TỔNG QUAN

I. MỤC ĐÍCH YÊU CẦU

Môn Lập Trình Căn Bản A cung cấp cho sinh viên những kiến thức cơ bản về lập trình thông qua ngôn ngữ lập trình C. Môn học này là nền tảng để tiếp thu hầu hết các môn học khác trong chương trình đào tạo. Mặt khác, nắm vững ngôn ngữ C là cơ sở để phát triển các ứng dụng.

Học xong môn này, sinh viên phải nắm được các vấn đề sau:

- Khái niệm về ngôn ngữ lập trình.
- Khái niệm về kiểu dữ liệu
- Kiểu dữ liệu có cấu trúc (cấu trúc dữ liệu).
- Khái niệm về giải thuật
- Ngôn ngữ biểu diễn giải thuật.
- Ngôn ngữ sơ đồ (lưu đồ), sử dụng lưu đồ để biểu diễn các giải thuật.
- Tổng quan về Ngôn ngữ lập trình C.
- Các kiểu dữ liệu trong C.
- Các lệnh có cấu trúc.
- Cách thiết kế và sử dụng các hàm trong C.
- Một số cấu trúc dữ liệu trong C.

II. ĐỐI TƯỢNG MÔN HỌC

Môn học lập trình căn bản được dùng để giảng dạy cho các sinh viên sau:

- Sinh viên năm thứ 2 chuyên ngành Tin học, Toán Tin, Lý Tin.
- Sinh viên năm thứ 2 chuyên ngành Điện tử (Viễn thông, Tự động hóa…)

III. NỘI DUNG CỐT LÕI

Trong khuôn khổ 45 tiết, giáo trình được cấu trúc thành 2 phần: Phần 1 giới thiệu về lập trình cấu trúc, các khái niệm về lập trình, giải thuật… Phần 2 trình bày có hệ thống về ngôn ngữ lập trình C, các câu lệnh, các kiểu dữ liệu…

PHẦN 1: Giới thiệu cấu trúc dữ liệu và giải thuật

PHẦN 2: Giới thiệu về một ngôn ngữ lập trình - Ngôn ngữ lập trình C

Chương 1: Giới thiệu về ngôn ngữ C & môi trường lập trình Turbo C

Chương 2: Các thành phần của ngôn ngữ C

Chương 3: Các kiểu dữ liệu sơ cấp chuẩn và các lệnh đơn

Chương 4: Các lệnh có cấu trúc

Chương 5: Chương trình con

Chương 6: Kiểu mảng

Chương 7: Kiểu con trỏ

Chương 8: Kiểu chuỗi ký tự

Chương 9: Kiểu cấu trúc

IV. KIẾN THỨC LIÊN QUAN

Để học tốt môn Lập Trình Căn Bản A, sinh viên cần phải có các kiến thức nền tảng sau:

- Kiến thức toán học.
- Kiến thức và kỹ năng thao tác trên máy tính.

V. DANH MỤC TÀI LIỆU THAM KHẢO

[1] Nguyễn Văn Linh, *Giáo trình Tin Học Đại Cương A*, Khoa Công Nghệ Thông Tin, Đại học Cần Thơ, 1991.

[2] Nguyễn Đình Tê, Hoàng Đức Hải , *Giáo trình lý thuyết và bài tập ngôn ngữ C*; Nhà xuất bản Giáo dục, 1999.

[3] Nguyễn Cẩn, *C – Tham khảo toàn diện*, Nhà xuất bản Đồng Nai, 1996.

[4] Võ Văn Viện, *Giúp tự học Lập Trình với ngôn ngữ C*, Nhà xuất bản Đồng Nai, 2002.

[5] Brain W. Kernighan & Dennis Ritchie, *The C Programming Language*, Prentice Hall Publisher, 1988.

VI. TỪ KHÓA

Bài toán, chương trình, giải thuật, ngôn ngữ giả, lưu đồ, biểu thức, gán, rẽ nhánh, lặp, hàm, mảng, con trỏ, cấu trúc, tập tin.

Phần 1: GIỚI THIỆU VỀ CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

Học xong chương này, sinh viên sẽ nắm bắt được các vấn đề sau:

- Khái niệm về ngôn ngữ lập trình.
- Khái niệm về kiểu dữ liệu
- Kiểu dữ liệu có cấu trúc (cấu trúc dữ liệu).
- Khái niệm về giải thuật
- Ngôn ngữ biểu diễn giải thuật.
- Ngôn ngữ sơ đồ (lưu đồ), sử dụng lưu đồ để biểu diễn các giải thuật.

Trọng tâm của phần này là giải thuật & cách biểu diễn giải thuật. Chính nhờ điều này ta mới có thể giải quyết các yêu cầu bằng chương trình máy tính.

I. TỪ BÀI TOÁN ĐẾN CHƯƠNG TRÌNH

Giả sử chúng ta cần viết một chương trình để giải phương trình bậc 2 có dạng $ax^2 + bx + c = 0$ hay viết chương trình để lấy căn bậc n của một số thực m ($\sqrt[n]{m}$). Công việc đầu tiên là chúng ta phải hiểu và biết cách giải bài toán bằng lời giải thông thường của người làm toán. Để giải được bài toán trên bằng máy tính (lập trình cho máy tính giải) thì chúng ta cần phải thực hiện qua các bước như:

- Mô tả các bước giải bài toán.
- Vẽ sơ đồ xử lý dựa trên các bước.
- Dựa trên sơ đồ xử lý để viết chương trình xử lý bằng ngôn ngữ giả (ngôn ngữ bình thường của chúng ta).
- Chọn ngôn ngữ lập trình và chuyển chương trình từ ngôn ngữ giả sang ngôn ngữ lập trình để tạo thành một chương trình hoàn chỉnh.
- Thực hiện chương trình: nhập vào các tham số, nhận kết quả.

Trong nhiều trường hợp, từ bài toán thực tế chúng ta phải xây dựng mô hình toán rồi mới xác định được các bước để giải. Vấn đề này sẽ được trình bày chi tiết trong môn Cấu Trúc Dữ Liệu.

II. GIẢI THUẬT

II.1. Khái niệm giải thuật

Giải thuật là một hệ thống chặt chẽ và rõ ràng các quy tắc nhằm xác định một dãy các thao tác trên những dữ liệu vào sao cho sau một số hữu hạn bước thực hiện các thao tác đó ta thu được kết quả của bài toán.

Ví dụ 1: Giả sử có hai bình A và B đựng hai loại chất lỏng khác nhau, chẳng hạn bình A đựng rượu, bình B đựng nước mắm. Giải thuật để hoán đổi (swap) chất lỏng đựng trong hai bình đó là:

- Yêu cầu phải có thêm một bình thứ ba gọi là bình C.
- Bước 1: Đổ rượu từ bình A sang bình C.
- Bước 2: Đổ nước mắm từ bình B sang bình A.
- Bước 3: Đổ rượu từ bình C sang bình B.

Ví dụ 2: Một trong những giải thuật tìm ước chung lớn nhất của hai số a và b là:

- Bước 1: Nhập vào hai số a và b.
- Bước 2: So sánh 2 số a,b chọn số nhỏ nhất gán cho UCLN.
- Bước 3: Nếu một trong hai số a hoặc b không chia hết cho UCLN thì thực hiện bước 4, ngược lại (cả a và b đều chia hết cho UCLN) thì thực hiện bước 5.
- Bước 4: Giảm UCLN một đơn vị và quay lại bước 3
- Bước 5: In UCLN - Kết thúc.

II.2 Các đặc trưng của giải thuật

- Tính kết thúc: Giải thuật phải dừng sau một số hữu hạn bước.
- Tính xác định: Các thao tác máy tính phải thực hiện được và các máy tính khác nhau thực hiện cùng một bước của cùng một giải thuật phải cho cùng một kết quả.
- Tính phổ dụng: Giải thuật phải "vét' hết các trường hợp và áp dụng cho một loạt bài toán cùng loại.
- Tính hiệu quả: Một giải thuật được đánh giá là tốt nếu nó đạt hai tiêu chuẩn sau:

  - Thực hiện nhanh, tốn ít thời gian.

  - Tiêu phí ít tài nguyên của máy, chẳng hạn tốn ít bộ nhớ.

Giải thuật tìm UCLN nêu trên đạt tính kết thúc bởi vì qua mỗi lần thực hiện bước 4 thì UCLN sẽ giảm đi một đơn vị cho nên trong trường hợp xấu nhất thì UCLN=1, giải thuật phải dừng. Các thao tác trình bày trong các bước, máy tính đều có thể thực hiện được nên nó có tính xác định. Giải thuật này cũng đạt tính phổ dụng vì nó được dùng để tìm UCLN cho hai số nguyeên dương a và b bất kỳ. Tuy nhiên tính hiệu quả của giải thuật có thể chưa cao; cụ thể là thời gian chạy máy có thể còn tốn nhiều hơn một số giải thuật khác mà chúng ta sẽ có dịp trở lại trong phần lập trình C.

II.3 Ngôn ngữ biểu diễn giải thuật

Để biểu diễn giải thuật, cần phải có một tập hợp các ký hiệu dùng để biểu diễn, mỗi ký hiệu biểu diễn cho một hành động nào đó. Tập hợp các ký hiệu đó lại tạo thành ngôn ngữ biểu diễn giải thuật.

II.3.1 Ngôn ngữ tự nhiên

Ngôn ngữ tự nhiên là ngôn ngữ của chúng ta đang sử dụng, chúng ta có thể sử dụng ngôn ngữ tự nhiên để mô tả giải thuật giống như các ví dụ ở trên.

Ví dụ: Ta có giải thuật giải phương trình bậc nhất dạng $ax + b = 0$ như sau:

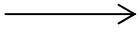
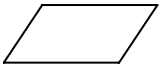
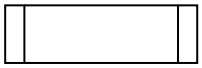
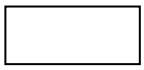
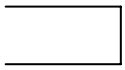
- Bước 1: Nhận giá trị của các tham số a, b
- Bước 2: Xét giá trị của a xem có bằng 0 hay không? Nếu a=0 thì làm bước 3, nếu a khác không thì làm bước 4.

▪ Bước 3: (a bằng 0) Nếu b bằng 0 thì ta kết luận phương trình vô số nghiệm, nếu b khác 0 thì ta kết luận phương trình vô nghiệm.

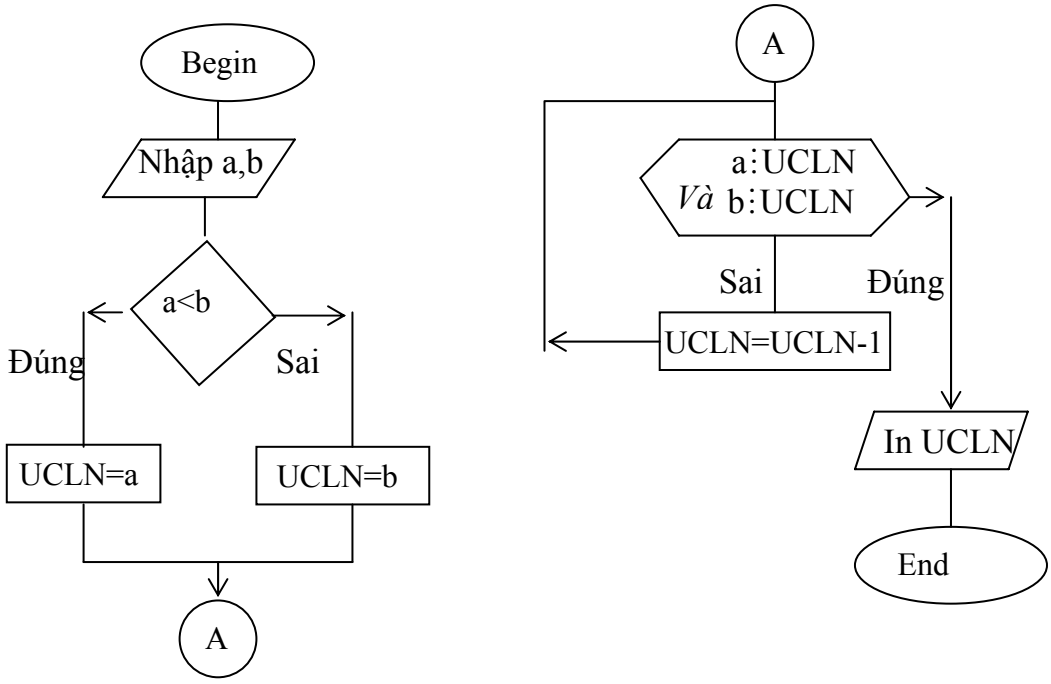
▪ Bước 4: (a khác 0) Ta kết luận phương trình có nghiệm x=-b/a

II.3.2 Ngôn ngữ sơ đồ (Lưu đồ)

Ngôn ngữ sơ đồ (lưu đồ) là một ngôn ngữ đặc biệt dùng để mô tả giải thuật bằng các sơ đồ hình khối. Mỗi khối qui định một hành động.

Khối	Tác dụng (Ý nghĩa của hành động)	Khối	Tác dụng (Ý nghĩa của hành động)
	Bắt đầu/ Kết thúc		Đường đi
	Nhập / Xuất		Chương trình con
	Thi hành		Khối nối
	Lựa chọn		Lời chú thích

Chẳng hạn ta dùng lưu đồ để biểu diễn giải thuật tìm UCLN nêu trên như sau:



II.3.3 Một số giải thuật cơ bản

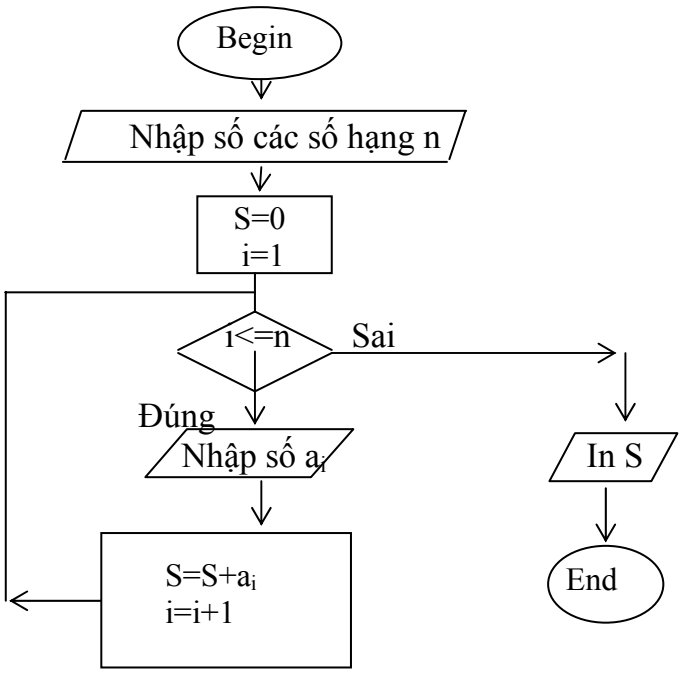
Ví dụ 1: Cần viết chương trình cho máy tính sao cho khi thực hiện chương trình đó, máy tính yêu cầu người sử dụng chương trình nhập vào các số hạng của tổng (n); nhập vào dãy các số hạng a_i của tổng. Sau đó, máy tính sẽ thực hiện việc tính tổng các số a_i này và in kết quả của tổng tính được.

Yêu cầu: Tính tổng n số $S=a_1+ a_2+a_3+.....+a_n$.

Để tính tổng trên, chúng ta sử dụng phương pháp "cộng tích lũy" nghĩa là khởi đầu cho S=0. Sau mỗi lần nhận được một số hạng a_i từ bàn phím, ta cộng tích lũy a_i vào S (lấy giá trị được lưu trữ trong S, cộng thêm a_i và lưu trở lại vào S). Tiếp tục quá trình này đến khi ta tích lũy được a_n vào S thì ta có S là tổng các a_i. Chi tiết giải thuật được mô tả bằng ngôn ngữ tự nhiên như sau:

- Bước 1: Nhập số các số hạng n.
- Bước 2: Cho S=0 (lưu trữ số 0 trong S)
- Bước 3: Cho i=1 (lưu trữ số 1 trong i)
- Bước 4: Kiểm tra nếu i<=n thì thực hiện bước 5, ngược lại thực hiện bước 8.
- Bước 5: Nhập a_i
- Bước 6: Cho S=S+a_i (lưu trữ giá trị S + a_i trong S)
- Bước 7: Tăng i lên 1 đơn vị và quay lại bước 4.
- Bước 8: In S và kết thúc chương trình.

Chí tiết giải thuật bằng lưu đồ:

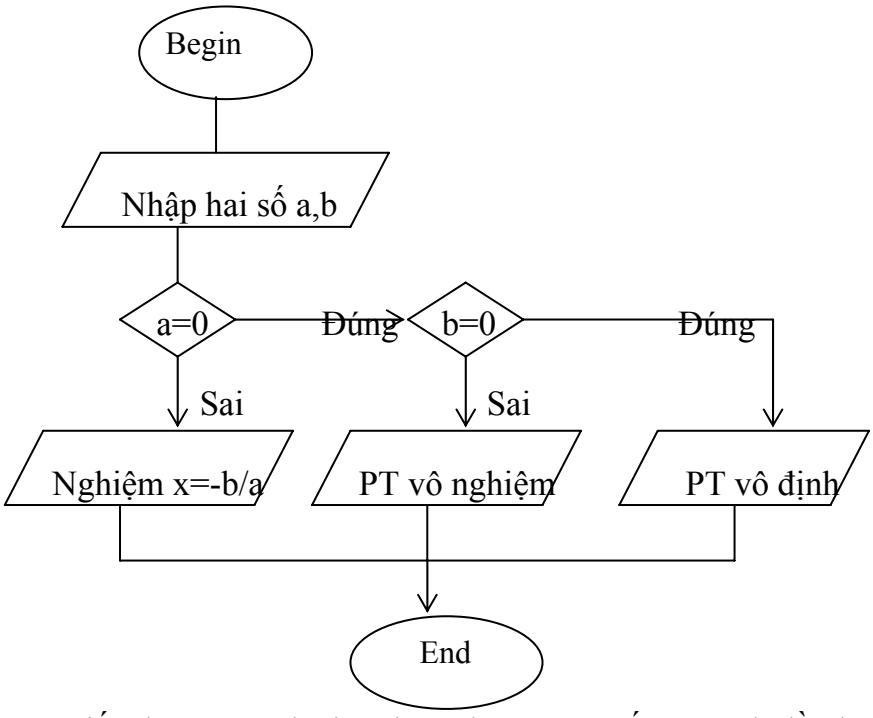


Ví dụ 2: Viết chương trình cho phép nhập vào 2 giá trị a, b mang ý nghĩa là các hệ số a, b của phương trình bậc nhất. Dựa vào các giá trị a, b đó cho biết nghiệm của phương trình bậc nhất ax + b = 0.

Mô tả giải thuật bằng ngôn ngữ tự nhiên:

- Bước 1: Nhập 2 số a và b
- Bước 2: Nếu a = 0 thì thực hiện bước 3, ngược lại thực hiện bước 4

- Bước 3: Nếu b=0 thì thông báo phương trình vô số nghiệm và kết thúc chương trình, ngược lại thông báo phương trình vô nghiệm và kết thúc chương trình.
- Bước 4: Thông báo nghiệm của phương trình là –b/a và kết thúc.



Ví dụ 3: Viết chương trình cho phép nhập vào 1 số n, sau đó lần lượt nhập vào n giá trị a_1, a_2,…,a_n. Hãy tìm và in ra giá trị lớn nhất trong n số a_1, a_2, …, a_n.

Để giải quyết bài toán trên, chúng ta áp dụng phương pháp "thử và sửa". Ban đầu giả sử a_1 là số lớn nhất (được lưu trong giá trị max); sau đó lần lượt xét các a_i còn lại, nếu a_i nào lớn hơn giá trị max thi lúc đó max sẽ nhận giá trị là a_i. Sau khi đã xét hết các a_i thì max chính là giá trị lớn nhất cần tìm.

Mô tả giải thuật bằng ngôn ngữ tự nhiên:

- Bước 1: Nhập số n
- Bước 2: Nhập số thứ nhất a_1
- Bước 3: Gán max=a_1
- Bước 4: Gán i=2
- Bước 5: Nếu i<=n thì thực hiện bước 6, ngược lại thực hiện bước 9
- Bước 6: Nhập a_i
- Bước 7: Nếu max < a_i thì gán max=a_i.
- Bước 8: Tăng i lên một đơn vị và quay lại bước 5
- Bước 9: In max - kết thúc

Phần mô tả giải thuật bằng lưu đồ, sinh viên tự làm xem như bài tập.

Ví dụ 4: Viết chương trình cho phép nhập vào 1 số n, sau đó lần lượt nhập vào n giá trị a_1, a_2,…,a_n. Sắp theo thứ tự tăng dần một dãy n số a_1, a_2,...a_n nói trên. Có rất

nhiều giải thuật để giải quyết bài toán này. Phần trình bày dưới đây là một phương pháp.

Giả sử ta đã nhập vào máy dãy n số a_1, a_2,..., a_n. Việc sắp xếp dãy số này trải qua (n-1) lần:

- Lần 1: So sánh phần tử đầu tiên với tất cả các phần tử đứng sau phần tử đầu tiên. Nếu có phần tử nào nhỏ hơn phần tử đầu tiên thì đổi chỗ phần tử đầu tiên với phần tử nhỏ hơn đó. Sau lần 1, ta được phần tử đầu tiên là phần tử nhỏ nhất.

- Lần 2: So sánh phần tử thứ 2 với tất cả các phần tử đứng sau phần tử thứ 2. Nếu có phần tử nào nhỏ hơn phần tử thứ 2 thì đổi chỗ phần tử thứ 2 với phần tử nhỏ hơn đó. Sau lần 2, ta được phần tử đầu tiên và phần tử thứ 2 là đúng vị trí của nó khi sắp xếp.

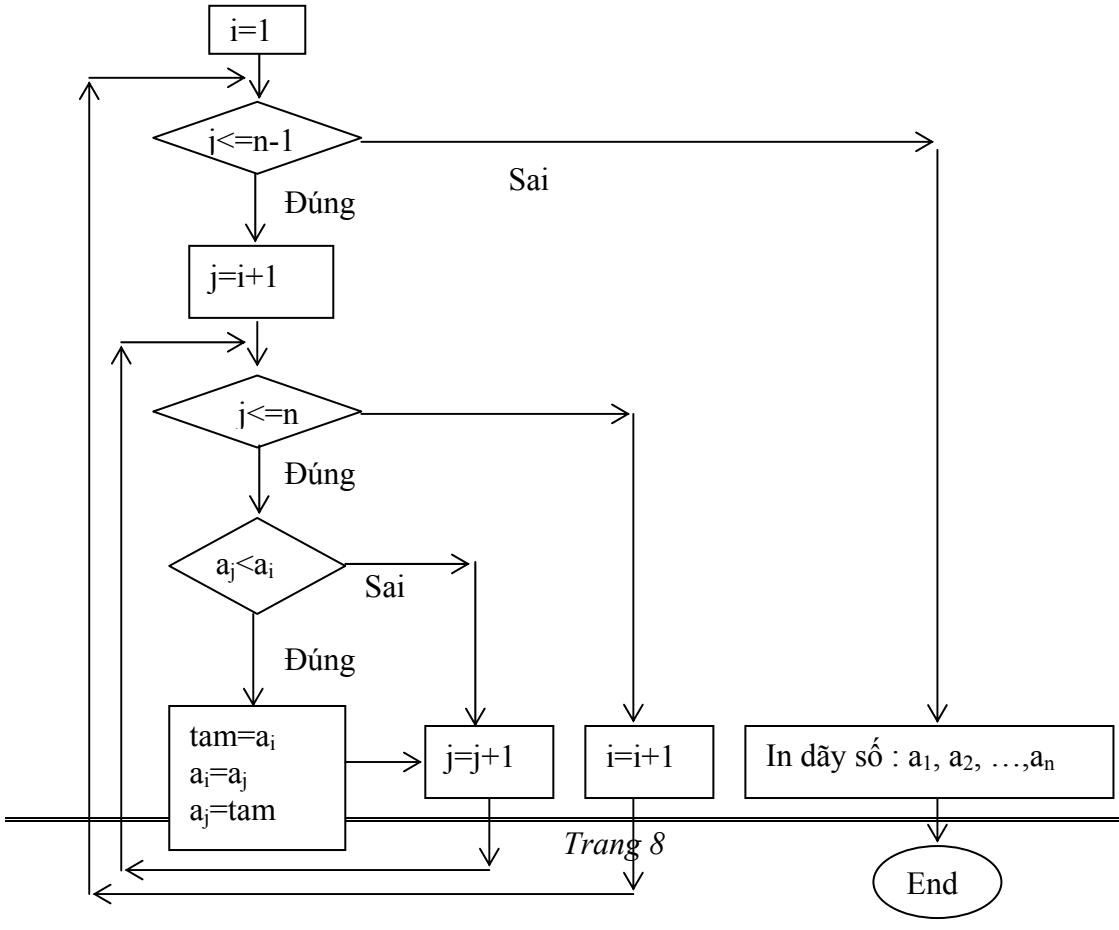
- …

- Lần (n-1): So sánh phần tử thứ (n-1) với phần tử đứng sau phần tử (n-1) là phần tử thứ n. Nếu phần tử thứ n nhỏ hơn phần tử thứ (n-1) thì đổi chỗ 2 phần tử này. Sau lần thứ (n-1), ta được danh sách gồm n phần tử được sắp thứ tự.

Mô tả giải thuật bằng ngôn ngữ tự nhiên:

- Bước 1: Gán i=1
- Bước 2: Gán j=i+1
- Bước 3: Nếu i <=n-1 thì thực hiện bước 4, ngược lại thực hiện bước 8
- Bước 4: Nếu j <=n thì thực hiện bước 5, ngược lại thì thực hiện bước 7.
- Bước 5: Nếu $a_i > a_j$ thì hoán đổi a_i và a_j cho nhau (nếu không thì thôi).
- Bước 6: Tăng j lên một đơn vị và quay lại bước 4
- Bước 7: Tăng i lên một đơn vị và quay lại bước 3
- Bước 6: In dãy số a_1, a_2,..., a_n - Kết thúc.

Mô tả giải thuật sắp xếp bằng lưu đồ



II.4 Các cấu trúc suy luận cơ bản của giải thuật

Giải thuật được thiết kế theo ba cấu trúc suy luận cơ bản sau đây:

***II.4.1 Tuần tự (Sequential)*: Các công việc được thực hiện một cách tuần tự, công việc này nối tiếp công việc kia.**

II.4.2 Cấu trúc lựa chọn (Selection)

Lựa chọn một công việc để thực hiện căn cứ vào một điều kiện nào đó. Có một số dạng như sau:

- *Cấu trúc 1*: Nếu < điều kiện> (đúng) thì thực hiện <công việc>

- *Cấu trúc 2*: Nếu < điều kiện> (đúng) thì thực hiện <công việc 1>, ngược lại (điều kiện sai) thì thực hiện <công việc 2>

- *Cấu trúc 3*: Trường hợp < i> thực hiện <công việc i>

II.4.3. Cấu trúc lặp (Repeating)

Thực hiện lặp lại một công việc không hoặc nhiều lần căn cứ vào một điều kiện nào đó. Có hai dạng như sau:

- *Lặp xác định*: là loại lặp mà khi viết chương trình, người lập trình đã xác định được công việc sẽ lặp bao nhiêu lần.

- *Lặp không xác định*: là loại lặp mà khi viết chương trình người lập trình chưa xác định được công việc sẽ lặp bao nhiêu lần. Số lần lặp sẽ được xác định khi chương trình thực thi.

Trong một số trường hợp người ta cũng có thể dùng các cấu trúc này để diễn tả một giải thuật.

III. KIỂU DỮ LIỆU

Các số liệu lưu trữ trong máy tính gọi là *dữ liệu* (data). Mỗi đơn vị dữ liệu thuộc một kiểu dữ liệu nào đó.

Kiểu dữ liệu là một tập hợp các giá trị có cùng một tính chất và tập hợp các phép toán thao tác trên các giá trị đó. Người ta chia kiểu dữ liệu ra làm 2 loại: kiểu dữ liệu sơ cấp và kiểu dữ liệu có cấu trúc.

III.1 Kiểu dữ liệu sơ cấp

Kiểu dữ liệu sơ cấp là kiểu dữ liệu mà giá trị của nó là đơn nhất.

Ví dụ: Trong ngôn ngữ lập trình C, kiểu int gọi là kiểu sơ cấp vì kiểu này bao gồm các số nguyên từ -32768 đến 32767 và các phép toán +, -, *, /, %…

III.2 Kiểu dữ liệu có cấu trúc

Kiểu dữ liệu có cấu trúc là kiểu dữ liệu mà các giá trị của nó là sự kết hợp của các giá trị khác.

Ví dụ : Kiểu chuỗi ký tự trong ngôn ngữ lập trình C là một kiểu dữ liệu có cấu trúc.

Các ngôn ngữ lập trình đều có những kiểu dữ liệu do ngôn ngữ xây dựng sẵn, mà ta gọi là các kiểu chuẩn. Chẳng hạn như kiểu int, char… trong C; integer, array… trong Pascal. Ngoài ra, hầu hết các ngôn ngữ đều cung cấp cơ chế cho phép người lập trình định nghĩa kiểu của riêng mình để phục vụ cho việc viết chương trình.

IV. NGÔN NGỮ LẬP TRÌNH

IV.1. Khái niệm ngôn ngữ lập trình

Ngôn ngữ lập trình là một ngôn ngữ dùng để viết chương trình cho máy tính. Ta có thể chia ngôn ngữ lập trình thành các loại sau: ngôn ngữ máy, hợp ngữ và ngôn ngữ cấp cao.

Ngôn ngữ máy (machine language): Là các chỉ thị dưới dạng nhị phân, can thiệp trực tiếp vào trong các mạch điện tử. Chương trình được viết bằng ngôn ngữ máy thì có thể được thực hiện ngay không cần qua bước trung gian nào. Tuy nhiên chương trình viết bằng ngôn ngữ máy dễ sai sót, cồng kềnh và khó đọc, khó hiểu vì toàn những con số 0 và 1.

Hợp ngữ (assembly language): Bao gồm tên các câu lệnh và quy tắc viết các câu lệnh đó. Tên các câu lệnh bao gồm hai phần: phần mã lệnh (viết tựa tiếng Anh) chỉ phép toán cần thực hiện và địa chỉ chứa toán hạng của phép toán đó. Ví dụ:

```
INPUT a ; Nhập giá trị cho a từ bàn phím
LOAD a ; Đọc giá trị a vào thanh ghi tổng A
PRINT a; Hiển thị giá trị của a ra màn hình.
INPUT b
ADD b; Cộng giá trị của thanh ghi tổng A với giá trị b
```

Trong các lệnh trên thì INPUT, LOAD, PRINT, ADD là các mã lệnh còn a, b là địa chỉ. Để máy thực hiện được một chương trình viết bằng hợp ngữ thì chương trình đó phải được dịch sang ngôn ngữ máy. Công cụ thực hiện việc dịch đó được gọi là Assembler.

Ngôn ngữ cấp cao (High level language): Ra đời và phát triển nhằm phản ánh cách thức người lập trình nghĩ và làm. Rất gần với ngôn ngữ con người (Anh ngữ) nhưng chính xác như ngôn ngữ toán học. Cùng với sự phát triển của các thế hệ máy tính, ngôn ngữ lập trình cấp cao cũng được phát triển rất đa dạng và phong phú, việc lập trình cho máy tính vì thế mà cũng có nhiều khuynh hướng khác nhau: lập trình cấu trúc, lập trình hướng đối tượng, lập trình logic, lập trình hàm... Một chương trình viết bằng ngôn ngữ cấp cao được gọi là chương trình nguồn (source programs). Để máy tính "hiểu" và thực hiện được các lệnh trong chương trình nguồn thì phải có một chương trình dịch để dịch chương trình nguồn (viết bằng ngôn ngữ cấp cao) thành dạng chương trình có khả năng thực thi.

IV.2 Chương trình dịch

Như trên đã trình bày, muốn chuyển từ chương trình nguồn sang chương trình đích phải có chương trình dịch. Thông thường mỗi một ngôn ngữ cấp cao đều có một chương trình dịch riêng nhưng chung quy lại thì có hai cách dịch: thông dịch và biên dịch.

Thông dịch (interpreter): Là cách dịch từng lệnh một, dịch tới đâu thực hiện tới đó. Chẳng hạn ngôn ngữ LISP sử dụng trình thông dịch.

Biên dịch (compiler): Dịch toàn bộ chương trình nguồn thành chương trình đích rồi sau đó mới thực hiện. Các ngôn ngữ sử dụng trình biên dịch như Pascal, C...

Giữa thông dịch và biên dịch có khác nhau ở chỗ: Do thông dịch là vừa dịch vừa thực thi chương trình còn biên dịch là dịch xong toàn bộ chương trình rồi mới thực thi nên chương trình viết bằng ngôn ngữ biên dịch thực hiện nhanh hơn chương trình viết bằng ngôn ngữ thông dịch.

Một số ngôn ngữ sử dụng kết hợp giữa thông dịch và biên dịch chẳng hạn như Java. Chương trình nguồn của Java được biên dịch tạo thành một chương trình đối tượng (một dạng mã trung gian) và khi thực hiện thì từng lệnh trong chương trình đối tượng được thông dịch thành mã máy.

V. BÀI TẬP

V.1 Mục đích yêu cầu

Làm quen và nắm vững các cách mô tả giải thuật; từ đó đứng trước một bài toán cụ thể, sinh viên có thể mô tả thật chi tiết các bước để giải quyết vấn đề.

V.2 Nội dung

Bằng ngôn ngữ tự nhiên và lưu đồ, anh (chị) hãy mô tả giải thuật cho các bài toán sau:

1. Giải phương trình bậc 2 dạng $ax^2 + bx + c = 0$ với a, b, c là các số sẽ nhập từ bàn phím.

2.Tính tổng bình phương của n số nguyên có dạng sau: S=$a_1^2 + a_2^2 + \cdots + a_n^2$, với n và a_i (i=1..n) là các số sẽ nhập từ bàn phím.

3. Tính tổng có dạng sau: S=$1 - a_1^2 + a_2^2 - a_3^2 + \cdots + (-1)^n a_n^2$, với n và a_i (i=1..n) là các số sẽ nhập từ bàn phím.

4. Trình bày kết quả qua các bước lặp (được mô tả ở trên) để sắp xếp dãy số sau theo thứ tự tăng dần.

a) 12 13 11 10 10 9 8 7 6 5

b) 11 12 13 14 3 4 5 6 7 11 8