Chương 1: TỔNG QUAN VỀ LẬP TRÌNH

1.1. Khái niệm về lập trình

- ***Khái niệm về lập trình***

Lập trình là công việc quan trọng trong quá trình sản xuất phần mềm cho máy tính, thực chất lập trình là viết ra các lệnh theo một cấu trúc nào đó yêu cầu máy thực hiện để thực hiện bài toán đặt ra.

Các lệnh viết ra khi lập trình không thể viết tùy tiện, phải tuân theo một cấu trúc đã định trước. Tức là viết những lệnh nào, thứ tự viết ra sao,... để máy tính thực hiện giải quyết được bài toán đặt ra. Cấu trúc này gọi là thuật toán sẽ được trình bày ở phần sau.

Sau khi lập trình xong ta được chương trình máy tính, như vậy chương trình máy tính là tập hợp các lệnh do chúng ta viết ra để máy thực hiện giải quyết một bài toán, và ta gọi đó là phần mềm(softwere)

Vậy sản phẩm của công việc lập trình có thể gọi là các phần mềm máy tính.

- ***Ngôn ngữ lập trình***

Như đã trình bày, công việc lập trình là viết ra các lệnh yêu cầu máy thực hiện để giải quyết bài toán đặt ra. Nếu sử dụng các lệnh dưới dạng mã máy 0 ,1 thì rất khó viết, khó nhớ và khó sửa khi bị lỗi... Ngôn ngữ lập trình được sinh ra cung cấp cho chúng ta một công cụ để lập trình.

Ngôn ngữ lập trình là tập hợp các ký pháp, quy tắc, qui ước để viết chương trình cho máy tính. Hiện nay có rất nhiều loại ngôn ngữ lập trình được chi thành từng nhóm sau:

- Ngôn ngữ máy: các lệnh dưới dạng mã máy 0 1
- Nhóm ngôn ngữ bậc thấp: Các lệnh gần với mã máy như ngôn ngữ Assembler.
- Nhóm ngôn ngữ bậc cao: các lệnh gần với ngôn ngữ tự nhiên như: Passcal, C, C++, Java ...

Ngôn ngữ lập trình cho phép chúng ta viết chương trình bằng ngôn ngữ tự nhiên, máy tính sẽ không hiểu và không thực hiện dược chương trình do chúng ta viết dưới dạng này. Ngôn ngữ lập trình phải thực hiện chuyển các lệnh trong chương trình thành các lệnh dưới dạng mã máy. Công việc chuyển này gọi là dịch chương trình.

1.2. Khái niệm về thuật toán

- ***Định nghĩa thuật toán***

Thuật toán là một hệ thống chặt chẽ và rõ ràng các qui tắc nhằm xác định một dãy các thao tác trên những đối tượng, sao cho sau một số hữu hạn các bước thực hiện các thao tác này, ta thu được kết quả mong muốn.

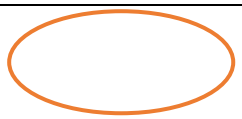
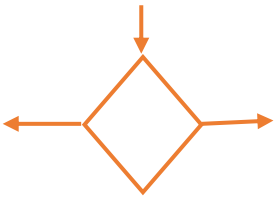
➢ ***Các đặc trưng của thuật toán***

- Tính kết thúc: Thuật toán phải được kết thúc sau một số hữu hạn thao tác.
- Tính rõ ràng, chặt chẽ: Các thao tác được trình bày trong thuật toán phải rõ ràng, phải thực hiện được bằng máy tính. Các bước này có thứ tự nhất định, nếu thay đổi trật tự này thì thuật toán sẽ bị sai.
- Tính phổ dụng: Thuật toán có thể áp dụng được cho lớp các bài toán tương ứng, tức là không chỉ giải quyết duy nhất một bài toán đã đặt ra.
- Tính hiệu quả: Thể hiện trong cả không gian và thời gian. Về không gian thuật toán phải thực hiện được trong điều kiện khả năng của máy tính hiện tại. Về thời gian đòi hỏi thuật toán phải cho kết quả sớm nhất có thể được.

➢ ***Cách thể hiện thuật toán***

Có hai cách thể hiện thuật toán:

- Cách thứ nhất là nêu trình tự các bước từ bước 1 đến bước cuối cùng, mỗi bước trình bày các thao tác phải làm bằng ngôn ngữ tự nhiên, càng chi tiết càng tốt.
- Cách thứ hai: sử dụng sơ đồ khối trong đó sử dụng các hình vẽ với ý nghĩa như sau:

Hình vẽ	Ý nghĩa
	Thể hiện sự bắt đầu và kết thúc thuật toán
	Thể hiện sự tính toán
	Thể hiện cho thao tác nhập dữ liệu, đưa kết quả dữ liệu ra.
	Thể hiện chiều đi của thuật toán
	Thể hiện sự lựa chọn đúng hoặc sai, có một chiều đi vào và 2 chiều đi ra tương ứng với 2 trương hợp

Một thuật toán được trình bày dưới dạng sơ đồ khối sẽ rõ ràng và tường minh hơn, người lập trình sẽ dễ dàng triển khai chương trình từ thuật toán

1.3. Quá trình xây dựng một phần mềm

- ***Phát biểu bài toán cần xây dựng***

Trong thực tế có rất nhiều bài toán về mọi lĩnh vực kinh tế, xã hội, các ngành khoa học, ... trong số đó có những bài toán đang thực hiện bằng sức lao động của con người, có những bài toán đòi hỏi làm việc trong môi trường đặc biệt mà con người không thể tham gia. Nhờ có máy tính cùng với việc sản xuất phần mềm để nhờ máy tính thực hiện thay thế giúp con người đem lại hhiệu quả cho cá nhân, cho xã hội.

Như vậy đầu tiên chúng ta phải xác định bài toán cần phải viết chương trình để cho máy thực hiện, bao gồm: Bài toán cho biết những thông tin gì, cung cấp những công cụ nào, những cái này gọi là đầu vào của bài toán(INPUT). Sau đó xác định những kết quả cần phải đưa ra sau khi thực hiện gọi là đầu ra(OUTPUT)

- ***Phân tích và thiết kế***

Từ bài toán ở trên chúng ta phải phân tích rõ những yếu tố của bài toán. Quan trọng ở đây là xác định rõ các dữ liệu đầu vào, các dữ liệu đầu ra một cách cụ thể. Sau đó hãy thiết kế một thuật toán để thực hiện bài toán và trình bày nó dưới dạng một trong hai cách ở trên.

Để thực hiện giai đoạn này một cách thành công chúng ta cần phải mắm vững kiến thức về: Phân tích thiết kế hệ thống, cấu trúc dữ liệu và giải thuật ...

- ***Lập chương trình***

Sau khi có được thuật toán ở giai đoạn thứ 2 chúng ta lựa chọn một ngôn ngữ lập trình nào đó để viết ra chương trình cho máy tính. Công việc lập trình chiếm một khối lượng lớn thời gian trong quá trình sản xuất phần mềm, vì vậy đòi hỏi nhiều nhân lực cho giai đoạn này.

- ***Chạy thử và kiểm tra***

Sau khi lập xong chương trình chúng ta phải chạy thử trên máy tính và kiểm tra kết quả. Nếu chưa đúng thì phải xem xét lại thuật toán hoặc chương trình. Công việc chạy thử và kiểm tra khá quan trọng, để đảm bảo cho chương trình thực hiện đúng đắn và ổn định trong mọi trường hợp.

1.4. Bài tập thực hành

Thuật toán hoán vị hai phần tử

Bài toán phát biểu như sau: Cho 2 số a và b có 2 giá trị khác nhau, đổi giá trị 2 số cho nhau.

Cách 1: sử dụng số chung gian

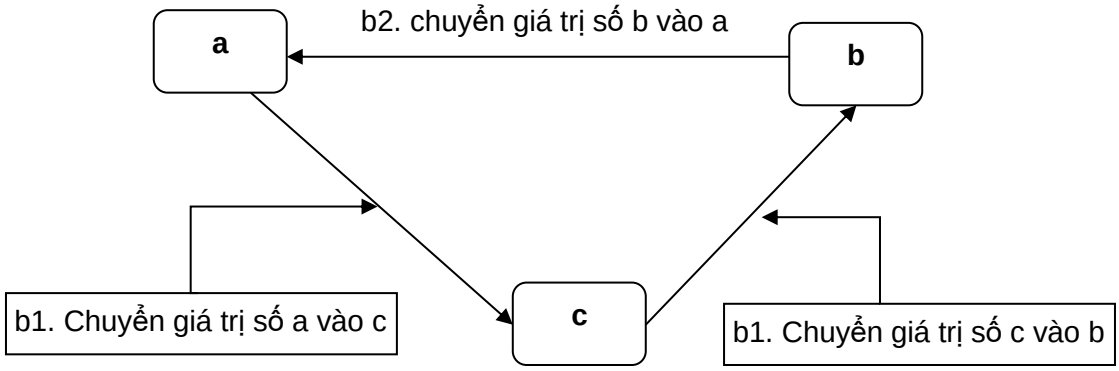
Để thực hiện thuật toán này chúng ta phải thêm một sô chung gian thứ 3, ta kí hiệu là số c. Thuật toán được trình như sau.

b1. Gán giá trị của số a vào số c.

b2. Gán giá trị của số b vào số a.

b3. Gán giá trị của số c vào số b.

Hình minh họa thuật toán như sau:



Cách 2: Không sử dụng số chung gian.

Sử phép cộng " + " trừ " - " để hoán vị hai phần tử thuật toán được trình bày như sau:

b1: Gán a = a + b;

b2: Gán b = a - b;

b3: Gán a = a - b;

Thuật toán tìm số lớn nhất, nhỏ nhất

Bài toán được phát biểu như sau:

- Cho một tập hợp có n phần tử A={X1, X2, X3, ..., Xn} (n>1)
- Tìm phần tử lớn nhất (Max) và phần tử bé nhất (Min) có trong tập hợp A

Thuật toán tìm Max và Min tương tự như nhau. Sau đây trình bày thuật toán tìm ***Min***

Ta kí hiệu Xi là số thứ i trong dãy trên với i= 1,2,3,… ,n và sử dụng ***Min*** để lưu số nhỏ nhất, giải thuật được trình bày như sau:

Bước 1: Xác định n và nhập các giá trị X1, X2, …, Xn

Bước 2: Nếu n < 1 thì thông báo không có các số và kết thúc.

Bước 3: Giả sử sô nhỏ nhất là số đầu tiên (đặt ***Min*** = X1).

Bước 4: Đặt i = 2 bắt đầu kiểm tra từ số thứ 2.

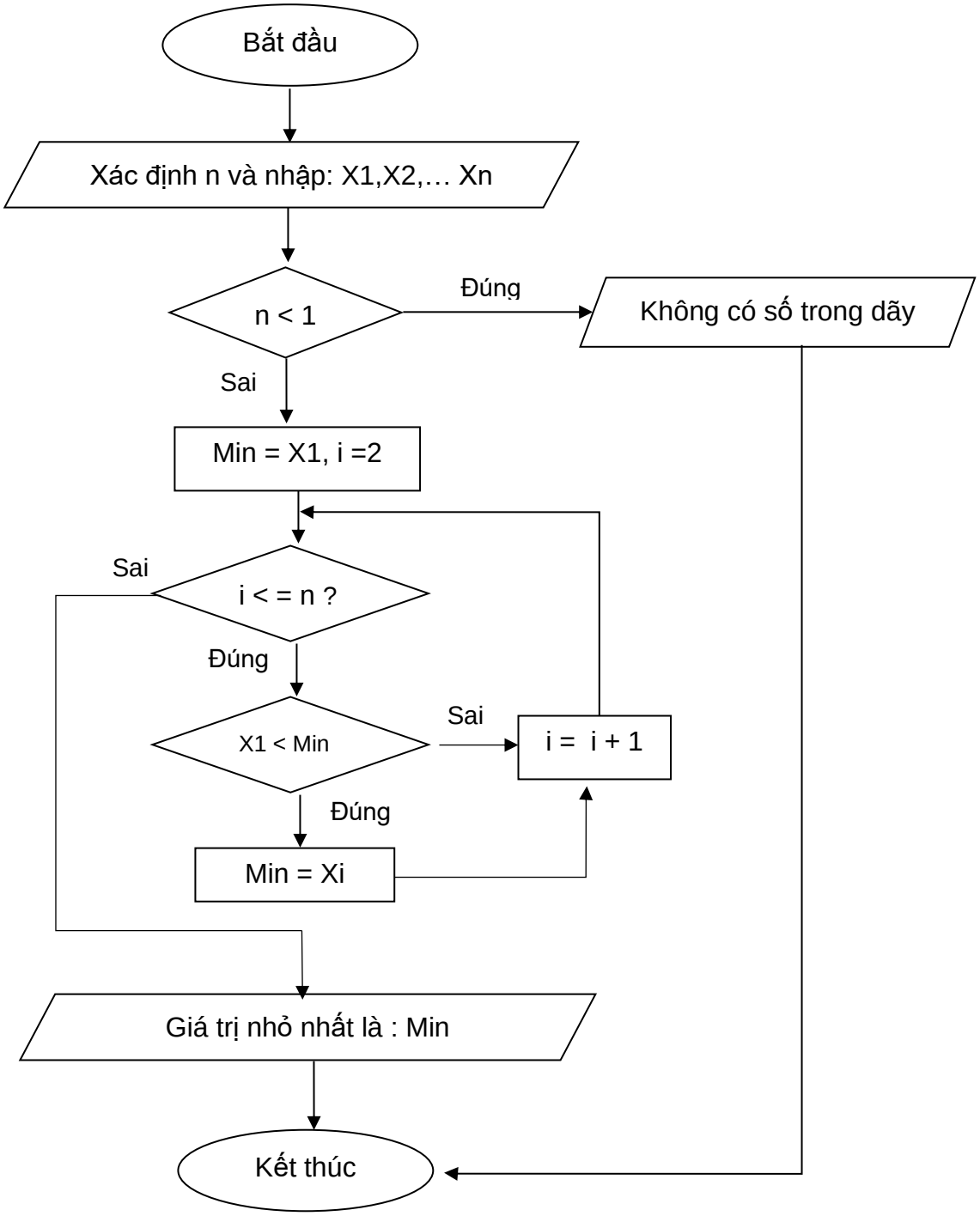
Bước 5: Nếu i > n thì chuyển đến bước 8.

Bước 6: Nếu Xi < ***Min*** thì đặt lại ***Min*** = Xi.

Bước 7: Tăng i lên 1, (i = i + 1) và quay lại bước 5.

Bước 8: Thông báo kết quả là **Min** và kết thúc.

Sơ đồ khối của thuật toán:



Chương 2: TỔNG QUAN VỀ NGÔN NGỮ C

2.1. Giới thiệu về ngôn ngữ C

- ***Lịch sử phát triển***

C là ngôn ngữ lập trình cấp cao, được sử dụng rất phổ biến để lập trình hệ thống cùng với Assembler và phát triển các ứng dụng.

Vào những năm cuối thập kỷ 60 đầu thập kỷ 70 của thế kỷ XX, Dennish Ritchie (làm việc tại phòng thí nghiệm Bell) đã phát triển ngôn ngữ lập trình C dựa trên ngôn ngữ BCPL (do Martin Richards đưa ra vào năm 1967) và ngôn ngữ B (Do Ken Thompson phát triển từ ngôn ngữ BCPL vào năm 1970 khi viết hệ điều hành UNIX) của máy DEC PDP-11.

Năm 1978, Dennish Ritchie và B.W Kernighan đã cho xuất bản quyển "Ngôn ngữ lập trình C" và được phổ biến rộng rãi đến ngày nay.

Lúc ban đầu, C được thiết kế nhằm lập trình trong môi trường của hệ điều hành UNIX nhằm mục đích cho công việc lập trình phức tạp. Nhưng về sau, với những nhu cầu phát triển ngày một tăng của công việc lập trình, C đã vượt qua khuôn khổ của phòng thí nghiệm Bell và nhanh chóng hội nhập vào thế giới lập trình để rồi các công ty lập trình sử dụng một cách rộng rãi. Sau đó, các công ty sản xuất phần mềm lần lượt đưa ra các phiên bản hỗ trợ cho việc lập trình bằng ngôn ngữ C và chuẩn ANSI C cũng được khai sinh từ đó.

- ***Sự cần thiết***

Ngôn ngữ lập trình C đã làm thay đổi thế giới máy tính. Sự tác động của nó nhiều khi không được đánh giá đúng mức. Nó thay đổi cách tiếp cận và nhìn nhận về lập trình một cách cơ bản. Sự ra đời của C là kết quả trực tiếp của sự cần thiết phải thay đổi về cấu trúc, hiệu quả, ngôn ngữ bậc cao thay thế cho mã máy khi viết các chương trình hệ thống. Như chúng ta biết, khi thiết kế một ngôn ngữ lập trình, sự cân nhắc luôn được xem xét giữa:

- Tính dễ dàng sử dụng và sức mạnh
- Sự an toàn
- Sự chặt chẽ và khả năng mở rộng

Trước khi có C, lập trình viên thường xuyên phải chọn giữa các ngôn ngữ đã thu gọn một tập các đặc điểm. Ví dụ, mặc dù FORTRAN có thể được dùng để viết các chương trình hiệu quả cho các ứng dụng khoa học, nhưng nó không tốt lắm cho viết mã hệ thống. Trong khi BASIC dễ dàng cho người học thì nó lại không đủ mạnh và thiếu các cấu trúc cho các chương trình lớn. Ngôn ngữ Assembly có thể được dùng để viết các chương trình hiệu quả cao nhất nhưng không dễ dàng để sử dụng cũng như gỡ rối rất khó khăn.

Một số vấn đề khác là các ngôn ngữ lập trình như BASIC, COBOL, không được thiết kế dựa trên các yếu tố cấu trúc. Chúng dựa trên lệnh GOTO để điều khiển chương trình. Vì vậy mà các chương trình được viết bởi ngôn ngữ này hướng tới

việc sinh ra mã spagheti – một khối lộn xộn các lệnh nhảy và rẽ nhánh làm cho chương trình không thể hiểu được.

Trong khi ngôn ngữ như Pascal là ngôn ngữ cấu trúc, nó không được thiết kế để đạt hiệu quả cao, và thiếu các đặc điểm cần thiết để ứng dụng cho những chương trình lớn. Vì thế, trước khi có ngôn ngữ C, không có ngôn ngữ nào có thể dung hòa được sự xung đột đã tồn tại trước đó.

Ngôn ngữ C là một ngôn ngữ lập trình hệ thống rất mạnh mẽ và rất "mềm dẻo", có một thư viện gồm rất nhiều các hàm (function) đã được tạo sẵn. Người lập trình có thể tận dụng các hàm này để giải quyết các bài toán mà không cần phải tạo mới. Hơn thế nữa, ngôn ngữ C hỗ trợ rất nhiều phép toán nên phù hợp cho việc giải quyết các bài toán kỹ thuật có nhiều công thức phức tạp. Ngoài ra, C cũng cho phép người lập trình tự định nghĩa thêm các kiểu dữ liệu trìu tượng khác. Tuy nhiên, điều mà người mới học lập trình C thường gặp "rắc rối" là "hơi khó hiểu" do sự "mềm dẻo" của C. Dù vậy, C được phổ biến khá rộng rãi và đã trở thành một công cụ lập trình khá mạnh, được sử dụng như là một ngôn ngữ lập trình chủ yếu trong việc xây dựng những phần mềm hiện nay.

2.2. Các khái niệm cơ bản

2.2.1. Bảng ký tự sử dụng trong C

Bảng ký tự sử dụng trong ngôn ngữ lập trình C bao gồm:

- Nhóm các chữ cái: Chữ in hoa A, B, C, ... Z và chữ in thường a, b, c ... z
- Nhóm các chữ số: 0,1,2,3, ..., 9
- Nhóm các dấu: +, -, *, /, >, <, &, ...

Chú ý: Ngôn ngữ C phân biệt chữ hoa và chữ thường do đó phải thận trọng khi sử dụng các chữ cái hoa và thường.

2.2.2. Tên và từ khóa

Tên là một dãy các chữ cái, chữ số và dấu gạch nối. Tên phải bắt đầu bằng chữ cái hoặc dấu gạch nối.

Ví dụ: Yenbai, baitap1 là những tên đúng. 1baitap, yen bai là những tên sai.

Tên có thể do ngôn ngữ C sinh ra để nhằm 2 mục đích: thứ nhất là định danh các thành phần có sẵn, thứ hai là viết các lệnh trong chương trình. Tuy nhiên tên có thể do người lập trình tự đặt để định danh các thành phần của bản thân người lập trình tạo ra trong chương trình.

Từ khóa(key word) là các tên do ngôn ngữ sinh ra để viết các lệnh, để định danh các thành phần đặc biệt.

Ví dụ: for, while, if ...

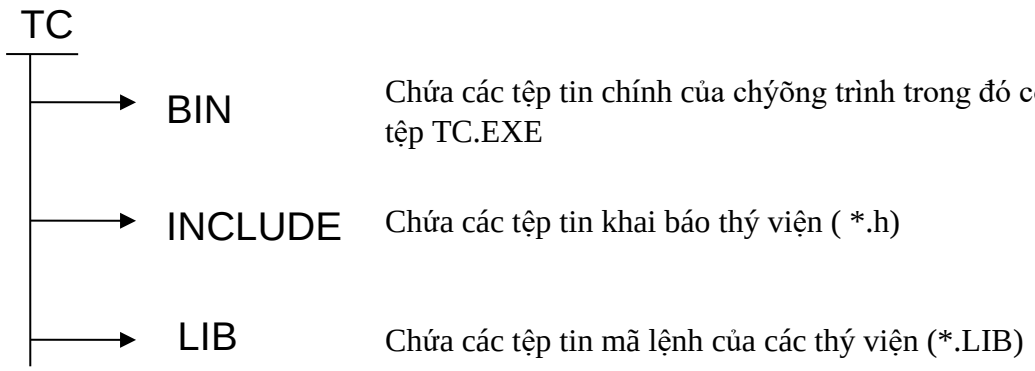
Chú ý: Tên do chúng ta đặt ra không được phép trùng với từ khóa.

2.3. Môi trường lập trình Turbo C

2.3.1. Giới thiệu TurboC

Turbo C là môi trường hỗ trợ lập trình C do hãng Borland cung cấp. Môi trường này cung cấp các chức năng như: soạn thảo chương trình, dịch, thực thi chương trình…Phiên bản sử dụng ở đây là Turbo C 3.0.

- Sau khi cài đặt song ta có thư mục TC hoặc TC30 trong thư mục này có các thư mục con sau.



2.3.2. Các thao tác lập trình trên TC

➢ ***Khởi động***

Để khởi động chương trình này chúng ta chạy tệp tin TC.EXE trong thư mục Bin của TurboC.

➢ ***Thoát khỏi***

Để thoát khỏi Turbo C và trở về DOS hay Windows có 2 cách:

- Cách 1: Vào menu File/Exit
- Cách 2 : Dùng phím tắt : Alt + X

➢ ***Tạo mới, ghi một chương trình C***

Muốn soạn thảo một chương trình mới ta chọn mục *New* trong menu *File* (File ->New). Trên màn hình sẽ xuất hiện một vùng trống để cho ta soạn thảo nội dung của chương trình.

➢ ***Sử dụng trợ giúp***

Trên menu Help bao gồm các lệnh gọi trợ giúp khi người lập trình cần giúp đỡ một số vấn đề nào đó như: Cú pháp câu lệnh, cách sử dụng các hàm có sẵn…

- Lệnh Contents: Hiển thị toàn bộ nội dung của phần help.
- Lệnh Index: Hiển thị bảng tìm kiếm theo chỉ mục.

Chương 3: CÁC THÀNH PHẦN CƠ BẢN

3.1. Các kiểu dữ liệu cơ bản

Chúng ta đã biết máy tính không thể chứa hết mọi thứ thông tin trong thực tiễn, cũng như vậy trong ngôn ngữ lập trình C chỉ cung cấp một số kiểu dữ liệu cơ bản để người lập trình có thể sử dụng trong việc biểu diễn, tính toán và xử lý trong chương trình.

- *Kiểu ký tự:*
  + Tên kiểu: char
  + Miền giá trị:
  + Độ lớn : 1 byte
- *Kiểu số:*
  + Tên kiểu: int (số nguyên), float(số thực)
  + Miền giá trị: -32768 đến 32767(int), Từ $3.4* 10^{-38}$ đến $3.4* 10^{38}$ (float)
  + Độ lớn : 2byte (số nguyên), 4byte(số thực)
- *Kiểu logic:*

  Trong ngôn ngữ C không có kiểu logic cụ thể mà xem xét số nguyên như là kiểu logic. Với giá trị 0 tương ứng với sai, giá trị 1 hoặc khác 0 tương ứng với đúng.

3.2. Biến, hằng, biểu thức

- *Hằng* (const): Là một đại lượng không thay đổi giá trị trong toàn chương trình. Hằng có thể là số nguyên, số thực, ký tự hoặc chuỗi ký tự, để biểu diễn các giá trị hằng ta viết như sau :
  + Đối với số nguyên: viết bình thường như trong toán học, ví dụ: 100 hoặc 150.
  + Đối với số thực: Dùng dấu chấm để ngăn cách phần nguyên với phần thập phân, ví dụ: 1.5 hoặc 0.15
  + Đối với ký tự: Viết ký tự vào trong hai nháy đơn, ví dụ: ‘a’ hoặc ‘B’
  + Đối với chuỗi ký tự: Viết chuỗi ký tự vào trong hai nháy kép, ví dụ: “Yen Bai” hoặc “Xin chao”.
- *Biến nhớ* (varible): Là một đại lượng có thể thay đổi giá trị trong chương trình. Biến nhớ được sử dụng để chứa dữ liệu của chương trình. Các biến trong bộ nhớ ở các thời điểm khác nhau có thể cất giữ các giá trị khác nhau. Mỗi biến nhớ sẽ được quy định một kiểu dữ liệu, với quy định này biến nhớ chỉ có thể chứa được dữ liệu thuộc kiểu đó mà thôi.

  Để sử dụng một biến nhớ nào đó chúng ta phải khai báo, cách thức sẽ được trình bày ở phần sau.
- *Biểu thức* : Là một sự kết hợp giữa các toán tử và các toán hạng để diễn đạt một công thức nào đó. Mỗi biểu thức có một giá trị.

Khi máy tính thực hiện tính toán biểu thức sẽ tuân theo thư tự ưu tiên giống như toán học.

Ví dụ:

(8 +3 * (4-2))/2 =7;

3.3. Các phép toán

Các phép toán trong C gọi là các toán tử (operator), bao gồm:

- Các phép toán số học
  - Ký hiệu: +, -,*, /,%
  - Ý nghĩa: cộng ,trừ, nhân, chia, chia dư
  - Dữ liệu tác động: Kiểu số nguyên hoặc số thực
  - Kết quả: Các phép toán số học cho kết quả là dữ liệu kiểu số

    Ví dụ: 2+5, 5*6, 3/2

Chú ý: đối với phép chia (/) nếu hai vế dữ liệu là số nguyên thì máy sẽ chia lấy phần nguyên, nếu một trong hai vế là số thực thì máy cho kết quả chính xác.

Ví dụ:

7/4 sẽ cho kết quả là 1(là phần nguyên của 7 chia 4)

7.0/4 hoặc 7/4.0 sẽ cho kết quả là 1.75

7%4 sẽ cho kết quả là 3(là phân dư của 7 chia 4)

- Các phép toán quan hệ
  - Ký hiệu: >, <, >=, <=, ==, !=
  - Ý nghĩa : Lớn hơn, nhỏ hơn, lớn hơn hoặc bằng, nhơ hơn hoặc bằng, so sánh bằng, và khác
  - Dữ liệu tác động: là các dữ liệu kiểu số và chữ, nếu là chữ thì máy sẽ so sánh mã ASCII của các chữ đo.
  - Kết quả: phép toán cho kết quả là loogic(đúng hoặc sai)
- Các phép kết nối logic
  - Ký hiệu: &&, ||, !
  - Ý nghĩa: và, hoặc, phủ định
  - Kiểu dữ liệu tác động: là các dữ liệu có giá trị đúng hoặc sai(kiểu logic)
  - Kết quả: cho kết quả logic(đúng hoặc sai)

    Bảng kết quả:

E1	E2	E1&&E2	E1|| E2	!E1

Đúng	Đúng	Đúng	Đúng	Sai
Đúng	Sai	Sai	Đúng	Sai
Sai	Đúng	Sai	Đúng	Đúng
Sai	Sai	Sai	Sai	Đúng

- Phép toán Tăng và giảm (++ & --)

- Toán tử: ++ tăng thêm 1 vào toán hạng của nó.

- Toán tử -- trừ bớt 1 vào toán hạng của nó.

Nói cách khác: x = x + 1 giống như x ++ hay x = x – 1 giống như x --

Cả 2 toán tử tăng và giảm đều có thể tiền tố (đặt trước) hay hậu tố (đặt sau) toán hạng.

Ví dụ: x = x + 1 có thể viết x++ (hay ++x) Tuy nhiên giữa tiền tố và hậu tố có sự khác biệt khi sử dụng trong 1 biểu thức. Khi 1 toán tử tăng hay giảm đứng trước toán hạng của nó, C thực hiện việc tăng hay giảm trước khi lấy giá trị dùng trong biểu thức. Nếu toán tử đi sau toán hạng, C lấy giá trị toán hạng trước khi tăng hay giảm nó.

Ví dụ:

x = 10 ; y = ++x ; // giá trị biến y = 11

x = 10 ; y = x++; // giá trị biến y = 10

Thứ tự ưu tiên của các toán tử số học:

++, -- sau đó là *, /, % rồi mới đến +, -

3.4. Cấu trúc chương trình C

Một chương trình C đơn giản có cấu trúc như sau:

```
Khai báo thư viện
void  main()
{
     - Khai báo biến nhớ
     - Các câu lệnh
}
```

Trong đó:

- Khai báo thư viện: Dùng để nạp các thư viện cần thiết vào chương trình, và được viết theo cú pháp sau:

#include<Tên thư viện>

Ví dụ: `#include<conio.h>`

- Khai báo biến nhớ: Biến nhớ thông thường được khai báo ở phần đầu của chương trình. Tuy nhiên có thể thực hiện bất kỳ chỗ nào nhưng phải trước khi sử dụng chúng. Cú pháp khai báo biến nhớ như sau:

Tên_kiểu_dữ_liệu ␣ tên_biến_nhớ ;

Ví dụ:
```
int a;
float b,c;
```

Có thể dử dụng toán tử gán trong khai báo biến nhớ để gán dữ liệu ban đầu cho biến nhớ như ví dụ sau:

```
int a=5, b=10;
```

Ví dụ về một chương trình C đơn giản:

```
#include<conio.h>
void main()
{
    int a =10 , b=5, c ;
    c= a + b;
    printf("tong 2 so a va b la: %d",c);
}
```

3.5. Bài tập thực hành

Bài 1: Viết các biểu thức sau theo ngôn ngữ C

a) $b^2 – 4ac < 0$

b) y<a-b hoặc y < c+d

c) D = 0 và (D_1 #0 hoặc D_2 =0)

e) a+b>c và b+c>a và c+a>b

Bài 2: Hãy cho biết kế quả của các biểu thức sau

a) 5+ 7/2

b) 15-2*3/4 + 7%2

c) 1978%5 + 2*6

d) (2018/2015 - 12)%2 + 5

e) ((10+(9/1-8)/2+7)/3-6)/4+5

f) (5>2*2)&&(5<=2+3)

Chương 4: CÁC LỆNH CƠ BẢN

4.1. Các lệnh nhập, xuất dữ liệu

4.1.1. Lệnh hiện dữ liệu lên màn hình

- Cú pháp:

printf("điều khiển",các dữ liệu cần hiện);

Trong đó:

- *Điều khiển* : Là các cặp kí tự điều khiển để hiện dữ liệu lên màn hình và phải được viết trong cặp dấu nháy kép, mỗi cặp kí tự điều khiển bao gồm dấu "%" và sau đó là một ký tự định kiểu.

Cách viết	Ý nghĩa
%d	Hiện số nguyên
%c	Hiện ký tự trong bảng mã ASCII
%f	Hiện số thực
%s	Hiện chuỗi ký tự

- *Dữ liệu cần hiện:* Là các biểu thức dữ liệu cần hiện ra màn hình, các biểu thức này cách nhau bởi dấu phẩy.

Để sử dụng được lệnh hiện dữ liệu lên màn hình ta phải nạp thư viện <stdio.h>

Ví dụ:

```
printf("%d",65);
```

thì kết quả hiện ra màn hình sẽ là: 65

```
printf ("%c",65);
```

thì máy sẽ hiện ra ký tự có mã là 65 và đó là: chữ A

Chú ý:

- Để hiện dữ liệu có xuống dòng trên màn hình ta sử dụng \n trong điều khiển của lệnh **printf**

Ví dụ

```
printf("yen bai \n ngày %d",12);
```

màn hình sẽ hiện ra như sau:

```
yen bai
ngay 12
```

- Để cách một khoảng trên màn hình như bấm phím Tab ta sử dụng \t trong điều khiển của lệnh **printf**.

Vidu: `printf("yen bai \n\t ngay %d",12);`

Màn hình sẽ hiện ra như sau:

```
Yen bai
        ngay 12
```

4.1.2. Lệnh nhập dữ liệu từ bàn phím

Cú pháp :

`scanf("điều khiển",& tên biến nhớ);`

Trong đó

- *điều khiển* : Để quy định dữ liệu nhập vào dưới dạng nào, cách viết như điều khiển trong lệnh **printf** .

- *Tên biến nhớ:* Dùng để lưu trữ dữ liệu nhập vào từ bàn phím, phải có dấu & ở trước.

Để sử dụng được lệnh nhập dữ liệu từ bàn phím ta phải nạp thư viện <stdio.h>.

Ví dụ:

```
scanf("%d",&a);
```

Nhập một số nguyên từ bàn phím vào cho biến nhớ a.

Chú ý:

- Có thể nhập nhiều dữ liệu vào nhiều biến trong một lệnh scanf. Ta phải điền vào các điều khiển nhập cùng với các biến nhớ tương ứng cách nhau bởi dấu phẩy.

Ví dụ: `scanf("%d%f",&a,&b);`

Sẽ nhập số nguyên vào biến nhớ a, số thực vào biến nhớ b

- Có thể quy định độ rộng dữ liệu khi nhập, phải viết độ rộng đó vào giữa dấu % và kí tự định kiểu tương ứng. Nếu gõ thừa máy sẽ tự động cắt bỏ.

Ví dụ:

```
scanf("%2d%5f",&a,&b);
```

nhập một số nguyên vào biến nhớ a tối đa là 2 chữ số, nhập số thực vào biến nhớ b, với độ rộng tối đa là 5 chữ số.

4.2. Cấu trúc lệnh rẽ nhánh

Lệnh rẽ nhánh cho phép chương trình thực hiện các khối lệnh tùy theo từng trường hợp cụ thể.

4.2.1. Lệnh if

Dạng đầy đủ

Lệnh rẽ nhánh này cho phép rẽ hai nhánh ứng với trường hợp đúng hoặc sai của biểu thức trọn, cách viết như sau:

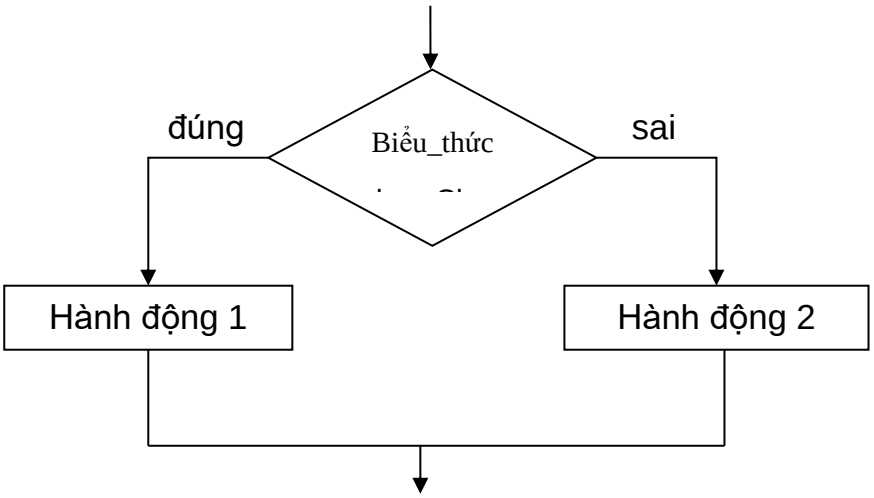
Cú pháp:

if(điều_kiện_lựa_trọn)
Hành_động_1;
else
hành_động_2;

Trong đó:

- Điều kiện lựa trọn: Là biểu thức logic có giá trị đúng hoặc sai
- Hành động 1 và 2: Là các khối lệnh để thực hiện cho trường hợp 1 và 2 tương ứng.

Sơ đồ khối:



Chức năng:

Máy sẽ thực hiện hành động 1 nếu điều kiện lựa chọn có giá trị đúng, ngược lại máy sẽ thực hiện hành động 2.

Ví dụ:

```
if(x%2 ==0)
    printf(" so x là so chan");
else
    printf("so x la so le");
```

Dạng không đầy đủ

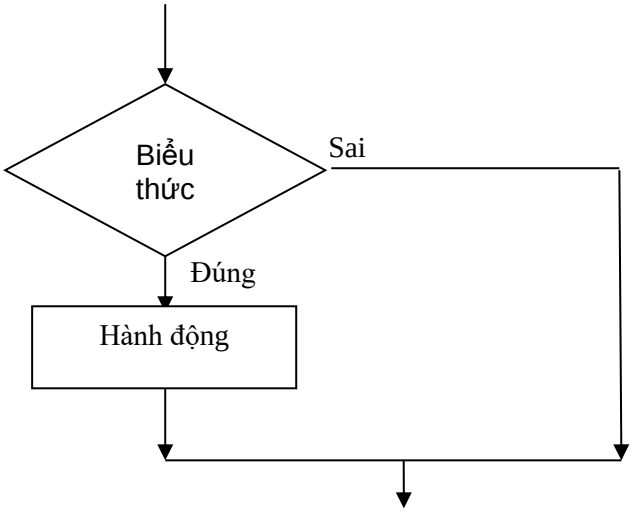
Cú pháp câu lệnh

if(điều_kiện_lựa_trọn)
Hành_động;

Trong đó:

Điều_kiện_lựa_chọn và Hành_động tương tự như trong câu lênh if ở dạng đầy đủ.

Sơ đồ khối



Nguyên tắc hoạt động:

Đầu tiên máy tính toán giá trị của biểu thức. Nếu biểu thức trả về giá trị đúng thì máy tiến hành thực hiện công _việc sau đó tiến hành thực hiện các câu lệnh tiếp theo sau câu lệnh if. Nếu biểu thức trả về giá trị sai thì máy bỏ qua việc thực hiện công_việc trong câu lệnh if mà tiến hành thực hiện ngay các công việc sau câu lệnh if.

Chú ý:

- Mỗi hành động có thể có một hoặc nhiều lệnh, nếu nhiều lệnh phải có cặp dấu đóng mở ngoặc nhọn { }

<u>ví dụ:</u>

```
int a,b;
printf("Nhap vao gia tri cua 2 so a, b!");
scanf("%d%d",&a,&b);
if (a>b)
{
   printf("\n Gia tri cua a lon hon gia tri cua b");
   printf("\n a=%d, b=%d",a,b);
```

```
}
```

4.2.2. Lệnh switch

Đây là lệnh để lựa chọn giữa các trường hợp tương tự If-else nhưng khác ở chỗ lệnh switch chỉ chấp nhận biểu thức hằng.

Cấu trúc:

```
switch (biểu thức)
{
        case <hằng 1>:  các lệnh;  break;
        case <hằng 2>:  các lệnh;  break;
        case <hằng n>:  các lệnh;  break;
        default:  các lệnh;
}
```

Các qui tắc của lệnh switch tronng ngôn ngữ C

Biểu thức nguyên trong switch được tính toán và kiểm tra lần lượt với giá trị của từng case. Đầu tiên, nó sẽ được so sánh với giá trị của case đầu tiên, nếu bằng nhau thì sẽ thực hiện các lệnh (statement) trong case này cho đến khi nó gặp được từ khoá break. Khi đó, cấu trúc switch…case kết thúc. Chương trình sẽ thực hiện tiếp những dòng lệnh sau cấu trúc switch…case. Ngược lại, nếu như giá trị biểu thức nguyên không bằng giá trị case đầu tiên thì nó sẽ tiếp tục so sánh đến giá trị của case thứ hai và tiếp tục thực hiện như những bước trên. Giả sử, đến cuối cùng vẫn không tìm được giá trị bằng nó thì các khối lệnh trong default sẽ được thực hiện nếu như có tồn tại default.

Chú ý

- Biểu thức trong lệnh switch phải là kiểu dữ liệu số nguyên, ký tự.
- Lệnh break trong switch không bắt buộc.(Lệnh break; ở sau mỗi case giúp cho chương trình thoát khỏi switch mà không tiếp tục chạy các case ở dưới.)

Ví dụ: Nhập vào số, in ra thứ tương tự với số đó trong một tuần

```
int thu;
printf("Nhap vao mot thu trong tuan");
scanf("%d",&thu);
switch(thu)
{
case 2:   printf("Thu 2 ");   break;
case 3:   printf("Thu 3 ");   break;
case 4:   printf("Thu 4 ");   break;
case 5:   printf("Thu 5 ");   break;
case 6:   printf("Thu 6 ");   break;
```

```
case 7:   printf("Thu 7 ");   break;
case 8:   printf("Chủ nhật");   break;
default:
    printf("Thu ban nhap khong hop le !");
}
```

4.2.3. Toán tử 3 ngôi

Toán tử điều kiện 3 ngôi là cấu trúc thay thế của biểu thức điều kiện if - else

Cú pháp:

[Biểu thức điều kiện] ? [Biểu thức 2] : [biểu thức 3];

Trong đó:

- Biểu thức điều kiện là phép toán so sánh logic, kết quả trả về true hoặc false.
- Nếu kết quả trả về của biểu thức điều kiện là true, thì biểu thức 2 được thực hiện, ngược lại biểu thức 3 được thực hiện.

Ví dụ:

```
int n ;
printf(“Nhap vao mot so bat ki ”);
scanf(“%d”,&n);
prinft(“so %d vua nhap la so ”, n);
(n%2==0)? printf(“chan”): printf(“le”);
```

4.3. Cấu trúc lệnh lặp

Lệnh lặp cho phép máy thực hiện một hành động nào đó lặp lại nhiều lần, mặc dù chúng ta chỉ viết một lần, ngôn ngữ C có các lệnh lặp sau đây.

4.3.1. Lệnh for

Cú pháp:

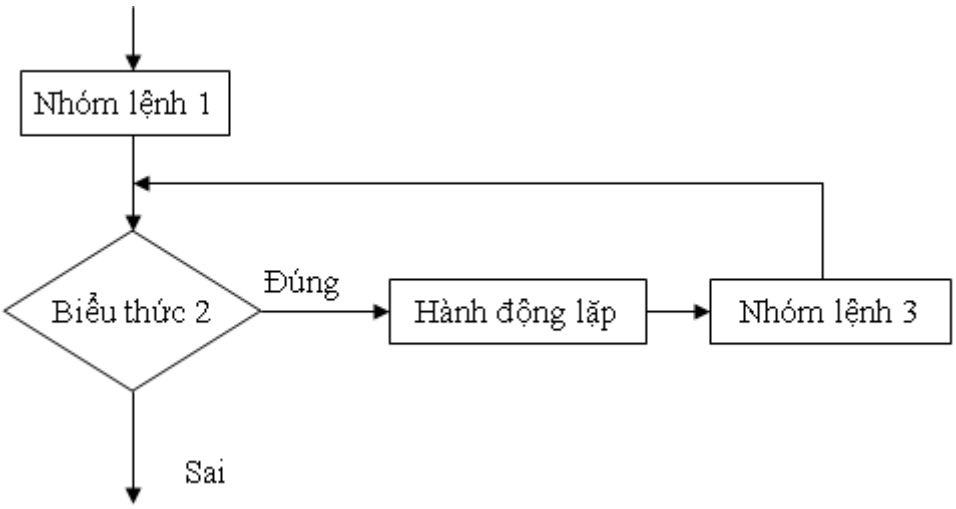
for (nhóm lệnh 1;biểu thức 2; nhóm lệnh3)

Hành động lặp;

Trong đó:

- Các nhóm lệnh 1,2,3 là các lệnh điều khiển trong quá trình lặp của for
- Hành động lặp là khối lệnh để máy tính thực hiện lặp lại nhiều lần.

Sơ đồ khối:



Chức năng:

Máy thực hiện nhóm lệnh 1 duy nhất một lần đầu tiên, sau đó thực hiện tính biểu thức 2 và nếu kết quả tương ứng với giá trị đúng thì máy thực hiện hành đông lặp, tiếp theo máy thực hiện nhóm lênh 3, và lặp lại kiểm tra biểu thức 2 cho đến khi kết quả tương ứng với giá trị sai.

Ví dụ:

```
for(i=1;i<5;i++)
printf("\n i= %d",i);
```

Sẽ hiện ra các số nguyên từ 1 đến 4 ra màn hình.

```
for (i = 4;i>=1;i--)
printf("\n  i = %d",i);
```

Sẽ hiện các số nguyên từ 4 xuống 1 ra màn hình.

Chú ý:

- Trong hành động lặp có thể có một lệnh hoặc nhiều lệnh, nếu có nhiều lệnh thì phải có cặp dấu mở đóng ngoặc nhọn {} để tạo khối
- Trong lệnh này chỉ có nhóm lệnh 1 được thực hiện một lần đầu tiên, còn lại các biểu thức 2 và nhóm lệnh 3 và hành động lặp, sẽ thực hiện nhiều lần trong quá trình lặp.
- Các nhóm lệnh 1, 3 và biểu thức 2 có thể không xuất hiện trong lênh for nhưng phải có dấu chấm phẩy (;) khi đó nếu không có biểu thức 2 thì máy sẽ xem như điều kiện lặp luôn luôn đúng và sẽ lặp vô tận.
- Trong mỗi nhóm lệnh 1 và 3 có thể có nhiều lệnh cách nhau bởi dấu phẩy đối với biểu thức 2 có thể thay đổi bằng một nhóm lệnh và khi đó máy sẽ lấy kết quả của lệnh cuối cùng làm điều kiện lặp hoặc dừng.

4.3.2. Lệnh while

Cú pháp

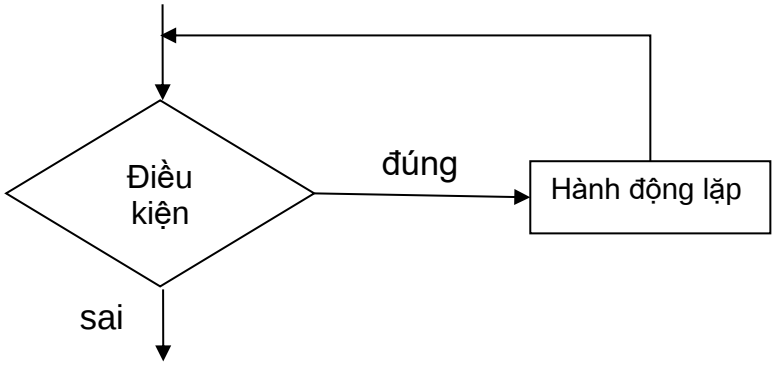
while (điều kiện lặp)

Hành động lặp;

Trong đó:

- Điều kiện lặp : là biểu thức so sánh hoặc biểu thức nguyên
- Hành động lặp: là các lệnh sẽ được thực hiện hiều lần trong quá trình lặp, nếu có nhiều lệnh phải có dấu { } để đóng khối.

Sơ đồ khối



Chức năng:

Trong khi điều kiện lặp vẫn có giá trị đúng thì máy sẽ lặp thực hiện khối lệnh, và sẽ dừng lại nếu điều kiện sai

Ví dụ:

```
x=1; y=10;
while(x<y)
{
     printf("\n x = %d va y = %d",x,y);
     x +=2;y--;
}
```

Chú ý:

Nếu hành động sai ngay từ đầu thì máy sẽ không thực hiện hành động lặp dù chỉ một lần.

4.3.3. Lệnh do – while

Không giống như các vòng lặp for và while, mà kiểm tra điều kiện vòng lặp ở ngay bước đầu tiên của vòng lặp, vòng lặp do…while trong Ngôn ngữ C kiểm tra điều kiện của nó tại phần cuối của vòng lặp.

Một vòng lặp do...while là tương tự như vòng lặp while, ngoại trừ ở điểm một vòng lặp do…while bảo đảm thực hiện vòng lặp ít nhất một lần.

Cú pháp:

```
do
{
    cac_lenh;
}while( dieu_kien );
```

Nếu điều kiện là true, các lệnh trong vòng lặp được thực hiện lần nữa. Tiến trình này lặp đi lặp lại tới khi nào điều kiện đã cho trở thành false.

Ví dụ:

```
int a = 10;
/* vong lap do...while */
do
{
   printf("Gia tri cua a la: %d\n", a);
   a = a + 1;
}while( a < 15 );
```

4.4. Lệnh breack, continue

4.4.1. Lệnh Break

Lệnh **break** có tác dụng dừng vòng lặp (for ,while, do-while) trong mọi trường hợp.

4.4.2. Lệnh continue

Lệnh ***continue*** làm cho chương trình bỏ qua phần còn lại của vòng lặp và nhảy sang lần lặp tiếp theo. Những câu lệnh viết sau lệnh **continue** sẽ không được thực hiện.

4.5. Bài tập thực hành

Bài tập 1:

Viết chương trình nhập vào từ bàn phím 2 số nguyên. Tính tổng hai số đó, hiện kết quả ra màn hình.

```
#include<stdio.h>
#include<conio.h>
Void main()
{
```

```
    int a,b;
    clrscr();
    printf("nhap so a =");
    scanf("%d",&a);
    printf("nhap so b =");
    scanf("%d",&b);
    printf("tong hai so %d va %d la: %d",a,b,a+b);
    getch();
}
```

Bài tập 2 Nhập x từ bàn phím, tính f(x), đưa kết quả ra màn hình.

$$f(x) = \sqrt[3]{x+3} + tg(x)$$

Nội dung chương trình

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
Void main()
   {
     int x;
     clrscr();
     printf("nhap so x =");
     scanf("%d",&x);
      printf("gia tri bieu thuc f(x) =  %f
        ",pow((x+3),1.0/3)+sin(x)/cos(x));
     getch();
   }
```

Bài tập 3

Lập chương trình tìm các số có 3 chữ số sao cho số đó bằng tổng lập phương các chữ số của nó

```
#include<conio.h>
#include<stdio.h>
#include<math.h>
void main()
{
```

```
 int i =100, tram,chuc,donvi;
 clrscr();
 printf("cac so tim duoc la:\n");
 do{
    tram=i/100;
    chuc=(i%100)/10;
    donvi=(i%100)%10;
    if(i==(pow(tram,3)+pow(chuc,3)+pow(donvi,3)))
    printf("  %d",i);
    i++;
    }while(i<1000);
  getch();
}
```

Chương 5: CHƯƠNG TRÌNH CON

5.1. Khái niệm

Một chương trình viết trong ngôn ngữ C là một dãy các hàm, trong đó có một hàm chính (hàm main). Hàm chia các bài toán lớn thành các công việc nhỏ, có thể có những đoạn chương trình viết lặp đi lặp lại nhiều lần, để tránh rườm rà và mất thời gian khi viết chương trình; người ta thường phân chia chương trình thành nhiều module, mỗi module giải quyết một công việc vào đó. Thứ tự các hàm trong chương trình là bất kỳ, song chương trình bao giờ cũng đi thực hiện từ main().

Trong C, chương trình con được gọi là hàm. Hàm trong C có thể trả về kết quả thông quan tên hàm hay có thể không trả về kết quả.

Hàm có hai loại: Hàm chuẩn và hàm tự định nghĩa. Trong chương này ta chú trọng đến cách định nghĩa hàm và cách sử dụng các hàm đó.

Hàm thư viện

Hàm thư viện là những hàm đã được định nghĩa sẵn trong một thư viện nào đó, muốn sử dụng các hàm thư viện thì phải khai báo thư viện trước khi sử dụng bằng lệnh *#inlcude <tên_thư_viện.h>*

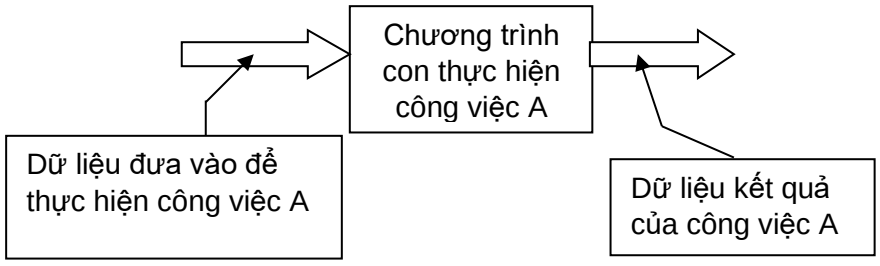
Hàm người dùng

Hàm người dùng là những hàm do người lập trình tự tạo ra nhằm đáp ứng nhu cầu xử lý của mình.

Một hàm khi được định nghĩa thì có thể sử dụng bất cứ đâu trong chương trình. Trong C, một chương trình bắt đầu thực thi bằng hàm main.

Chương trình con được dùng để tối ưu hóa việc tổ chức chương trình, chia một chương trình lớn thành nhiều công việc độc lập nhỏ. Dùng chương trình con thực hiện các công việc nhỏ, tạo thành mô-đun. Khi đó nhiệm vụ chương trình chính chỉ là cung cấp dữ liệu đầu vào cho các mô-đun để hoàn thành công việc của mình. Một chương trình viết theo cách này gọi là chương trình cấu trúc.

Có thể minh họa chương trình con như hình sau:



Hình ảnh minh họa nhiệm vụ của chương trình con

Để nhận các dữ liệu đưa vào cho chương trình con và nếu có thể chứa dữ liệu kết quả ra chúng ta phải sử dụng tham số (parameters) của chương trình con.

Tham số tồn tại dưới hai hình thức đó là tham số thực và tham số hình thức. Tham số hình thức là tham số để khai báo và xây dựng trương trình con, còn tham số thực để xác định dữ liệu đưa vào khi gọi chương trình con.

Trong một số ngôn ngữ lập trình cung cấp hai loại chương trình con riêng biệt đó là hàm(function) và thủ tục (procedures) nhưng trong C chỉ cung cấp một loại đó là hàm

Để tạo một hàm chúng ta cần xác định

- Hàm tạo ra sẽ thực hiện công việc gì ?
- Sau khi thực hiện song có cần trả về một dữ liệu hay nhiều dữ liệu không ?
- Để thực hiện được công việc đó ta cần những dữ liệu nào ?

5.2. Xây dựng hàm

5.2.1. Xây dựng hàm

➢ *Khai báo hàm*

Khai báo hàm được thực hiện ở phần đầu chương trình, khai báo để chỉ cho máy biết các thông tin về hàm bao gồm : kiểu dữ liệu trả về có hay không, tên hàm, các tham số bao gồm kiểu và tên của từng tham số.

Cú pháp khai báo như sau :

Tên_kiểu_trả_về tên_hàm(kiểu1 tham_số1, kiểu2 tham_số2 .....) ;

Trong đó :

- Tên kiểu trả về : là một tên kiểu dữ liệu quy định kết quả trả về là khiểu gì.
- Tên hàm : Tự đặt theo quy định đặt tên của ngôn ngữ C.
- Kiểu1 tham_số1 : xác định tên của tham số thứ nhất và kiểu của tham số đó,... Nếu có nhiều tham số, mỗi tham số phân cách nhau bởi dấu phẩy (,)

Ví dụ:

```
int sum(int a , int b) ;
float max(float x, float y);
```

Nếu trong hàm không có dữ liệu trả về thì viết kiểu trả về là **void** nếu không có tham số thì bỏ trống và phải có cặp dấu đóng mở ngoăc () sau tên hàm.

Ví dụ:

```
void  hien();
```

➢ *Xây dựng hàm*

Sau khi khai báo xong hàm chúng ta có thể viết lệnh thực hiện công việc đặt ra cho chương trình con.

Cú pháp:

Tên_kiểu_trả_về tên_hàm(kiểu1 tham_số1,kiểu2 tham_số2 .....)

{

Các câu lệnh thực hiện công việc đặt ra cho hàm.

}

Khi cần kết thúc thực hiện hàm và trả về dữ liệu nào đây, chúng ta viết lệnh ***return*** vào vị trí cần thiết nhất trong hàm.

Cú pháp:

return giá-trị-dữ-liệu-trả-về;

Ví dụ:

```
int sum(int a, int b)
{
    return   a+b;
}
```

Hoặc

```
float max(float x, float y)
{
    if(x>y)
    return x;
   else
    return y;
}
```

Đối với hàm có kiểu void có thể không cần lệnh **return**

Ví dụ:

```
voidhien()
{
    printf("ho va ten: Nguyen Van Nam");
    printf("Ngay sinh :12\08\2014");
    printf("que quan : Yen Bai");
}
```

Chú ý:

Thông thường với những chương trình đơn giản có ít chương trình con thì người ta viết lệnh cho hàm ngay tại nơi khai báo.

5.2.2. Sử dụng hàm

Một hàm khi định nghĩa thì chúng vẫn chưa được thực thi trừ khi ta có một lời gọi đến hàm đó. Khi thực hiện chương trình máy sẽ thực hiện các lệnh trong chương trình chính(hàm main), do đó để yêu cầu máy thực hiện công việc của chương trình con ta phải viết lệnh gọi chương trình con với cú pháp sau.

Cú pháp:

<tên_hàm>([Danh sách các tham số])

Lời gọi này có thể là một câu lệnh độc lập hoặc đặt trong biểu thức, nếu đặt trong biểu thức thì hàm đó phải có giá trị trả về để thực hiện tính toán biểu thức đó.

Ví dụ:

```
printf("Tong hai so 5 va 6 la: %d",tong(5,6));
```

Hoặc:

```
float x ;
x = max(a,b) ;
```

Hoặc: `hien();`

Chú ý:

Thông thường lời gọi hàm được viết trong chương trình chính, tuy nhiên có thể viết trong chương trình con khác. Các chương trình con có thể gọi lẫn nhau.

5.3. Các tham số của hàm

5.3.1. *Phân biệt các loại tham số*

Trong lập trình, tham số là biến được thu nhận bởi một chương trình con. Tại thời gian chạy, chương trình con sử dụng các giá trị được gán cho các tham số để thay đổi cách ứng xử của mình. Hầu hết các ngôn ngữ lập trình có thể định nghĩa các chương trình con không có tham số hoặc chấp nhận một vài tham số.

Tham số hình thức:

Là biến được liệt kê trong danh sách tham số (thường nằm tại phần đầu của định nghĩa chương trình con).

Tham số thực sự : Là giá trị cụ thể của biến đó tại thời gian chạy.

Để phân biệt rõ hai khái niệm trên, xét ví dụ dưới đây:

```
int sum(int gt1, int gt2)
{
  return (gt1+gt2);
}
```

Hàm **sum** nhận hai tham số hình thức: gt1và gt2. Nó lấy tổng của các giá trị được truyền vào các tham số này và trả về kết quả cho nơi gọi hàm (bằng cách sử dụng một kỹ thuật được cung cấp tự động bởi trình biên dịch C). Mã gọi hàm sum có thể trông như dưới đây:

```
int a=40;
int b=2;
int c =  sum(a,b);
```

Các biến **a** và **b** được khởi tạo với các giá trị 40 và 2. Các biến này không phải tham số hình thức hay tham số thực sự. Tại thời gian chạy, giá trị đã được gán cho các biến này được truyền vào cho hàm **sum** . Trong hàm **sum**, các tham số hình thức gt1 và gt2 được tính giá trị và lần lượt cho kết quả là hai tham số thực sự 40 và 2. Giá trị của các tham số thực sự được cộng lại, kết quả được trả về cho nơi gọi hàm - nơi nó được gán cho biến **c**

Tham số hình thức thường được gọi tắt là tham số. Tham số thực sự còn được gọi là tham số thực, tham đối hoặc đối số.

5.3.2. *Cách truyền tham số*

Có hai cách để truyền tham số cho hàm: Truyền theo *tham trị* và truyền theo *tham chiếu*.

Truyền bằng tham trị

Trong cơ chế truyền tham số bằng giá trị (gọi là truyền tham trị), khi chương trình con được chạy, một bản sao của tham số thực sự được gán cho tham số hình thức (sao chép giá trị). Nghĩa là mọi sửa đổi của chương trình con đối với tham số hình thức không gây ảnh hưởng tới biến được truyền vào chương trình con theo kiểu truyền tham trị.

Các tham số được truyền bằng giá trị được gọi là tham trị. Do chỉ có giá trị được truyền vào chương trình con, tham số thực sự không nhất thiết phải là một biến thông thường mà có thể là hằng giá trị, hằng biến, biểu thức trả về giá trị...

Truyền bằng biến

Trong cơ chế truyền tham số bằng biến (gọi là truyền tham biến), khi chương trình con được chạy, tham số hình thức trở thành một tham chiếu tham số thực sự. Nghĩa là mọi sửa đổi của chương trình con đối với tham số hình thức sẽ có tác dụng với tham số thực sự. Đây được gọi là hiệu ứng phụ của chương trình con.

Các tham số được truyền bằng biến được gọi là tham biến. Ngược lại với cơ chế truyền bằng giá trị, cơ chế truyền bằng biến đòi hỏi tham số thực sự phải là một biến.

- *Biến toàn cục và biến cục bộ*

5.4. Bài tập thực hành

Bài tập 1:

Viết chương trình con tìm Max, Min 3 số nguyên, chương trình chính nhập vào 3 số nguyên, hiện số lớn nhất và nhỏ nhất ra màn hình.

```
#include<conio.h>
#include<stdio.h>
int max(int a,int b,int c)
{     int max=a;
      if(max<b)max=b;
      if(max<c)max=c;
      return max;
}
int min(int a, int b, int c)
{
      int min=a;
      if(min>b)min=b;
      if(min>c)min=c;
      return min;
}
void main()
{
      int a,b,c;
      clrscr();
      printf("\nnhap vao 3 so nguyen:");
      printf("\nso thu nhat: ");scanf("%d",&a);
      printf("\nso thu hai: ");scanf("%d",&b);
      printf("\nso thu ba : ");scanf("%d",&c);
      printf("\n so lon nhat la: %d",max(a,b,c));
      printf("\n so nho nhat la: %d",min(a,b,c));
```

```
        getch() ;
}
```

Chương 6: MẢNG

6.1. Khái niệm mảng

Mảng là cấu trúc dữ liệu cho phép quản lý một danh sách hữu hạn các dữ liệu cùng kiểu.

Ví dụ: Để chứa 5 dữ liệu số nguyên sau 10, 12, 25, 30, 40

Chúng ta có thể sử dụng 5 biến nhớ khác nhau

a = 10, b = 12, c = 25, d = 30, e = 40

Tuy nhiên nếu số lượng dữ liệu tăng thì cách này không phù hợp, và chúng ta sẽ sử dụng cấu trúc mảng, minh hoạ như sau:

10	12	25	30	40	....

Các dữ liệu trong mảng phải cùng kiểu (số nguyên, số thực, ký tự ...).

Mảng để chứa dữ liệu như trên là mảng 1 chiều, chúng ta có thể sử dụng mảng 2 chiều hoặc mảng nhiều chiều.

Ví dụ về mảng 2 chiều:

10	2	5	....
45	23	56	....
8	5	20	....
....	....	....	....

Mảng hai chiều là một bảng các dữ liệu được xếp theo hàng và cột, mỗi dữ liệu sẽ chứa trong ô được xác định bằng cách chỉ ra ở hàng nào và cột nào.

6.2. Khai báo mảng

Mảng một chiều

Mảng một chiều giống như một vector. Mỗi phần tử của mảng một chiều có giá trị *không phải là một mảng khác.*

– *Khai báo với số phần tử xác định*

Cú pháp: <Kiểu> <Tên_mảng> <[Số phần tử]>

Trong đó:

+ Tên_mảng: Đây là tên mảng đạt theo đúng quy tắc đặt tên của danh biểu. Tên này cũng mang ý nghĩa là tên biến mảng.

+ Số_phần_tử : Là một hằng số nguyên, cho biết số lượng phàn tử tối đa trong mảng là bao nhiêu (hay nói khác đi kích thước của mảng là gì).

+ Kiểu : Mỗi phần tử của mảng có dữ liệu thuộc kiểu gì.

Ở đây, ta khai báo một biến mảng gồm có Số_phần_tử phần tử, phần tử thứ nhất là tên_mảng [0], phần tử cuối cùng là tên_mảng[Số_phần_tử -1]

VD:

int a[10]; /*Khai báo biến mảng tên a, phần tử thứ nhất là a[0], phần tử cuối cùng là a[9].*/

Ta có thể coi mảng a là một dãy liên tiếp các phần tử trong bộ nhớ như sau:

Vị trí	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Tên phần	**a[0]**	**a[1]**	**a[2]**	**a[3]**	**a[4]**	**a[5]**	**a[6]**	**a[7]**	**a[8]**	**a[9]**

– *Khai báo với số phần tử không xác định (khai báo không tường minh)*

Cú pháp:

<Kiểu> <Tên_mảng> <[]>

Khi khai báo, không cho biết rõ số phần tử của mảng, kiểu khai báo này thường được áp dụng trong các trường hợp: vừa khai báo vừa gán giá trị, khai báo mảng là tham số hình thức của hàm.

– *Vừa khai báo vừa gán giá trị*

Cú pháp:

<Kiểu> <Tên_mảng> [] = {Các giá trị cách nhau bởi dấu phẩy}

Nếu vừa khai báo vừa gán giá trị thì mặc nhiên C sẽ hiểu số phần tử của mảng là số giá trị mà chúng ta gán cho mảng trong cặp dấu {}.

Ví dụ:

int S[] = {3,5,6};

6.3. Các thao tác trên mảng

6.3.1. Truy xuất mảng

Đối với mảng một chiều các phần tử chứa dữ liệu được sắp xếp liên tục trong bộ nhớ máy, và mỗi phần tử sẽ có một chỉ số (là số thứ tự) của phần tử đó trong mảng.

Phần tử đầu tiên có chỉ số là 0, tiếp theo là 1 và tiếp tục cho đến hết, vậy sẽ dừng lại ở chỉ số bằng số lượng phần tử -1.

Minh hoạ bằng hình sau:

10	25	15	30	.....
0	1	2	3	....

Kích thước của mảng (tính bằng byte) sẽ là: số lượng phần tử của mảng nhân với kích thước của kiểu dữ liệu của mảng đó.

Ví dụ:

int a[20];

Thì mảng sẽ chiếm kích thước bộ nhớ là: 20 x 2 = 40 byte

Để truy nhập đến các phần tử ta sử dụng tên và các chỉ số của phần tử đó theo cú pháp sau:

Tên_mảng[chỉ_số_phần_tử_cần_truy_nhập];

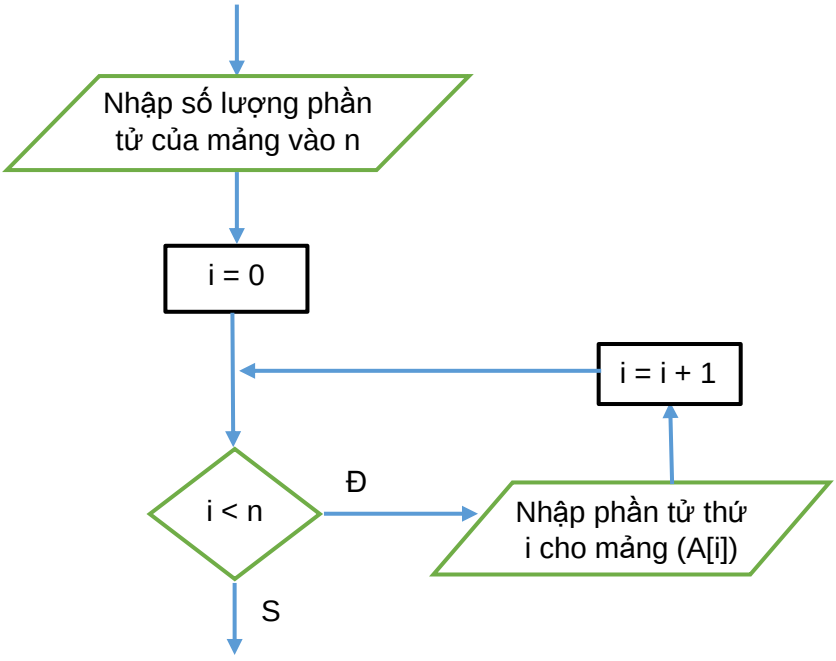
Ví dụ:

a[0]=10;

Truy xuất đến phần tử đầu tiên của mảng a rồi gán giá trị cho phần tử đó bằng 10.

6.3.2. Nhập dữ liệu vào mảng

Sơ đồ thuật toán nhập dữ liệu vào mảng



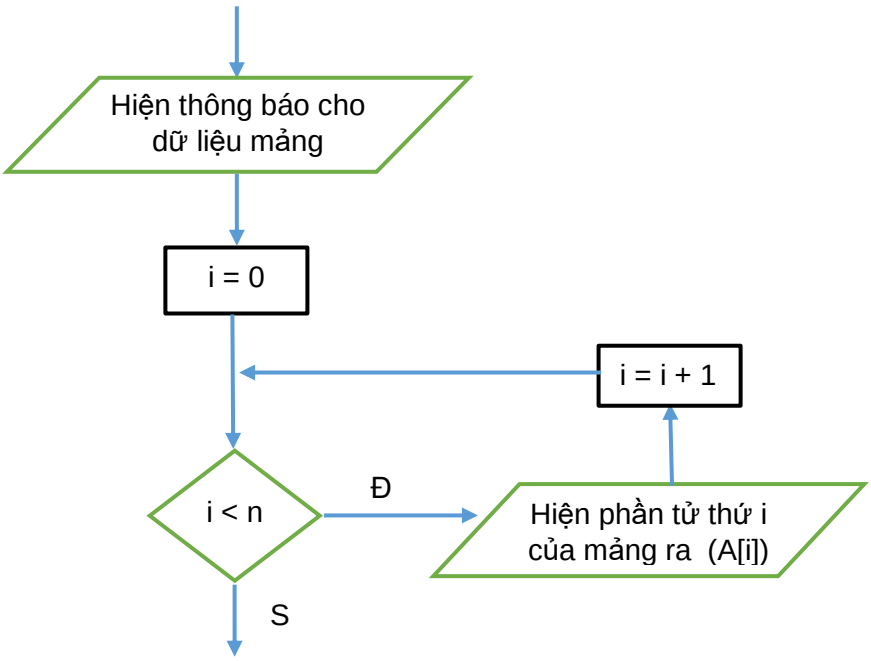
Thể hiện bằng chương trình

```
printf("nhap so luong phan tu cua mang n = ");
scanf("%d",&n);
for(i=0; i< n; i++)
{
    printf("Nhap phan tu thu: %d ", i+1);
    scanf("%d",&A[i]);
```

```
}
```

6.3.3. Hiển dữ liệu mảng ra màn hình

Sơ đồ thuật toán hiện dữ liệu từ mảng ra màn hình



- Thể thiện bằng chương trình

```
printf("Cac phan tu cua mang la: ");
for(i=0; i< n; i++)
{
    printf("%5d ", A[i]);
}
```

- Nếu muốn hiển thị dữ liệu các phần tử của mảng theo một điều kiện nào đó ta chỉ cần thêm điều kiện vào sau lệnh for

Ví dụ: Hiện các số lẻ của mảng

```
printf("Cac phan tu cua mang la: ");
for(i=0; i< n; i++)
if(A[i]%2==1)
{
    printf("%5d ", A[i]);
}
```

6.4. Sắp xếp các phần tử trên mảng

Một trong những thuật toán sắp xếp đơn giản nhất là phương pháp sắp xếp kiểu chọn. Ý tưởng cơ bản của cách sắp xếp này là:

- Ở lượt thứ nhất, ta chọn trong dãy khoá k[1..n] ra khoá nhỏ nhất (khoá ≤ mọi khoá khác) và đổi giá trị của nó với k[1], khi đó giá trị khoá k[1] trở thành giá trị khoá nhỏ nhất.
- Ở lượt thứ hai, ta chọn trong dãy khoá k[2..n] ra khoá nhỏ nhất và đổi giá trị của nó với k[2].

…

- Ở lượt thứ i, ta chọn trong dãy khoá k[i..n] ra khoá nhỏ nhất và đổi giá trị của nó với k[i].

…

- Làm tới lượt thứ n - 1, chọn trong hai khoá k[n-1], k[n] ra khoá nhỏ nhất và đổi giá trị của nó với k[n-1].

Xét ví dụ sau:

i	**Lượt** / **K_i**	**1**	**2**	**3**	**4**	**...**	**9**
1	42	**11**	11	11	11		11
2	23	23	**23**	23	23		23
3	74	74	74	**36**	36		36
4	11	**42**	42	42	**42**		**42**
5	65	65	65	65	65		58
6	58	58	58	58	58		65
7	94	94	94	94	94		74
8	36	36	36	**74**	74		87
9	99	99	99	99	99		94
10	87	87	87	87	87		99

Sau đây là giải thuật

/* cho dãy K gồm n phần tử, giải thuật này thực hiện sắp xếp các phần tử của K theo thứ tự tăng dần, dựa vào phép chọn phần tử nhỏ nhất trong mỗi lượt */

```
int i,j,tg,min //min dùng để lưu chỉ số, số nhỏ nhất trong mỗi lượt.
for ( i = 0 ; i< n; i++)
  {
     min = i;
     for (j = i+ 1; j<n; j++ )
       {
```

```
            if (K[j]<K[min])
              {
                min= j;
               }
           }
          // đổi giá trị
        tg = K[i];
        K[i]= K[min];
        K[min]= tg;
      }
```

6.5. Tìm kiếm trên mảng

Tìm kiếm theo phương pháp tuần tự

Tìm kiếm tuần tự là một phương pháp tìm kiếm rất đơn giản, lần lượt duyệt qua toàn bộ các bản ghi một cách tuần tự. Tại mỗi bước, khoá của bản ghi sẽ được so sánh với giá trị cần tìm. Quá trình tìm kiếm kết thúc khi đã tìm thấy bản ghi có khoá thoả mãn hoặc đã duyệt hết danh sách.

Hàm tìm kiếm tuần tự trên một mảng các số nguyên như sau:

```
int sequential_search(int *a, int x, int n)
 {
      int i;
      for (i=0; i<n ; i ++)
        {
          if (a[i] == X)
               return(i);
        }
      return(-1);
 }
```

Hàm tục này tiến hành duyệt từ đầu mảng. Nếu tại vị trí nào đó, giá trị phần tử bằng với giá trị cần tìm thì hàm trả về chỉ số tương ứng của phần tử trong mảng. Nếu không tìm thấy giá trị trong toàn bộ mảng thì hàm trả về giá trị -1.

6.6. Bài tập thực hành

Bài tập 1:

Viết chương trình nhập vào một mảng số nguyên A, tối đa 100 phần tử. Hiện các số vừa nhập, tính tổng, trung bình cộng, max, min của dãy số đó.

```
#include<conio.h>
```

```
#include<stdio.h>
void main()
{
   clrscr();
   int a[100],n ,i;
   printf("\n nhap so luong phan tu n= ");
   scanf("%d",&n);
   //nhap du lieu
   for(i=0;i<n;i++)
   {
     printf("\n Nhap so thu %d ",i+1);
     scanf("%d",&a[i]);
   }
  // hien du lieu
  printf("\n day so vua nhap la:  ");
  for(i=0; i<n; i++)
  {
      printf(" %d",a[i]);
  }
  //tinh tong
  int tong=0;
  for(i=0; i<n; i++)
  {
      tong=tong + a[i];
  }
  printf("\n tong cua day so do la: %d",tong);
  //tinh tb cong
  printf("\n Trung binh cong cua day do la:
          %f",tong/(n*1.0));
  // tim max
  int max= a[0];
  for(i=0; i<n; i++)
  {
```

```
      if(max< a[i])
      max= a[i];
  }
  printf("\n so lon nhat la: %d",max);
  //tim min
  int min = a[0];
  for(i=0; i<n; i++)
  {
     if(min >a[i])
     min = a[i];
  }
  printf("\n so nho nhat la: %d", min);
   getch();
}
```

Bài tập 2

Nhập vào từ bàn phím một dãy số nguyên. Tìm số lớn nhất, nhỏ nhất và vị trí của số đó trong mảng.

```
#include<conio.h>
#include<stdio.h>
void main()
{
   int i,n,A[100],max=0,min=0;
   clrscr();
   printf("nhap so luong phan tu cua day: ");
   scanf("%d",&n);
   //nhap gia tri cho day
   for(i=0;i<n;i++)
   {
     printf("\n Nhap A[%d] = ",i+1);
     scanf("%d",&A[i]);
   }
   //tim Max, Min
   for(i=1;i<=n;i++)
   {
     if(A[max]<A[i]) max=i;
     if(A[min]>A[i]) min=i;
```

```
    }
    printf("\nSo lon nhat trong day la: %d",A[max]);
    printf("\t Vi tri so do la: %d",max+1);
    printf("\nSo nho nhat trong day la: %d",A[min]);
    printf("\t Vi tri so do la: %d",min+1);
    getch();
}
```

Bài tập 3

Nhập và từ bàn phím một dãy số nguyên, xắp xếp dãy đó theo chiều tăng dần và giảm dần. Hiện lên màn hình: dãy số vừa nhập, dãy đã sắp xếp.

```
#include<conio.h>
#include<stdio.h>
void nhap(int n,int C[])
  {
     for(int i=0;i<n;i++)
       {
        printf("\n Nhap so thu %d  ",i+1);
        scanf("%d",&C[i]);
       }
  }
void hien(int n,int C[])
  {
      for(int i=0;i<n;i++)
      {
         printf("  %d",C[i]);
      }
  }
void tang(int n, int C[])
  {
    int tg;
     for(int i=0;i<n;i++)
    for(int j=0;j<n-1;j++)
      if(C[j] >C[j+1])
       {
        tg=C[j];
        C[j]=C[j+1];
        C[j+1]=tg;
       }
  }
void giam(int n, int C[])
  {
    int tg;
     for(int i=0;i<n;i++)
```

```
    for(int j=0;j<n-1;j++)
      if(C[j] <C[j+1])
       {
        tg=C[j];
        C[j]=C[j+1];
        C[j+1]=tg;
       }
  }
void main()
{
      clrscr();
      int A[100],n;
      printf("\nnhap so luong phan tu day : n= ");
      scanf("%d",&n);
      printf("\n Nhap gia tri cho day :");
      nhap(n,A);
      printf("\n Day vua nhap la:");
      hien(n,A);
      //hien day da xap xep tang dan
      printf("\n Day da xap xep tang dan la:");
      tang(n,A);
      hien(n,A);
      //hien day da xap xep giam dan
      printf("\n Day da xap xep giam dan:");
      giam(n,A);
      hien(n,A);
      getch() ;}
```