BÀI 26 PHƯƠNG PHÁP LÀM MỊN DẦN TRONG THIẾT KẾ CHƯƠNG TRÌNH

SAU BÀI HỌC NÀY EM SẼ:

- Biết và giải thích được phương pháp làm mịn dần trong lập trình.
- Vận dụng được phương pháp làm mịn dần để thiết kế chương trình.

Em đã biết thiết kế một số thuật toán và chương trình: tìm kiếm tuần tự, tìm kiếm nhị phân, sắp xếp chèn, sắp xếp chọn, sắp xếp nổi bọt. Tất cả các thiết kế chương trình đó có điểm nào chung?

Theo em, để thiết kế một thuật toán đúng giải một bài toán cho trước cần trải qua các bước như thế nào? Nêu quan điểm của riêng em và trao đổi với các bạn.

1. PHƯƠNG PHÁP THIẾT KẾ LÀM MỊN DẦN

Hoạt động 1 **Tìm hiểu các bước thiết kế làm mịn dần**

Cùng trao đổi, thảo luận các bước thiết kế chương trình theo thuật toán sắp xếp chèn, từ đó đưa ra phương pháp chính khi thiết kế chương trình. Sau mỗi bước thiết kế cần trao đổi và trả lời các câu hỏi sau:

1. Bước này đã thực hiện được công việc gì?
2. Kết quả vừa thực hiện với kết quả của bước trước đó khác nhau như thế nào?

Bài toán gốc. Cho trước dãy số A: A[0], A[1], ..., A[n-1]. Cần tiến hành sắp xếp dãy trên theo thứ tự tăng dần. Kết quả phải nhận được:

`A[0] ≤ A[1] ≤ ... ≤ A[n-1]`

Ví dụ với bộ dữ liệu đầu vào là dãy [2,1,7,10,4] thì kết quả thu được dãy [1,2,4,7,10].

Quá trình phân tích, thiết kế được mô tả theo các bước sau.

a) Tìm hiểu bài toán

Bài toán gốc là cho trước dãy A, cần sắp xếp lại dãy này theo thứ tự tăng dần.

b) Thiết kế chương trình giải bài toán

Việc thiết kế chương trình giải bài toán được chia thành nhiều bước, trong đó các hành động ở bước sau là sự cụ thể hoá hơn ý tưởng, thao tác được nêu trong bước trước.

Bước 1. Thiết lập ý tưởng thiết kế ban đầu.

Ý tưởng ban đầu của thuật toán đơn giản như sau: Cần duyệt một lượt từ phần tử thứ hai đến phần tử cuối của dãy sao cho khi kết thúc thì dãy cũng được sắp xếp xong. Như vậy phần chính của thuật toán là một vòng lặp với biến i chạy từ chỉ số 1 đến n – 1. Với mỗi giá trị i, cần thực hiện một số thao tác để bổ sung A[i] vào dãy các phần tử đã được sắp xếp A[0], A[1]..., A[i-1] sao cho dãy mới thu được từ A[0] đến A[i] được sắp xếp đúng.

Như vậy, thuật toán ban đầu có thể được mô tả như sau:

```
1 for i in range(1,n):
2     <Đặt A[i] vào đúng vị trí của dãy A[0],A[1],...,A[i-1]>
```

Tại dòng 2 của sơ đồ trên, bài toán được đặt ra là: "Chèn phần tử A[i] vào đúng vị trí của dãy A[0], A[1], ..., A[i-1]".

Bước 2. Làm chi tiết hơn, thực hiện việc "Chèn A[i] vào đúng vị trí."

Vì các phần tử bên trái của A[i] là A[0], A[1], ..., A[i-1] đã được sắp xếp đúng nên thao tác "chèn" phần tử A[i] sẽ được thực hiện như sau:

```
<Lấy phần tử A[i] ra và lần lượt chuyển các phần tử bên trái A[i] nhưng có giá trị lớn hơn A[i] sang phải. Sau đó đặt A[i] vào vị trí trống>
```

Theo mô tả trên, việc "Chèn A[i] vào đúng vị trí" có thể được thực hiện như sau:

Chèn A[i] vào đúng vị trí

```
1 Nhấc phần tử A[i] lên.
2 Chuyển các phần tử bên trái A[i] và lớn hơn A[i] sang phải.
3 Chèn A[i] vào vị trí trống.
```

Các bước tiếp theo sẽ làm mịn hơn, chi tiết hơn các thao tác trên.

Bước 3. Nhấc A[i] lên.

Thao tác này sẽ được thực hiện đơn giản bằng việc tạo ra một biến mới **value** để lưu trữ giá trị A[i].

```
value = A[i]
```

Bước 4. Chuyển các phần tử bên trái A[i] và lớn hơn A[i] sang phải.

Thao tác này có thể được thực hiện như sau: Thiết lập biến j = i – 1 là chỉ số của phần tử ngay bên trái A[i]. Sau đó liên tục so sánh A[j] với value. Nếu A[j] > value thì chuyển A[j] sang phải một vị trí bằng lệnh A[j+1] = A[j] và giảm j = j – 1. Quá trình sẽ kết thúc khi đi hết bên trái của dãy hoặc A[j] <= value. Tất cả các công việc này được thể hiện bằng đoạn chương trình sau:

```
1 j = i-1
2 while j >= 0 and A[j] > value:
3     A[j+1] = A[j]
4     j = j - 1
```


Bước 5. Chèn A[i] vào đúng vị trí trống.

Từ bước 4 chúng ta đã biết quá trình chuyển sang phải của các phần tử A[j] sẽ kết thúc khi A[j] <= A[i], do đó vị trí j+1 chính là vị trí trống cần chèn. Việc chèn phần tử A[i] (giá trị được lưu trong value) vào vị trí j + 1 được thực hiện bằng câu lệnh:

```
A[j+1] = value
```

Như vậy ba thao tác đã nêu ở bước 2 trên có thể được thực hiện bằng các câu lệnh chương trình như sau:

Chèn A[i] vào đúng vị trí

```
1 value = A[i]
2 j = i-1
3 while j >= 0 and A[j] > value:
4     A[j+1] = A[j]
5     j = j - 1
6 A[j+1] = value
```

Tới đây quá trình thiết kế kết thúc vì chúng ta đã chi tiết hoá bằng các câu lệnh tất cả các thao tác được mô tả trong các bước trên.

c) Chương trình hoàn chỉnh

Chương trình giải bài toán đặt ra được thiết kế hoàn chỉnh dưới dạng hàm InsertionSort(A). Tổng hợp các bước trên chúng ta có chương trình hoàn chỉnh như sau.

```
1 def InsertionSort(A):
2     n = len(A)
3     for i in range(1,n):
4         value = A[i]
5         j = i-1
6         while j >= 0 and A[j] > value:
7            A[j+1] = A[j]
8            j = j -1
9         A[j+1] = value
```

Như vậy quá trình thiết kế chương trình theo thuật toán sắp xếp chèn đã trải qua một số bước, mỗi bước sẽ thực hiện chi tiết hoá hay còn gọi là làm mịn dần các phân tích của bước trước đó.

> Phương pháp làm mịn dần trong thiết kế chương trình là quá trình chi tiết hóa từ ý tưởng của các bước trước thành những hành động cụ thể hơn ở các bước sau. Ở bước cuối cùng, các hành động tương ứng với các câu lệnh của ngôn ngữ lập trình để viết chương trình hoàn chỉnh.

1. Trong các bước đã thực hiện của bài toán sắp xếp chèn ở trên, bước nào là đơn giản nhất theo nghĩa có thể thực hiện ngay bằng các lệnh lập trình.
2. Nếu bài toán đặt ra là sắp xếp dãy A theo thứ tự giảm dần thì các bước thiết kế như trên có cần thay đổi không? Thay đổi như thế nào?

2. THIẾT KẾ CHƯƠNG TRÌNH BẰNG PHƯƠNG PHÁP LÀM MỊN DẦN

Hoạt động 2 **Thiết kế chương trình bằng phương pháp làm mịn dần**

Thực hiện thiết kế thuật toán và chương trình bằng phương pháp làm mịn dần theo các bài toán sau. Trao đổi, thảo luận với bạn bè để thiết lập được lời giải tốt hơn.

Bài toán. Cho trước dãy số A: A[0], A[1], ..., A[n-1]. Cặp phần tử A[i], A[j] được gọi là nghịch đảo nếu i < j nhưng A[i] > A[j]. Cần viết chương trình đếm số các cặp nghịch đảo của dãy A. Ví dụ dãy 3, 4, 2, 1 sẽ có 5 cặp nghịch đảo là (3,2), (3,1), (4,2), (4,1), (2,1).

Thiết kế theo phương pháp làm mịn dần

a) Tìm hiểu bài toán

Bài toán gốc là cho trước dãy số A có n phần tử, cần đếm số các cặp phần tử nghịch đảo của A.

b) Thiết kế chương trình giải bài toán

Chúng ta sẽ thiết kế lời giải bài toán theo phương pháp làm mịn dần.

Bước 1. Thiết lập ý tưởng thiết kế ban đầu.

Bài toán có yêu cầu chính là đếm tất cả các cặp chỉ số nghịch đảo của dãy A. Vậy phần khung chính của chương trình sẽ là:

```
<Đếm số lượng các cặp số nghịch đảo (A[i],A[j]) của dãy A, trả về giá trị này>
```

Như vậy để thực hiện được yêu cầu trên chúng ta cần thực hiện 2 công việc: cần tìm tất cả các cặp chỉ số (i, j) có thể tạo cặp nghịch đảo A[i], A[j], sau đó kiểm tra xem cặp này có là nghịch đảo không, nếu có thì tăng biến đếm lên 1 đơn vị.

Lược đồ thuật toán ban đầu có thể được mô tả như sau:

```
count = 0
Tìm tất cả các cặp chỉ số (i,j) có thể tạo ra cặp phần tử nghịch đảo
Kiểm tra nếu cặp A[i], A[j] là nghịch đảo thì tăng count lên 1 đơn vị.
return count
```

Bước 2. Tìm tất cả các cặp chỉ số (i,j)

Cách tìm tự nhiên tất cả các cặp nghịch đảo là cần duyệt trên tất cả các bộ (i, j) trong đó i, j chạy từ 0 đến n – 1. Như vậy có thể thiết lập 2 vòng lặp theo i, j để tìm. Chú ý để tiết kiệm thời gian chúng ta sẽ chỉ tìm các chỉ số i chạy từ 0 đến n-2, chỉ số j tính từ i+1 đến n – 1. Kết quả bước làm mịn này là đoạn chương trình sau:

```
for i in range(n-1):
    for j in range(i+1,n):
```

Như vậy tới bước này, thuật toán gốc có thể được mô tả như sau:

```
count = 0
for i in range(n-1):
    for j in range(i+1,n):
        if <cặp (i,j) là nghịch đảo>:
            tăng count lên 1 đơn vị
return count
```

Bước 3. Kiểm tra tính nghịch đảo của cặp (i,j).

Chúng ta đã biết cặp (i, j) sẽ là nghịch đảo khi và chỉ khi i < j and A[i] > A[j]. Tuy nhiên tại bước 2 chúng ta đã thiết lập được tất cả các cặp (i, j) với điều kiện i < j do đó việc kiểm tra nghịch đảo chỉ còn một điều kiện là A[i] > A[j]. Kết quả làm mịn của bước 3 như sau:

```
if A[i] > A[j]:
    count = count + 1
```

Tới bước này các thao tác chi tiết cần thực hiện để giải bài toán đã gần hoàn thành như sau:

```
count = 0
for i in range(n-1):
    for j in range(1,n):
        if A[i] > A[j]:
            count = count + 1
return count
```

c) Chương trình hoàn chỉnh

Trên cơ sở các phân tích trên chúng ta có thể thiết lập hàm Nghichdao(A) để đếm số các cặp nghịch đảo của dãy A cụ thể như sau:

```
1  def Nghichdao(A):
2      n = len(A)
3      count = 0
4      for i in range(n-1):
5          for j in range(i+1,n):
6              if A[i] > A[j]:
7                  count = count + 1
8      return count
```

Phương pháp làm mịn dần trong thiết kế chương trình phải tuân thủ các quy trình và nguyên tắc sau:

- Chia việc thiết kế thành từng bước và thực hiện lần lượt các bước.
- Mỗi bước lớn có thể được chia thành nhiều bước nhỏ hơn để giải quyết độc lập.
- Tiếp cận bài toán từ tổng quan đến chi tiết, mỗi bước tiếp theo sẽ phải là thiết kế chi tiết hơn bước trước đó. Quá trình như vậy sẽ tiếp tục cho đến khi viết xong toàn bộ các câu lệnh của chương trình giải bài toán đã cho.

1. Với Bài toán 1, có thể tách các dòng lệnh từ 4 đến 9 thành một hàm con độc lập được không?

2. Trong thiết kế bài toán tìm các cặp phần tử nghịch đảo, các bước sau đã thực hiện những thay đổi quan trọng nào so với bước trước đó?

LUYỆN TẬP

1. Phát biểu sau đúng hay sai?

Khi thiết kế chương trình thì việc đầu tiên là tìm hiểu yêu cầu chung của bài toán, xác định đầu vào, đầu ra của bài toán, sau đó mới đi cụ thể vào chi tiết.

2. Sử dụng thiết kế của Bài toán 2, tìm tất cả các cặp nghịch đảo của dãy: 3, 2, 1, 5, 4.

VẬN DỤNG

1. Sử dụng phương pháp làm mịn dần để giải bài toán sau: Cho trước số tự nhiên không âm n, viết chương trình kiểm tra xem số n có phải là số nguyên tố hay không? Chương trình cần thông báo "CÓ" nếu n là số nguyên tố, ngược lại thông báo "KHÔNG".

2. Với thuật toán sắp xếp chèn, chứng minh rằng nếu thay toàn bộ phần

`<Chèn A[i] vào vị trí đúng của dãy con A[0], A[1], ..., A[i – 1]>`

bằng các lệnh sau thì chương trình vẫn đúng:

```
1  j = i
2  while j > 0 and A[j] < A[j-1]:
3      Đổi chỗ A[j] và A[j-1]
4      j = j – 1
```