường sử dụng cấu trúc dữ liệu mảng);
- Lời giải (kết quả) của những bài toán con thường được sử dụng lại (rất) nhiều lần sau đó mà không cần giải lại chúng.

6

## TỔNG HỢP LỜI GIẢI

Lần lượt từ lời giải của các bài toán con kích thước nhỏ hơn tìm cách xây dựng lời giải của bài toán con kích thước lớn hơn, cho đến khi thu được lời giải của bài toán con xuất phát.

7

## TÍNH CHẤT CỦA BÀI TOÁN CÓ THỂ GIẢI BẰNG QHĐ

### Cấu trúc con tối ưu:

- Để giải được bài toán đặt ra một cách tối ưu, mỗi bài toán con cũng phải được giải một cách tối ưu.
- Mặc dù sự kiện này có vẻ hiển nhiên, nhưng nó thường không được thỏa mãn do lời giải của các bài toán con này có thể giao nhau.

### Số lượng các bài toán con không quá lớn:

- Tổng số các bài toán con cần giải phải bị chặn bởi một đa thức của kích thước dữ liệu vào.

8

## MỘT SỐ BÀI TOÁN QHĐ ĐIỀN HÌNH

+Toàn văn chương trình một số ví dụ điển hình

9

## DÃY CON TĂNG DÀI NHẤT

Cho dãy số nguyên  $A = a_1, a_2, \dots, a_n$  ( $n \leq 1000$ ,  $-10000 \leq a_i \leq 10000$ ).

Một dãy con của  $A$  là một cách chọn ra trong  $A$  một số phần tử giữ nguyên thứ tự.

**Yêu cầu:** Tìm dãy con không giảm của  $A$  có độ dài dài nhất (lưu ý: Có thể có nhiều dãy con kết quả; và ta chỉ yêu cầu tìm 01 dãy con thỏa mãn).

\*Đánh giá độ phức tạp thời gian tính của thuật toán.

10

## VÍ DỤ

Test 1:

INPUT:

10

2 8 11 3 5

9 10 4 17 6

OUTPUT:

2 3 5 9 10 17

Test2:

INPUT:

10

2 8 1 3 -3 5 -2 3 0 1

OUTPUT:

-3 -2 0 1

11

Hướng dẫn giải:

Với  $\forall i: 1 \leq i \leq n$ . Gọi  $L[i]$  là độ dài của dãy con tăng dài nhất kết thúc tại  $a_i$ .

Cơ sở quy hoạch động

$L[1]$  là độ dài dãy con tăng dài nhất kết thúc tại  $a_1$ . Dãy con này chỉ gồm một phần tử nên  $L[1] = 1$ .

12

## Công thức truy hồi

Giả sử với  $i$  từ 2 đến  $n$ , ta cần tính  $L[i]$ : độ dài dãy con tăng dài nhất kết thúc tại  $a_i$ .  $L[i]$  được tính trong điều kiện các  $L[i-1], L[i-2], \dots, L[1]$  đã biết.

Dãy con tăng dài nhất kết thúc tại  $a_i$  sẽ được thành lập bằng cách lấy  $a_i$  ghép vào cuối một trong số những dãy con tăng dài nhất kết thúc tại vị trí  $a_j$  đứng trước  $a_i$ . Ta sẽ chọn dãy nào để ghép  $a_i$  vào cuối? Tất nhiên là chỉ được ghép  $a_i$  vào cuối những dãy con kết thúc tại  $a_j$  nào đó nhỏ hơn  $a_i$  (để đảm bảo tính tăng) và dĩ nhiên ta sẽ chọn dãy dài nhất để ghép  $a_i$  vào cuối (để đảm bảo tính dài nhất). Và do đó  $L[i]$  được tính như sau: Xét tất cả các chỉ số  $j$  trong khoảng từ 1 đến  $i-1$  mà  $a_j < a_i$ , chọn ra chỉ số  $j_{max}$  có  $L[j_{max}]$  lớn nhất. Đặt  $L[i] = L[j_{max}] + 1$ .

13

## Truy vết

Tại bước xây dựng dãy  $L$ , mỗi khi tính  $L[i] = L[j_{max}] + 1$ , ta đặt  $P[i] = j_{max}$ . Để lưu lại rằng: Dãy con dài nhất kết thúc tại  $a_i$  sẽ có phần tử kế tiếp trước đó là  $a_{j_{max}}$ . Dãy con cần tìm sẽ kết thúc tại trong những vị trí mà giá trị  $L[i]$  là lớn nhất.

14

## Minh họa thuật toán

i	1	2	3	4	5	6	7	8	9	10
A[i]	2	8	11	3	5	9	10	4	17	6
L[i]	1	2	3	2	3	4	5	3	6	4
P[i]	0	1	2	1	4	5	6	4	7	8
KQ	2			3	5	9	10		17	

Notes

L[i]: Chiều dãy của dãy con tăng dài nhất tính đến phần tử A[i]

P[i]: Là vị trí trước phần tử ở vị trí i trong dãy con tăng dài nhất tính đến phần tử A[i]

KQ: Dãy kết quả

Các mảng L,P được tính như thế nào ?

15

```
// Day con tang dai nhat
#include<iostream.h>
#include<iomanip.h>
#define maxn 10000
void input(int a[], int &n);
void createtable(int a[], int n);
void result(int a[], int n);
int a[maxn],L[maxn],P[maxn],n;
int main()
{
    input(a,n);
    createtable(a,n);
    result(a,n);
    return 0;
}
```

16

```
void input(int a[], int &n)
{
    FILE *f;
    f=fopen("dayso.inp","rt");
    fscanf(f,"%d",&n);
    for(int i=1;i<=n;i++)
        fscanf(f,"%d",&a[i]);
    fclose(f);
}
```

17

```
void createtable(int a[], int n)
{
    for (int i=1;i<=n;i++)
    {
        int maxLj=0;
        int maxPi=0;
        for (int j=i-1;j>=1;j--)
            if (a[j]<=a[i]&& L[j]>maxLj)
            {
                maxLj=L[j];
                maxPi=j;
            }
        L[i]=maxLj+1;
        P[i]=maxPi;
    }
}
```

18

```
void result(int a[], int n)
{
    /*Xuat bang
    for (int i=1;i<=n;i++)
    cout<<setw(4)<<a[i];
    cout<<endl;
    for (int i=1;i<=n;i++)
    cout<<setw(4)<<L[i];
    cout<<endl;
    for (int i=1;i<=n;i++)
    cout<<setw(4)<<P[i];
    cout<<endl;
```

19

```
//tra bang tim ket qua
int maxL=0,pos;
for (int i=1;i<=n;i++)
if (L[i]>maxL)
{
    maxL=L[i];
    pos=i;
}
cout<<"Chieu dai: "<<maxL<<": "<<endl;;
int i=pos;
while (P[i]!=0)
{
    cout<<a[i]<<" ";
    i=P[i];
}
cout<<a[i]<<" ";
```

20

## BA LÔ 1

Cho  $n$  món hàng, món thứ  $i$  có khối lượng là  $a[i]$  ( $i=1..n$ ,  $a[i]$  là số nguyên) và một ba lô có khối lượng  $M$ .

**Yêu cầu:** Chọn những món hàng nào vào một ba lô sao cho tổng khối lượng của các món hàng được chọn là lớn nhất nhưng không vượt quá khối lượng ba lô ?  
(mỗi món hàng chỉ được chọn hoặc không được chọn).  
\*Đánh giá độ phức tạp thời gian tính của thuật toán.

21

## VÍ DỤ

INPUT:

5 12

6 1 8 7 1

OUTPUT

10

2 3 5

22

## Minh họa thuật toán

5 12

6 1 8 7 1

v k \	1	2	3	4	5	6	7	8	9	10	11	12
k	0	0	0	0	0	6	6	6	6	6	6	6
1	0	0	0	0	0	6	6	6	6	6	6	6
2	1	1	1	1	1	6	7	7	7	7	7	7
3	1	1	1	1	1	6	7	8	9	9	9	9
4	1	1	1	1	1	6	7	8	9	9	9	9
5	1	2	2	2	2	6	7	8	9	10	10	10

Mảng F[i,j] được tính như thế nào ?

23

## BA LÔ 1 (tạo bảng)

```
14 #include <iostream.h>
15 #include <iomanip.h>
16 #define maxn 100
17 #define maxM 10000
18 int f[maxn][maxM], a[maxn], n, M, result[maxn];
19 void input(int a[], int &n, int &M)
20 {
21     FILE *f;
22     f=fopen("Knapsack_01.inp","rt");
23     fscanf(f,"%d%d",&n,&M);
24     for(int i=1;i<=n;i++)
25         fscanf(f,"%d",&a[i]);
26     fclose(f);
27 }
28
```

24

```
29 int max(int a, int b)
30 {
31     return (a>b) ?a:b;
32 }
33
```

25

```
34 void createtable(int f[][]maxM, int a[], int n, int M)
35 {
36     for(int i=1;i<=M;i++)
37         if(i>=a[1])
38             f[1][i]=a[1];
39         else
40             f[1][i]=0;
41     for(int i=2;i<=n;i++)
42         for(int j=1;j<=M;j++)
43             if(j>=a[i])
44                 f[i][j]=max(f[i-1][j-a[i]]+a[i], f[i-1][j]);
45             else
46                 f[i][j]=f[i-1][j];
47 }
```

26

```

49 void reftable(int f[][][maxM], int a[], int n, int M)
50 {
51     int volume=0;
52     int K=n, V=M;
53     memset(result,0,sizeof(result));
54     do
55     {
56         while (f[K][V]==f[K-1][V]) K--;
57         result[K]=1;
58         V=f[K][V]-a[K];
59         volume+=a[K];
60     }
61     while (V>0);
62     cout<<"result: "<<volume<<" ";
63     for(int i=1;i<=n;i++)
64     if(result[i])
65         cout<<a[i]<<" ";
66 }
```

27

```

68 int main()
69 {
70     srand(time(NULL));
71     input(a,n,M);
72     createtable(f,a,n,M);
73     reftable(f,a,n,M);
74 }
```

28

## BA LÔ 2 (bài ba lô tổng quát)

Cho  $n$  món hàng, món thứ  $i$  có khối lượng  $a[i]$  và có giá trị là  $c[i]$  (trong đó  $a_i, c_i \leq 100$  và là các số nguyên) và một ba lô khối lượng  $M$ .

**Yêu cầu:** Chọn những món hàng nào để bỏ vào ba lô sao cho khối lượng các món hàng được chọn không vượt quá  $M$  ( $M$  là số nguyên dương  $\leq 10000$ ) và khi đó ba lô có giá trị lớn nhất.

(Lưu ý mỗi món hàng chỉ có thể chọn tối đa một lần).

\*Đánh giá độ phức tạp thời gian tính của thuật toán.

29

### VÍ DỤ

INPUT:

6 12

5 7

1 5

6 3

4 6

9 14

5 8

OUTPUT:

20

30

### Hướng dẫn giải:

Nếu gọi  $F[i][j]$  là giá trị lớn nhất có thể có bằng cách chọn trong các món hàng  $\{1, 2, \dots, i\}$  với giới hạn trọng lượng  $j$  thì giá trị lớn nhất khi được chọn trong số  $n$  món hàng với giới hạn trọng lượng  $M$  chính là  $F[n][M]$ .

31

### Công thức truy hồi tính $F[i][j]$

Với giới hạn trọng lượng  $j$ , việc chọn tối ưu trong số các món hàng  $\{1, 2, \dots, i - 1, i\}$  để có giá trị lớn nhất sẽ có hai khả năng:

- Nếu không chọn món hàng thứ  $i$  thì  $F[i][j]$  là giá trị lớn nhất có thể bằng cách chọn trong số các món hàng  $\{1, 2, \dots, i - 1\}$  với giới hạn trọng lượng là  $j$ . Tức là  $F[i][j] = F[i - 1][j]$ .
- Nếu có chọn món hàng thứ  $i$  (tất nhiên chỉ xét tới trường hợp này khi mà  $W_i \leq j$ ) thì  $F[i][j]$  bằng giá trị món hàng thứ  $i$  là  $V_i$  cộng với giá trị lớn nhất có thể có được bằng cách chọn trong số các món hàng  $\{1, 2, \dots, i - 1\}$  với giới hạn trọng lượng  $j - W_i$ . Tức là về mặt giá trị thu được:  $F[i][j] = C_i + F[i - 1][j - W_i]$ .

Vì theo cách xây dựng  $F[i][j]$  là giá trị lớn nhất có thể, nên  $F[i][j]$  sẽ là *max* trong hai giá trị thu được ở trên.

32

### Cơ sở quy hoạch động

Dễ thấy  $F[0][j] =$  giá trị lớn nhất có thể bằng cách chọn trong số 0 món hàng = 0.

### Tính bảng phương án

Bảng phương án  $F$  gồm  $n + 1$  dòng,  $M + 1$  cột, trước tiên được điền cơ sở quy hoạch động: Dòng 0 gồm toàn số 0. Sử dụng công thức truy hồi, dùng dòng 0 tính dòng 1, dùng dòng 1 tính dòng 2,... đến khi tính hết dòng  $n$ .

33

### Truy vết

Tính xong bảng phương án thì ta quan tâm đến  $F[n][M]$ , đó chính là giá trị lớn nhất thu được khi chọn trong cả n món hàng với giới hạn trọng lượng  $M$ . Nếu  $F[n][M] = F[n - 1][M]$  thì tức là không chọn món hàng thứ  $n$ , ta truy tiếp  $F[n - 1][M]$ . Còn nếu  $F[n][M] \neq F[n - 1][M]$  thì ta thông báo rằng phép chọn tối ưu có chọn món hàng thứ  $n$  và truy tiếp  $F[n - 1][M - Wn]$ . Cứ tiếp tục cho tới khi truy lên tới hàng 0 của bảng phương án.

34

INPUT:

6 12

5 7

1 5

6 3

4 6

9 14

5 8

OUTPUT:  
20

## Minh họa thuật toán

k \ v	1	2	3	4	5	6	7	8	9	10	11	12
k	1	0	0	0	7	7	7	7	7	7	7	7
v	1	5	5	5	5	7	12	12	12	12	12	12
1	0	0	0	0	7	7	7	7	7	7	7	7
2	5	5	5	5	5	7	12	12	12	12	12	12
3	5	5	5	5	5	7	12	12	12	12	12	15
4	5	5	5	5	6	11	12	12	12	13	18	18
5	5	5	5	6	11	12	12	12	14	19	19	19
6	5	5	5	6	11	13	13	13	14	19	20	20

Mảng  $F[i,j]$  được tính như thế nào ?

35

## TÍNH CHẤT

Thời gian giải quyết bài toán ba lô bằng thuật toán

QHĐ tỉ lệ với  $n * M$

$n$ : Số món hàng

$M$ : Khối lượng của ba lô

36

## BA LÔ 2 (tạo bảng)

```
1 #include<iostream.h>
2 #define maxn 1000
3 #define maxM 10000
4 int a[maxn], c[maxn], f[maxn][maxM];
5 int n, M;
6
7 void input()
8 {
9 FILE *fi;
10 fi=fopen("Knapsack_01.inp","rt");
11 fscanf(fi,"%d%d",&n,&M);
12 for (int i= 1;i<=n;i++)
13 fscanf(fi,"%d%d",&a[i],&c[i]);
14 fclose(fi);
15 }
```

37

```
17 int max(int a, int b)
18 {
19     return a>b?a:b;
20 }
21 void createtable() //ba lô 2
22 {
23 for (int i = 1;i<=maxn;i++)
24     f[0][i]=0;
25 for (int i = 1;i<=n;i++)
26 for (int j = 0;j<=M;j++)
27 {
28     if (j < a[i])
29         f[i][j] = f[i - 1][j];
30     else
31         f[i][j] = max(f[i - 1][j],f[i - 1][j - a[i]] + c[i]);
32     }
33 }
```

38

```

35 void reftable()
36 {
37     cout<<"Max Value : "<<f[n][M];
38 }
39
40 int main()
41 {
42     input();
43     createtable();
44     reftable();
45     return 0;
46 }

```

39

## DI CHUYỂN TRÊN BẢNG

Cho một bảng A kích thước  $m \times n$  ô (bắt đầu từ 1); mỗi ô chứa một số nguyên.

Từ ô  $A[i,j]$  chỉ có thể di chuyển sang một trong 3 ô  $A[i,j+1]$ ,  $A[i-1,j+1]$  và  $A[i+1,j+1]$ .

**Yêu cầu:** Tìm vị trí ô xuất phát từ cột 1 sáng cột  $n$  sao cho tổng các số ghi trên đường đi là lớn nhất (Yêu cầu xuất giá trị tổng này).

\*Đánh giá độ phức tạp thời gian tính của thuật toán.

40

## VÍ DỤ

MOVE.INP

4 5

1 3 4 2 10

5 8 2 3 8

7 5 9 4 6

2 9 4 5 7

MOVE.OUT

38

41

### Hướng dẫn giải:

Gọi  $B[i,j]$  là giá trị lớn nhất có thể có khi di chuyển đến ô  $A[i,j]$ . Rõ ràng với những ô ở cột 1 thì  $B[i,1]=A[i,1]$ .

Với những ô  $(i,j)$  ở các cột khác. Vì chỉ những ô  $(i,j-1)$ ,  $(i-1,j-1)$ ,  $(i+1,j-1)$  là có thể sang được ô  $(i,j)$ , và khi sang ô  $(i,j)$  thì giá trị được cộng thêm  $A(i,j)$  nữa. Chúng ta cần  $B[i,j]$  là số điểm lớn nhất có thể nên  $B[i,j]=\max(B[i,j-1], B[i-1,j-1], B[i+1,j-1])+A[i,j]$ . Ta dùng công thức truy hồi này tính tất cả các  $B[i,j]$ . Cuối cùng chọn ra  $B[i,n]$  là phần tử lớn nhất trên cột  $n$  của bảng  $B$  và từ đó truy vết tìm ra đường đi có giá trị lớn nhất.

42

## DI CHUYỂN TRÊN BÀNG (tạo bảng)

```
1 #include <iostream>
2 #include <fstream>
3 using namespace std;
4 const int maxn = 1000;
5 int m,n,re=0;
6 int a[maxn][maxn];
7 int b[maxn][maxn];
8
9 void readfile()
10 {
11     ifstream fi ("MOVE_02.INP");
12     fi>>m>>n;
13     for (int i=1;i<=m;i++)
14         for (int j=1;j<=n;j++)
15             fi>>a[i][j];
16     fi.close();
17 }
```

43

```
19 void createtable()
20 {
21     for (int i=1;i<=m;i++) b[i][1]=a[i][1];
22     for (int j=2;j<=n;j++)
23         for (int i=1;i<=m;i++)
24         {
25             int m;
26             m=max(b[i-1][j-1],b[i+1][j-1]);
27             m=max(m,b[i][j-1]);
28             b[i][j]=a[i][j]+m;
29         }
30     re = b[1][n];
31     for (int i=2;i<=m;i++)
32         re=max(re,b[i][n]);
33 }
```

44

```

35 void reftable()
36 {
37     ofstream fo ("MOVE.OUT");
38     fo<<re<<endl;
39     fo.close();
40 }
41 int main()
42 {
43     readfile();
44     createtable();
45     reftable();
46     return 0;
47 }

```

45

## TAM GIÁC SỐ

Cho một tam giác chứa các số nguyên không âm từ 0 đến 99. Dòng thứ  $h$  của tam giác chứa  $h$  số ( $1 < h \leq 1000$ ).

Một đường đi là hợp lệ nếu đường đi bắt đầu từ ô đỉnh của tam giác xuống đến một ô nào đó ở đáy của tam giác; và tại mỗi ô ở dòng  $i$  ta chỉ có thể đi tiếp đến ô ở dòng  $i+1$  theo cách: Hoặc sang bên trái hoặc sang bên phải.

**Yêu cầu:** Tìm một đường đi có tổng số các ô trên đường đi là lớn nhất; cho biết tổng số các ô trên đường đi tìm được.

\*Đánh giá độ phức tạp thời gian tính của thuật toán.

46

## VÍ DỤ

TRIANGLE.INP

5  
7  
3 8  
8 1 0  
2 7 4 4  
4 5 2 6 5

TRIANGLE.OUT

30

47

## TAM GIÁC SỐ (tạo bảng)

```
2 #include<iostream.h>
3 int max(int x, int y);
4 void readfile();
5 void findpath();
6 void result();
7 int f[1000][1000];
8 int a[1000][1000],n;
9 int main()
10 {
11     readfile();
12     findpath();
13     result();
14     return 0;
15 }
```

48

```

17 void readfile()
18 {
19     FILE *fi;
20     fi=fopen("triangle1.inp","rt");
21     fscanf(fi,"%d",&n);
22     for (int i=1;i<=n;i++)
23         for (int j=1;j<=i;j++)
24             fscanf(fi,"%d",&a[i][j]);
25     fclose(fi);
26 }
```

49

```

28 int max(int x, int y)
29 {
30     if (x>y) return x;
31     return y;
32 }
33 void findpath()
34 {
35     f[1][1]=a[1][1];
36     for (int i=2;i<=n;i++)
37         f[i][1]=f[i-1][1]+a[i][1];
38     for (int i=2;i<=n;i++)
39         for (int j=2;j<=i;j++)
40             f[i][j]=max(f[i-1][j-1],f[i-1][j])+a[i][j];
41 }
```

50

```

43 void result()
44 {
45     int m=INT_MIN;
46     for (int i=1;i<=n;i++)
47         if (f[n][i]>m)
48             m=f[n][i];
49     cout<<m<<endl;
50 }

```

51

## KẾT LUẬN

- Cho đến hiện tại, chưa xác định được một cách chính xác những bài toán nào có thể giải quyết hiệu quả bằng thuật toán QHĐ;
- QHĐ là một công cụ/chiến lược/phương pháp/thuật toán hiệu quả nhất để tìm lời giải đúng cho các bài toán tối ưu;

52

Học kỳ: I

Năm học: 2019-2020

Trình độ đào tạo: Đại học

Hình thức đào tạo: Chính quy

Họ tên sinh viên:

Mã số sinh viên:

*Sinh viên không sử dụng tài liệu – Cán bộ coi thi không giải thích đề.***CÂU 1 (3 điểm)**Dãy số  $x_n$  được định nghĩa như sau:

$$x_0 = 1; x_1 = 1;$$

$$x_n = nx_0 + (n-1)x_1 + (n-2)x_2 + (n-3)x_3 + \dots + x_{n-1}, \forall n > 1.$$

- a. Hãy cho biết giá trị  $x_7$ .  
 b. Viết hàm tính giá trị  $x_n$ .

**CÂU 2 (3 điểm)**

Nhập vào một chuỗi  $s$  chứa các ký tự là chữ cái thường và ký tự khoảng trắng. Giá thiết đầu chuỗi  $s$ , cuối chuỗi  $s$  không có khoảng trắng và giữa các từ có đúng một khoảng trắng.

Hãy viết các hàm thực hiện các công việc sau:

- a. Đếm số từ có  $k$  ký tự trong chuỗi  $s$ .  
 b. Tìm chuỗi con gồm  $p$  từ bên trái của chuỗi  $s$  (giả sử chuỗi có nhiều hơn  $p$  từ).

Ví dụ:  $s=$ ”ky thuat lap trinh” và  $k=5, p=2$  thì kết quả câu a là 2 và kết quả câu b là chuỗi “ky thuat”.

**CÂU 3 (3 điểm)**

Cho mảng hai chiều  $a$  có  $m$  dòng và  $n$  cột ( $m, n \leq 200$ ); các phần tử là các số nguyên dương.

Các dòng được đánh số từ 1 đến  $m$ , các cột được đánh số từ 1 đến  $n$ .

Hãy viết chương trình hoàn chỉnh thực hiện các công việc sau:

- a. Tìm tổng các phần tử trong một bảng con 3 dòng và 3 cột sao cho tổng các số trong bảng con đó là lớn nhất.  
 b. Tìm giá trị lớn nhất của các giá trị nhỏ nhất của từng dòng.

Dữ liệu vào được cho từ file văn bản table.inp; trong đó:

+ Dòng đầu ghi 2 số  $m, n$

+ Trong  $m$  dòng tiếp theo mỗi dòng ghi  $n$  số; mỗi số cách nhau ít nhất một khoảng trắng.

Kết quả ghi vào file văn bản table.out gồm 2 dòng; mỗi dòng ghi duy nhất một giá trị kết quả của mỗi câu tương ứng.

Ví dụ:

table.inp

4 5

3	4	2	2	1
5	10	101	3	10
4	20	7	101	13
101	31	10	101	31

**CÂU 4 (1 điểm)**

Cho  $n$  món hàng, và một ba lô chứa được khối lượng tối đa là  $w$ ; món hàng thứ  $i$  có khối lượng là  $a[i]$  ( $n$  là số nguyên dương;  $i=1..n$ ;  $a[i]$  là số nguyên dương). Trình bày thuật toán chọn những món hàng nào để bỏ vào một ba lô sao cho tổng khối lượng của các món hàng được chọn là lớn nhất mà không vượt quá  $w$  (giả thiết mỗi món hàng chỉ có thể chọn tối đa một lần).

Ví dụ:

$n=5, w=12$

6 1 8 7 1

Kết quả: Các món hàng được chọn có số thứ tự là 2,3,5 và khối lượng các món hàng được chọn là 10.

--- Hết ---

## ĐỀ KIỂM TRA THỰC HÀNH HỌC PHẦN KỸ THUẬT LẬP TRÌNH\_lần 1 (Đề 1: Nội dung: file & string)

Cho file STR.INP chứa nhiều chuỗi, mỗi chuỗi nằm hoàn toàn trên một dòng; mỗi chuỗi chứa các ký tự là chữ cái thường tiếng Anh hoặc khoảng trắng; giữa các từ có đúng một khoảng trắng, đầu và cuối mỗi chuỗi không có khoảng trắng (giả thiết mỗi chuỗi có ít nhất 2 từ).

Hãy lập trình thực hiện các công việc sau (tính cho tất cả các chuỗi có trong file)

**CÂU 1** (2.0 đ):

Hỏi file có bao nhiêu dòng ? Hỏi file có bao nhiêu chuỗi con “information technology” (lưu ý: mỗi dòng có thể xuất hiện nhiều lần chuỗi con này).

**CÂU 2** (2.0 đ):

Cho biết số lượng từ ? Từ dài nhất có bao nhiêu ký tự ?

**CÂU 3** (2.0 đ):

Hãy cho biết số lượng ký tự là nguyên âm ? Hãy cho biết có bao nhiêu từ có chứa ít nhất một ký tự nguyên âm ?

**CÂU 4** (2.0 đ):

Tìm từ đầu tiên của mỗi chuỗi; hãy cho biết chiều dài dài nhất của các từ này. Tương tự, Tìm từ thứ hai tính từ bên trái của mỗi chuỗi; hãy cho biết chiều dài dài nhất của các từ này.

**CÂU 5** (2.0 đ):

Từ có  $k$  ký tự được gọi là *tù loại k*. Hỏi từ loại nào xuất hiện nhiều lần nhất; xuất hiện bao nhiêu lần (chỉ cần tìm một kết quả thỏa yêu cầu).

Kết quả ghi vào file STR.OUT theo cấu trúc như minh họa ở ví dụ sau:

Ví dụ:

STR.INP

faculty of information technology

programming techniques

programming is an art

software technology

information technology has changed the world

STR.OUT (nếu Câu x không làm được ghi kết quả là CAU x:- -)

CAU 1: 5 2

CAU 2: 18 11

CAU 3: 43 18

CAU 4: 11 10

CAU 5: 11 4 // có thể là 10 4

Hết

**Ghi chú:**

-Sinh viên đặt tên file là <số hiệu bài kiểm tra>\_họ và tên (không dấu, không có khoảng trắng) và 3 chữ số cuối của mã số sinh viên; ví dụ de1b1\_phanthuha061.cpp và gửi file cpp qua email GV, đặt Subject là KTTLT\_baikiemtraso1

## ĐỀ KIỂM TRA THỰC HÀNH HỌC PHẦN KỸ THUẬT LẬP TRÌNH\_lần 1 (Đề 3: Nội dung: file & array)

Cho file TABLE.INP có cấu trúc như sau:

-Dòng đầu ghi 4 nguyên dương  $m,n,u,v$  ( $m,n \leq 200$ ).

-Trong  $m$  dòng tiếp theo, mỗi dòng ghi  $n$  số nguyên dương nhỏ hơn  $10^4$  (chỉ số dòng,cột tính từ 1). Xem mảng  $a$  ứng với dữ liệu của  $m$  dòng  $n$  cột này, các số cách nhau ít nhất một khoảng trắng. Dữ liệu vào là đúng không cần kiểm tra. Hãy lập trình thực hiện các công việc sau:

**CÂU 1** (2.0 đ).

Tìm số có tổng các chữ số của nó là nhỏ nhất; xuất tổng các chữ số của số tìm được; có bao nhiêu số thỏa tính chất của số tìm được ?

**CÂU 2** (2.0 đ).

Tìm số nguyên tố lớn nhất của mảng. Đếm xem trong mảng có bao nhiêu số nguyên tố mà mỗi chữ số của nó cũng là số nguyên tố ?

**CÂU 3** (2.0 đ).

Tìm số nhỏ nhất trong số các số lớn nhất trên từng cột của mảng; có bao nhiêu số bằng số tìm được.

**CÂU 4** (2.0 đ).

Tìm một bảng vuông con có kích thước lớn nhất chỉ chứa toàn số nguyên tố; xuất kích thước hình vuông tìm được; có bao nhiêu hình vuông có cùng kích thước này thỏa tính chất của bài toán.

**CÂU 5** (2.0 đ).

Nếu đặt các số của mảng tăng dần từ trái qua phải và từ trên xuống dưới thì phần tử tại vị trí dòng  $u$  cột  $v$  có giá trị là bao nhiêu ? ( $u, v$  bắt đầu tính từ 1). Nếu đặt các số của mảng giảm dần từ trái qua phải và từ trên xuống dưới thì phần tử tại vị trí dòng  $u$  cột  $v$  có giá trị là bao nhiêu ? ( $u, v$  bắt đầu tính từ 1).

Kết quả ghi vào file TABLE.OUT có cấu trúc như minh họa ở ví dụ sau:

Ví dụ:

TABLE.INP

4	5	3	4	
101	12	21	42	49
5	101	13	22	10
101	2	3	101	4
22	101	23	44	23

TABLE.OUT (nếu Câu x nào không làm được thì ghi kết quả là CAU x:- -)

CAU 1: 1 1

CAU 2: 101 5

CAU 3: 23 2

CAU 4: 2 3

CAU 5: 44 13

Hết

**Ghi chú:**

-Sinh viên đặt tên file là <số hiệu đẻ số hiệu bài kiểm tra>\_họ và tên (không dấu, không có khoảng trắng) và 3 chữ số cuối của mã số sinh viên; ví dụ de3b1\_phanthuha061.cpp và gửi file cpp qua email GV, đặt Subject là KTLT\_baikiemtraso1

## KIỂM TRA HỌC PHẦN KỸ THUẬT LẬP TRÌNH\_lần 2 (Đề 1:Quy hoạch động)

Viết chương trình hoàn chỉnh giải quyết các bài toán sau (input/output từ file text).

### BÀI 1. KNAPSACK (3 điểm)

Cho  $n$  món hàng và một ba lô chứa được khối lượng tối đa là  $M$ , món hàng thứ  $i$  có khối lượng  $a[i]$  và có giá trị là  $c[i]$ . Hãy lập trình chọn một số món hàng đặt vào ba lô sao các món hàng được chọn có tổng khối lượng không vượt quá  $M$  và có tổng giá trị là lớn nhất (giả thiết  $n$  là số nguyên dương  $\leq 100$ ;  $a_i, c_i$  là các số nguyên dương  $\leq 100$ ;  $M$  là số nguyên dương  $\leq 10000$ ; mỗi món hàng chỉ có thể chọn tối đa một lần).

Xuất ra tổng giá trị là lớn nhất đó.

Cấu trúc file dữ liệu vào/ra được mô tả như ví dụ sau:

Ví dụ:

KNAPSACK.INP

6 12  
5 7  
1 5  
6 3  
4 6  
9 14  
5 8

KNAPSACK.OUT

20

### BÀI 2. MOVE (3 điểm)

Cho một bảng  $A$  kích thước  $m \times n$  ô (bắt đầu từ 1;  $1 \leq m, n \leq 100$ ); mỗi ô chứa một số nguyên.

Tù ô  $A[i,j]$  chỉ có thể di chuyển sang một trong 3 ô  $A[i,j+1]$ ,  $A[i-1,j+1]$  và  $A[i+1,j+1]$ .

**Yêu cầu:** Tìm vị trí ô xuất phát từ cột 1 sáng cột  $n$  sao cho tổng các số ghi trên đường đi là lớn nhất (Yêu cầu xuất giá trị tổng này).

Cấu trúc file dữ liệu vào/ra được mô tả như ví dụ sau:

MOVE.INP

4 5  
1 3 4 2 10  
5 8 2 3 8  
7 5 9 4 6  
2 9 4 5 7

MOVE.OUT

38

### BÀI 3. TRIANGLE (4 điểm)

Cho một tam giác chứa các số nguyên không âm từ 0 đến 99. Dòng thứ  $h$  của tam giác chứa  $h$  số (số dòng của tam giác  $>1$  và  $\leq 1000$ ). Một đường đi là hợp lệ nếu đường đi bắt đầu từ ô đỉnh của tam giác xuống đến một ô nào đó ở đáy của tam giác; và tại mỗi ô ở dòng  $i$  ta chỉ có thể đi tiếp đến ô ở dòng  $i+1$  theo cách: Hoặc sang bên trái hoặc sang bên phải. Hãy lập trình tính tổng lớn nhất các số trên các đường đi hợp lệ.

7  
3 8 //ví dụ nếu đang ở ô ứng với số 3 thì ta chỉ có thể đến với một trong hai ô  
8 1 0 // ứng với số 8 hoặc số 1  
2 7 4 4  
4 5 2 6 5

Dữ liệu vào được cho trong file triangle.inp như sau:

-Dòng đầu ghi số n;

-Trong  $n$  dòng tiếp theo; dòng thứ  $h$  ghi  $h$  số. Các số cách nhau ít nhất một khoảng trắng.  
Kết quả được ghi vào file triangle.out là một số nguyên duy nhất ghi kết quả của bài toán.  
Cấu trúc file dữ liệu vào/ra được mô tả như ví dụ sau:

**Ví dụ:**

TRIANGLE.INP

5 // ghi chú: là số n  
7  
3 8  
8 1 0  
2 7 4 4  
4 5 2 6 5

TRIANGLE.OUT

30

**Hết**

**Ghi chú:**

-Sinh viên đặt tên các file bài làm là bai1.cpp, bai2.cpp, bai3.cpp; đặt các file này vào thư mục <số hiệu đẻ số hiệu bài kiểm tra>\_họ và tên (không dấu, không có khoảng trắng) và 3 chữ số cuối của mã số sinh viên; ví dụ de1b2\_phanthuha061, nên thư mục này và gửi file nén qua email GV, đặt Subject là KTLT\_baikiemtraso2