LỜI NÓI ĐẦU

Môn lập trình cơ sở (LTCS) rất cần thiết cho mọi sinh viên thuộc khối công nghệ gồm công nghệ thông tin, điện, điện tử, viễn thông v.v.

Nhờ có nền tảng từ LTCS, sinh viên thuộc khối công nghệ sẽ phát triển thành các hướng lập trình khác nhau để tạo ra các ứng dụng mạng máy tính, ứng dụng trên máy tính client/server, ứng dụng trên điện thoại hay các ứng dụng điều khiển thiết bị v.v

Môn LTCS sẽ sử dụng ngôn ngữ lập trình C (C) để hướng dẫn sinh viên nhập môn với lập trình. Lý do là vì C giúp sinh viên tiếp cận phương pháp lập trình hướng chức năng một cách cơ bản, rõ rang.. Tuy nhiên sau này sinh viên các chuyên ngành mạng, chuyên ngành công nghệ phần mềm, chuyên ngành điện tử v.v thường sử dụng các ngôn ngữ có nền tảng công nghệ mạnh như C#, Java, PHP v.v để phát triển ứng dụng hay giao thức v.v.

Bài giảng gồm 2 phần :

1. Trình bày quá trình giải quyết một bài toán trên máy tính.
2. Giới thiệu ngôn ngữ C dùng để giải quyết bài toán trên máy tính .

Với 2 phần trên được chia ra làm 6 bài:

1. Quá trình giải quyết bài toán trên máy tính
2. Giới thiệu các thành phần cơ bản của ngôn ngữ C
3. Các cấu trúc điều khiển của ngôn ngữ C
4. Hàm trong ngôn ngữ C
5. Kiểu dữ liệu có cấu trúc và con trỏ trong C
6. Kiểu dữ liệu tệp

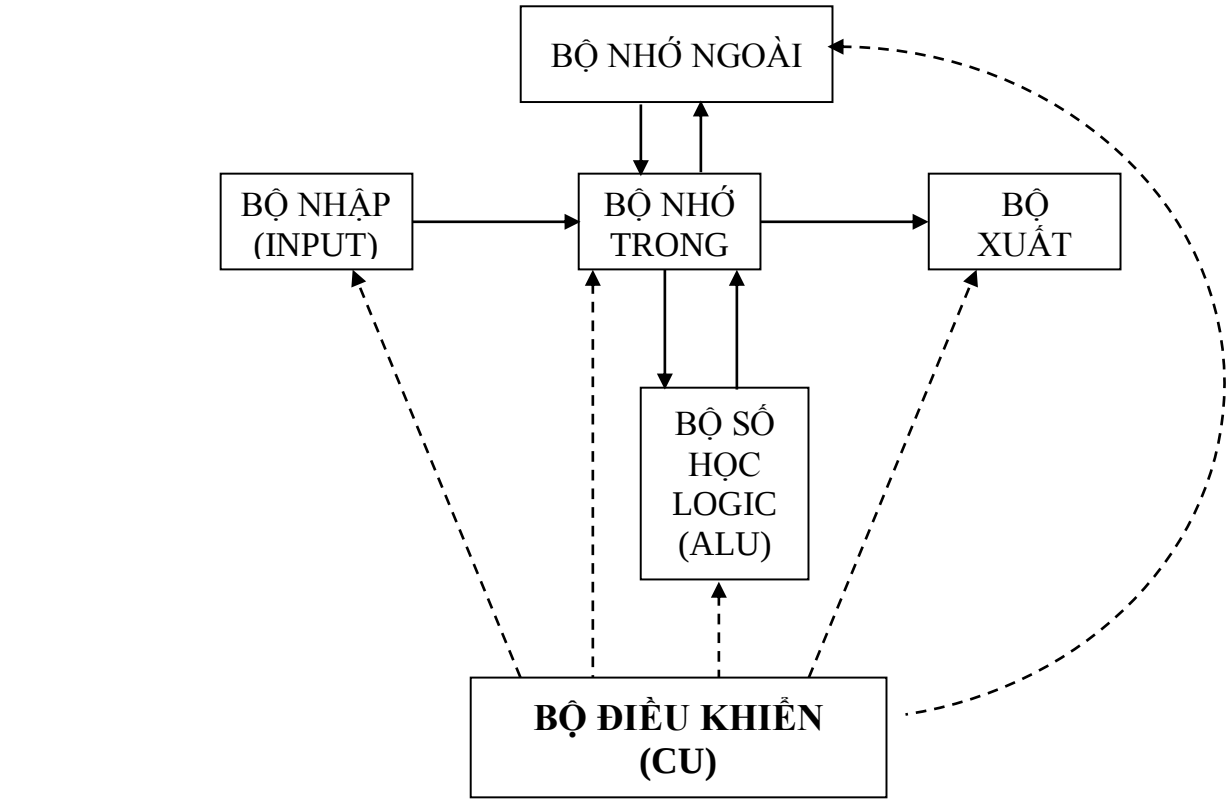
MỤC LỤC

BÀI 1: GIẢI QUYẾT BÀI TOÁN TRÊN MÁY TÍNH ĐIỆN TỬ

Bài này trình bày quá trình giải quyết một bài toán trên máy tính qua các bước cơ bản để giải quyết một bài toán, từ yêu cầu của bài toán đến phân tích, xây dựng và biểu diễn thuật toán…Bên cạnh đó, trong phần xây dựng thuật toán, nội dung bài cũng trình bày các phương pháp tìm kiếm thuật toán cũng như lượt qua các loại ngôn ngữ lập trình được sử dụng để ra lệnh cho máy tính. Quan trọng hơn với mỗi thuật toán cho 1 bài toán cụ thể, sinh viên biết vận dụng các ký hiệu hoặc ngôn ngữ tự nhiên để biểu diễn chúng.

1.1. MÁY TÍNH ĐIỆN TỬ

Mô hình của một MTĐT



Trong đó:

Bộ nhớ (Memory): Là thiết bị lưu trữ các thông tin cần cho máy giải bài toán, bộ nhớ máy tính thường chia thành các phần nhỏ gọi là các ô nhớ. Các ô nhớ này được đánh số từ 0, 1, 2, … gọi là địa chỉ của ô nhớ. Bộ nhớ máy tính thường được chia thành 2 khối lớn gọi là bộ nhớ trong (RAM, ROM)và bộ nhớ ngoài (đĩa cứng, đĩa mềm, đĩa CD...).

Bộ nhớ trong có dung lượng không lớn lắm nhưng tốc độ làm việc cao, là nơi chứa thông tin trực tiếp của bài toán đang giải.

Bộ nhớ ngoài có dung lượng rất lớn, chứa thông tin cần cho việc giải toán nhưng không trực tiếp tham gia giải bài toán. Khi cần thiết, các thông tin từ bộ nhớ ngoài sẽ đưa vào bộ nhớ trong để trực tiếp tham gia giải bài toán. Như vậy trong quá trình máy tính làm việc luôn có sự trao đổi thông tin giữa bộ nhớ trong và bộ nhớ ngoài dưới sự điều khiển của CU

Bộ nhập(Input):là thiết bị dùng để đưa thông tin vào bộ nhớ của MTĐT, bộ nhập phổ biến là bàn phím và một số thiết bị khác là chuột, máy scaner…

Bộ xuất(Output):là thiết bị đưa thông tin từ bộ nhớ của MTĐT ra cho người sử dụng. Bộ xuất phổ biến là màn hình và một số thiết bị khác như máy in, máy vẽ…

Bộ số học –lôgic(Arithmetic-Logical Unit):là thiết bị thực hiện tất cả các phép tính số học và phép tính lôgic để giải các bài toán

Bộ điều khiển (Control Unit):là thiết bị điều khiển toàn bộ hoạt động của MTĐT hoàn toàn tự động theo chương trình có sẵn trong bộ nhớ của máy. Bộ điều khiển thường được kết hợp với bộ số học lôgic để tạo thành khối xử lý trung tâm CPU (Centrol Processing Unit).

Thông tin được đưa từ bộ nhập vào bộ nhớ trong hoặc thông tin từ bộ nhớ ngoài đưa sang bộ nhớ trong. Từ bộ nhớ trong thông tin được đưa sang bộ số học –lôgic để tham gia thực hiện phép toán . Kết quả thực hiện phép toán được ghi lại vào bộ nhớ trong. Từ bộ nhớ trong thông tin được đưa ra ngoài cho người sử dụng hoặc đưa sang bộ nhớ ngoài để lưu trữ. Bộ điều khiển trung tâm sẽ điều khiển toàn bộ các thiết bị này tạo nên MTĐT.

1.2. CHƯƠNG TRÌNH MÁY TÍNH

Chương trình máy tính là dãy các lệnh mà MTĐT hiểu được và có thể thực hiện theo một thứ tự xác định để giải một bài toán.

Các loại chương trình

- Chương trình nguồn:là chương trình viết trong một ngôn ngữ nào đấy. Bản thân MTĐT không hiểu được và không thực hiện được chương trình nguồn. chương trình này gọi là chương trình gốc.
- Chương trình đích:là chương trình bằng ngôn ngữ tương đương với chương trình nguồn. Chương trình đích là kết quả của công việc dịch chương trình nguồn ra ngôn ngữ máy. Việc dịch chương trình nguồn sang chương trình đích là nhiệm vụ của chương trình dịch (compiler) của ngôn ngữ lập trình.
    - Có 2 loại chương trình dịch, đó là trình biên dịch và trình thông dịch.
    - Trình biên dịch: Là dịch toàn bộ chương trình nguồn sang chương trình đích, rồi sau đó MTĐT mới thực hiện chương trình đích.
    - Trình thông dịch:dịch lần lượt từng lệnh của chương trình nguồn và MTĐT tiến hành thực hiện luôn câu lệnh vừa dịch đó, cho tới khi thực hiện xong toàn bộ chương trình.
- Chương trình khả thi:Là chương trình thu được do liên kết chương trình đích với các mã tương ứng trong thư viện các hàm. Thông thường các ngôn ngữ lập trình có chứa một thư viện các hàm đã được định nghĩa và đã được dịch sang mã máy. Chương trình liên

kết (link) của ngôn ngữ sẽ liên kết tệp đích với các mã tương ứng trong thư viện hàm. Kết quả của bước này tạo ra một tệp khả thi (tệp có thể thực hiện được) là tệp thường có dạng "*.EXE". MTĐT có thể hiểu và thực hiện chương trình khả thi để giải bài toán.

1.3. CÁC BƯỚC CƠ BẢN KHI GIẢI QUYẾT BÀI TOÁN TRÊN MTĐT

1.3.1. Xác Định Bài Toán

Đầu vào Xử lý Đầu ra

- Việc xác định bài toán là xem thử ta giải quyết vấn đề gì?. với dữ liệu đầu vào thì ta cần tìm lời giải như thế nào để đạt được kết quả theo yêu cầu của đầu ra. Trong thực tế kết quả trả ra có thể là không hoàn toàn chính xác nhưng có thể đáp ứng được yêu cầu ở mức chấp nhận được, do quá trình giải quyết mất quá nhiều thời gian hoặc chi phí quá cao hoặc không thể tính chính xác.
- Xác định đúng yêu cầu bài toán là rất quan trọng bởi nó ảnh hưởng tới cách thức giải quyết bài toán và kết quả của lời giải. Một bài toán thực tế thường cho bởi những thông tin khá mơ hồ và hình thức, cần phải phát biểu lại một cách chính xác, chặt chẻ để hiểu đúng bài toán

Ví dụ:

Bài toán: Một dự án có n người tham gia thảo luận, họ muốn chia thành các nhóm và mỗi nhóm thảo luận riêng một phần của dự án. Nhóm có bao nhiêu người thì được trình lên bấy nhiêu ý kiến. Nếu lấy mỗi nhóm 1 ý kiến ghép lại thì được 1 bộ ý kiến triển khai dự án. Hãy tìm cách chia để số bộ ý kiến cuối cùng thu được là lớn nhất

Phát biểu lại:cho số nguyên dương n, tìm cách phân tích n thành tổng của các số nguyên sao cho tích của các số này là lớn nhất.

1.3.2. Tìm Cấu Trúc Dữ Liệu Để Biểu Diễn Bài Toán

Trước khi giải bài toán ta cần phải định nghĩa tập hợp dữ liệu để biểu diễn bài toán trong quá trình giải quyết bài toán. Tuỳ thuộc dữ liệu đầu vào và cách thức giải quyết bài toán mà ta chọn cấu trúc dữ liệu cho thích hợp.

Các tiêu chuẩn lựa chọn cấu trúc dữ liệu:

- Cấu trúc dữ liệu phải biểu diễn đầy đủ các thông tin nhập xuất của bài toán
- Cấu trúc dữ liệu phải phù hợp với các thao tác của thuật toán đã được chọn để giải quyết bài toán
- Cấu trúc dữ liệu phải cài đặt được trên máy tính với ngôn ngữ lập trình đang sử dụng

Ví dụ: để giải quyết các bài toán trên ma trận chúng ta dùng lưu trữ kế tiếp thông qua mảng 2 chiều.

1.3.3. Tìm Thuật Toán

1.3.3.1. Khái niệm

Thuật toán là một hệ thống chặt chẽ và rõ ràng các quy tắc nhằm xác định dãy các thao tác trên cấu trúc dữ liệu đã được chọn sao cho: với bộ dữ liệu đầu vào, sau một số hữu hạn bước thực hiện các thao tác đã chỉ ra thì ta đạt được đầu ra như đã định.

Các đặc trưng của thuật toán:

- Tính đơn định:Ở mỗi bước của thuật toán, các thao tác phải hết sức rõ ràng, không gây nên sự nhập nhằng, lộn xộn, tuỳ tiện, đa nghĩa. Khi thực hiện đúng các bước của thuật toán thì với một dữ liệu vào nhất định thì chỉ cho duy nhất một kết quả ra.
- Tính dừng: Thuật toán không được rơi vào quá trình vô hạn, phải dừng lại và cho kết quả sau một số hữu hạn các bước với thời gian chấp nhận được.
- Tính đúng: Sau khi thực hiện tất cả các bước của thuật toán theo đúng quá trình đã định, ta phải được kết quả mong muốn với mọi bộ dữ liệu đầu vào
- Tính phổ dụng:thuật toán phải dễ sửa đổi để thích ứng được với bất kỳ bài toán nào trong lớp các bài toán tương ứng và có thể làm việc trên các dữ liệu khác nhau.
- Tính khả thi:Kích thước thuật toán phải đủ nhỏ, thuật toán phải được máy tính thực hiện trong thời gian cho phép, thuật toán phải dễ hiểu và dễ cài đặt.

1.3.3.2. Các phương pháp xây dựng thuật toán

Trong phương pháp này có 3 loại:

1) Biểu diễn cho các bài toán có lời giải chính xác bằng 1 công thức toán học nào đó.(ví dụ:Tính tổng n số tự nhiên đầu tiên = n*(n+1)/2, giải phương trình bậc 2…)
2) Loại 2:Biểu diễn cho các bài toán bằng các công thức tính gần đúng (ví dụ: ex, sin(x)…)
3) Loại 3: Biểu diễn bài toán có các lời giải không tường minh bằng kỹ thuật đệ quy (ví dụ: n!=n*(n-1)!...).

- +Phương pháp gián tiếp: Phương pháp này được sử dụng khi chưa tìm ra lời giải chính xác của vấn đề, nó đưa ra những giải pháp mang tính đặc trưng của máy tính đó là dựa vào sức mạnh tính toán của máy tính. Có 3 phương pháp gián tiếp: Phương pháp thử sai, phương pháp Heuristic và phương pháp trí tuệ nhân tạo.
- Phương pháp thử sai:Ta có thể hình dung phương pháp này như việc "tìm cây kim trong đống rơm bằng cách rút từng sợi rơm ra cho đến khi tìm được cây kim". Trong phương pháp này dựa trên 3 nguyên lý, đó là:
- Vét cạn toàn bộ: Liệt kê tất cả trường hợp có thể xãy ra, không bỏ sót khả năng nào. Sau đó lần lượt kiểm tra từng trường hợp xem đó có phải là lời giải không
- Nguyên lý ngẫu nhiên: Thử một số khả năng được chọn một cách ngẫu nhiên trong tập các khả năng. Với một số điều kiện cụ thể của bài toán việc áp dụng nguyên lý ngẫu nhiên đã giúp đưa ra được những lời giải rất tốt đôi lúc có thể là tối ưu.

- Nguyên lý mê cung: Phương pháp này được áp dụng khi không biết được hình dạng chính xác của lời giải mà phải xây dựng dần lời giải qua từng bước giống như tìm đường đi trong mê cung.
- Phương pháp Heuristic: có nhiều bài toán nếu dùng phương pháp thử sai thì số phép thử quá lớn, thời gian để thực hiện xong thuật toán là quá lâu không chấp nhận được. Phương pháp Heuristic đơn giản gần gủi với suy nghĩ của con người, thuật giải này dựa trên các nguyên lý đơn giản như: Vét cạn thông minh, tối ưu cục bộ, nguyên lý hướng đích…
- Phương pháp trí tuệ nhân tạo: Phương pháp này dựa trên trí thông minh của máy tính. Trong phương pháp này, người ta đưa vào máy trí thông minh nhân tạo giúp máy tính bắt chước một phần khả năng suy luận như con người. Từ đó, khi gặp một vấn đề cần giải quyết, máy tính dựa trên những điều nó đã được học để tự đưa ra phương pháp giải quyết vấn đề.

1.3.3.3. Diễn đạt thuật toán

a) Dùng ngôn ngữ tự nhiên:

Sử dụng ngôn ngữ hàng ngày để liệt kê các bước của thuật toán. Phương pháp này không yêu cầu người viết lẫn người đọc thuật toán nắm vững các quy tắc. Khi mô tả thuật toán bằng ngôn ngữ tự nhiên, ta không cần phải quá chi tiết các bước và tiến trình thực hiện mà chỉ cần mô tả một cách hình thức đủ để chuyển thành ngôn ngữ lập trình .

Ví dụ: Tìm ước số chung lớn nhất của 2 số nguyên dương

Đầu vào: hai số nguyên dương a và b

Đầu ra: ước số chung lớn nhất của a và b

Bước 1:Nhập hai số dương a và b

Bước 2:Nếu a≠b thì chuyển sang bước 3, nếu không thì bỏ bước 3 chuyển bước 4

Bước 3: Nếu a>b thì thay a bằng a-b, nếu không thì thay b bằng b-a.Quay lại bước 2

Bước 4: Kết luận ước số chung lớn nhất là giá trị a

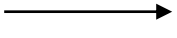
Kết thúc thuật toán.

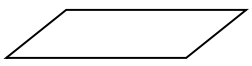
b) Lưu đồ thuật toán:

Lưu đồ là một công cụ trực quan để biểu diễn thuật toán, nó giúp cho người đọc theo dõi được sự phân cấp của các trường hợp và quá trình xử lý thuật toán.

Các ký hiệu sử dụng trong lưu đồ:

Bắt đầu hoặc kết thúc 

Hướng đi của thuật toán 

- Nhập xuất dữ liệu 

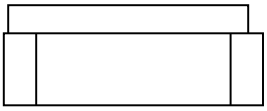
Với nhập dự liệu ta kèm theo Input bên cạnh, còn Output cho xuất dữ liệu hoặc dữ liệu đầu ra

Thao tác chọn lựa 

Biểu thức điều kiện chỉ cho 1 trong 2 giá trị là đúng hoặc sai, do đó sẽ có tới 2 hướng đi ra từ hình thoi này, một cho biểu thức điều kiện có kết quả là đúng, một cho biểu thức có kết quả là sai.

Thực hiện công việc

(thường là phép gán)

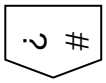


Gọi hàm

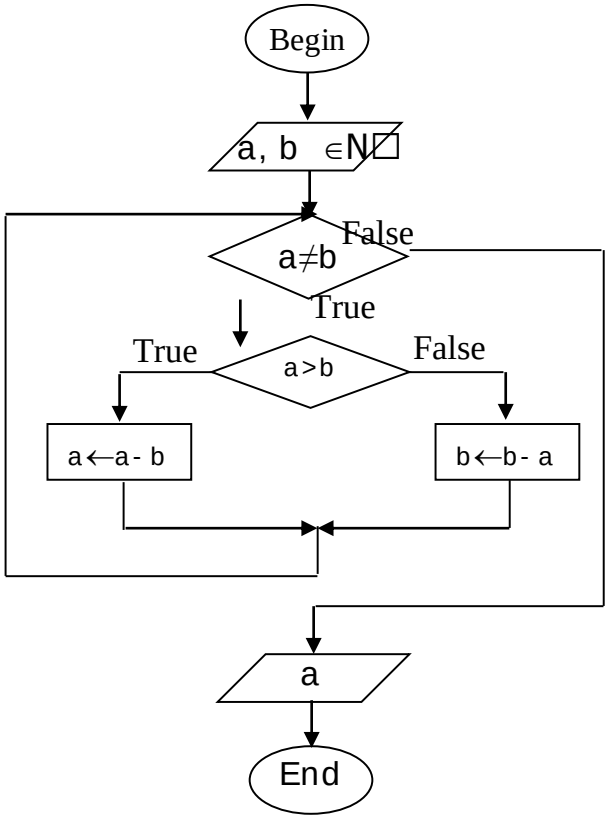
(được dùng để gọi hàm nhằm thực hiện một chức năng nào đó)

Điểm nối 

Được dùng để nối các thành phần khác nhau của lưu đồ lại với nhau. Bên trong điểm nối có đặt một ký hiệu là một giá trị nguyên đề biết sự liên kết giữa các điểm nối

Điểm nối sang trang 

Ví dụ: lưu đồ thuật toán:



c) Dùng mã giả:

Tuy sơ đồ khối thể hiện đươc quá trình xử lý và sự phân cấp các trường hợp của thuật toán nhưng lại rất cồng kềng. Hơn nữa việc biểu diễn vòng lặp trong sơ đồ khối là tương đối khó khăn đối với những sơ đồ có nhiều trang. Vì vậy chúng ta có thể sử dụng mã giả để biểu diễn thuật toán.

Trong mã giả ta sử dụng cú pháp của ngôn ngữ lập trình cụ thể nào đó kết hợp với ngôn ngữ tự nhiên để diễn đạt thuật toán

Ví dụ: Tìm ước số chung lớn nhất của 2 số nguyên dương

Bước 1: Read(a,b) với a,b là số nguyên dương

Bước 2: WHILE a≠b DO

IF a>b THEN

a □ a-b

ELSE

b □ b-a

Bước 3: Kết quả là a

1.3.4. Lập Trình

Sau khi có thuật toán, ta tiến hành thể hiện thuật toán đó trên máy tính qua ngôn ngữ lập trình:

- Ban đầu chương trình được thể hiện bằng ngôn ngữ tự nhiên, thể hiện thuật toán với các bước tổng thể, mỗi bước nêu lên mỗi công việc cần phải thực hiện
- Với một công việc đơn giản hoặc một thuật toán nhỏ đã được biết thì ta tiến hành viết mã lệnh ngay bằng ngôn ngữ lập trình. Còn đối với một công việc phức tạp thì ta lại chia thành những công việc nhỏ hơn để lại tiếp tục với công việc nhỏ hơn đó

1.3.5. Kiểm Thử

Kiểm thử chương trình gồm 2 công việc chính như sau:

- Chạy thử và tìm lỗi: Có 3 loại lỗi:
  - Lỗi cú pháp: Lỗi này hay gặp nhất nhưng lại dễ sửa nhất, chỉ cần nắm vững ngôn ngữ lập trình là đủ
  - Lỗi cài đặt:Việc cài đặt thể hiện không đúng thuật toán đã định, đối với lỗi này thì phải xem lại tổng thể chương trình, kết hợp với các chức năng gỡ rối của chương trình
  - Lỗi thuật toán: lỗi này gặp khi chúng ta xây dựng thuật toán sai. Lỗi này ít gặp nhưng nguy hiểm nhất, nếu nhẹ thì phải điều chỉnh lại thuật toán, nếu nặng thì có khi phải bỏ toàn bộ thuật toán và làm lại từ đầu.
- Xây dựng bộ test:

Bắt đầu với những test nhỏ, đơn giản, làm bằng tay có đáp số để so sánh với kết quả chương trình chạy ra

Tiếp theo cũng là những bộ test nhỏ nhưng chứa giá trị đặc biệt hoặc tầm thường

Cuối cùng là các bộ test lớn để kiểm tra tính chịu đựng của chương trình

1.3.6. Tối Ưu Chương Trình

Việc tối ưu chương trình dựa trên các tiêu chuẩn sau:

- Tính tin cậy:chương trình phải chạy đúng như dự định.
- Tính uyển chuyển:Chương trình dễ sửa đổi, nó sẽ làm bớt công sức của lập trình viên khi phát triển chương trình

- Tính trong sáng:chương trình phải dễ đọc dễ hiểu, để sau một thời gian dài khi đọc lại còn hiểu mình làm gì? Khi có điều kiện có thể sửa sai (do phát hiện lỗi), cải tiến hay biến đổi để chương trình giải quyết bài toán khác.
- Tính hữu hiệu:chương trình phải chạy nhanh ít tốn bộ nhớ tức là tiết kiệm được cả không gian và thời gian. Để được như vậy cần có giải thuật tốt và những tiểu xảo trong lập trình

1.4. SỰ PHÁT TRIỂN CỦA CÁC CÔNG CỤ LẬP TRÌNH

1.4.1. Lập Trình Bằng Ngôn Ngữ Máy

Mỗi MTĐT đều có một hệ lệnh riêng của mình đó chính là ngôn ngữ của MTĐT, đây chính là ngôn ngữ máy. Ngôn ngữ máy gồm các nhóm lệnh sau:

- Các lệnh nhập dữ liệu từ các thiết bị nhập khác nhau
- Các lệnh thực hiện các phép tính số học và lôgic
- Các lệnh điều khiển
- Các lệnh xuất dữ liệu cho các thiết bị khác nhau

Mỗi lệnh của ngôn ngữ máy gọi là một lệnh máy. Một lệnh máy là dãy các chữ số 0 và Thường một lệnh máy có 2 nhóm chữ số: một nhóm xác định mã phép toán cần thực hiện và một nhóm xác định địa chỉ ô nhớ chứa dữ liệu hoặc dữ liệu tham gia thực hiện phép toán. Phần nhóm địa chỉ có thể chứa một, hai hoặc ba địa chỉ tuỳ thuộc vào mã lệnh

Ví dụ : để đặt 3 vào ô nhớ nào đó 1011000000000011, hoặc cộng 1 vào ô nhớ nào đó 0000010000000001

Đặc điểm của lập trình bằng ngôn ngữ máy:

- Lập trình viên phải nhớ hệ lệnh của MTĐT là bảng dài các con số 0& 1 đơn điệu, dễ quên, khó nhớ
- Chương trình rất dài, con số vô hồn, khô khan, dễ nhầm
- Khó tìm và sửa chữa lỗi
- Khó tận dụng những chương trình đã có sẵn để viết chương trình mới.

1.4.2. Lập Trình Bằng Hợp Ngữ

Hợp ngữ ASSEMBLY là một loại ngôn ngữ giúp cho lập trình viên viết chương trình dễ dàng hơn là viết trong ngôn ngữ máy. Một lệnh của hợp ngữ thường tương đương với một lệnh của ngôn ngữ máy, nhưng hợp ngữ khác ngôn ngữ máy ở chổ là dùng mã lệnh dễ nhớ thay cho các con số vô hồn.

Ví dụ : để đặt 3 vào ô nhớ nào đó MOV AL, 3 hoặc cộng 1 vào ô nhớ nào đó ADD AL, 1

So với ngôn ngữ máy, lập trình hợp ngữ dễ dàng hơn vì:

- Mã phép toán được thay bằng các từ viết tắt có nghĩa nên dễ nhớ, dễ sử dụng đỡ nhầm lẫn

- Các địa chỉ ô nhớ được thay bằng các tên biến do người lập trình tự đặt, nên tạo ra sự thoải mái trong lập trình
- Lập trình viên dễ dàng tận dụng những chương trình đã có để phát triển thêm những chương trình mới

1.4.3. Lập Trình Bằng Ngôn Ngữ Lập Chương Trình

Ngôn ngữ lập chương trình, hay còn gọi là ngôn ngữ lập trình bậc cao, là loại ngôn ngữ do các nhà khoa học tạo ra dùng để viết các chương trình cho MTĐT.

Mỗi ngôn ngữ lập trình thường là hệ thống phức tạp gồm nhiều thành phần cơ bản là chương trình soạn thảo, chương trình dịch, chương trình liên kết, thư viện chương trình và các công cụ phát triển chương trình.

- Hệ soạn thảo của ngôn ngữ có chức năng như một hệ soạn thảo văn bản nhỏ gọn dùng để soạn thảo, sửa chữa và ghi lên đĩa chương trình nguồn.
- Chương trình dịch làm nhiệm vụ dịch chương trình nguồn sang chương trình đích
- Chương trình liên kết dùng để thực hiện việc liên kết chương trình đích với các mã tương ứng trong thư viện các hàm để tạo ra chương trình khả thi
- Thư viện chương trình là thành phần rất quan trọng và không thể thiếu được của ngôn ngữ lập trình. Đây là tệp chứa các hàm đã được định nghĩa sẵn trong ngôn ngữ.
- Công cụ phát triển là tập hợp các tệp dữ liệu và chương trình giúp cho việc viết, thử nghiệm và triển khai chương trình được thuận lợi hơn.

Đặc điểm của ngôn ngữ lập trình:

- Các câu lệnh của ngôn ngữ lập trình thường dùng các từ của ngôn ngữ tự nhiên, vì vậy chương trình như một văn bản dễ đọc, dễ hiểu.
- Một câu lệnh của ngôn ngữ lập trình tương đương với nhiều câu lệnh máy, vì vậy chương trình thường ngắn và gọn hơn.
- Việc tìm và sửa lỗi dễ dàng do chương trình dịch của ngôn ngữ lập trình sẽ phát hiện lối cú pháp và đưa ra thông báo
- Hệ soạn thảo là nơi lập trình viên soạn thảo chương trình gốc của mình hoặc sửa chữa những chương trình đã có để tạo ra một chương trình mới.

Một số ngôn ngữ lập trình bậc cao như: FORTRAN, ALGOL-60, COBOL, BASIC, PASCAL, C, C++, JAVA, PROLOG, SCHEME….

1.5. MỘT SỐ KỸ THUẬT LẬP TRÌNH

1.5.1. Lập Trình Thủ Tục

Lập trình thủ tục có những đặc điểm sau:

- Chương trình giải bài toán được chia thành các phần nhỏ hơn, tương đối độc lập với nhau gọi là các khối chương trình. Mỗi khối chương trình có chức năng riêng. Các khối

có thể được viết và thử nghiệm độc lập nhau theo tiêu chuẩn nhất định bởi một hoặc nhiều lập trình viên.

- Mỗi khối có một lối vào duy nhất ở đầu khối
- Mỗi khối có một lối ra duy nhất ở cuối khối
- Khi MTĐT thực hiện xong một khối nó sẽ chuyển đến khối khác hoặc kết thúc chương trình

1.5.2. Lập Trình Hướng Đối Tượng

Lập trình hướng đối tượng cho phép xây dựng chương trình từ các đối tượng. Trong đó, mỗi đối tượng có chứa dữ liệu để thể hiện tình trạng và được trang bị các hành vi để thực hiện một số công việc nhất định nhằm thông báo hoặc thay đổi dữ liệu của nó. Sự kết hợp giữa thuộc tính và hành vi của đối tượng được gọi là sự đóng gói.

Lập trình hướng đối tượng cho phép mô phỏng thế giới thực một cách tự nhiên bởi các đối tượng và mối quan hệ của giữa chúng, thuận tiện cho việc thiết kế hệ thống phức tạp. Bên cạnh đó người lập trình có thể thừa kế mã đã có sẵn để thực hiện một công việc khác dễ dàng nhằm tiết kiệm công sức và nâng cao năng suất lập trình.

1.5.3. Lập Trình Khai Báo

Lập trình khai báo bao gồm các phương pháp lập trình như: lập trình Lôgic, lập trình hàm. Trong lập trình hàm, lập trình viên định nghĩa các hàm toán học dễ suy luận và dễ hiểu mà không cần quan tâm đến chúng được cài như thế nào trong máy. Một đặc điểm quan trọng trong lập trình hàm là sử dụng hàm đệ quy.

Trong lập trình logic, Prolog là ngôn ngữ được sử dụng nhiều, nhất là trong lãnh vực trí tuệ nhân tạo. Một chương trình Prolog là một cơ sở dữ liệu gồm các mệnh đề. Mỗi mệnh đề được xây dựng từ các vị từ là các phát biểu về mối liên quan giữa các đối tượng chỉ có giá trị đúng hoặc sai. Trong Prolog một hệ thống Lôgic sẽ thực hiện chương trình theo cách suy luận dựa trên cơ sở dữ liệu đã có sẵn để chứng minh câu hỏi là một khẳng định đúng hay sai.

1.6. GIỚI THIỆU VỀ NGÔN NGỮ LẬP TRÌNH C

Ngôn ngữ lập trình C là một ngôn ngữ mệnh lệnh được phát triển từ đầu thập niên 1970 bởi Dennis Ritchie để dùng trong hệ điều hành UNIX. Từ đó, ngôn ngữ này đã lan rộng ra nhiều hệ điều hành khác và trở thành một những ngôn ngữ phổ dụng nhất. C là ngôn ngữ rất có hiệu quả và được ưa chuộng nhất để viết các phần mềm hệ thống, mặc dù nó cũng được dùng cho việc viết các ứng dụng. Ngoài ra, C cũng thường được dùng làm phương tiện giảng dạy trong khoa học máy tính mặc dù ngôn ngữ này không được thiết kế dành cho người nhập môn.

C là một ngôn ngữ lập trình tương đối nhỏ gọn vận hành gần với phần cứng và nó giống với ngôn ngữ Assembler hơn hầu hết các ngôn ngữ bậc cao. Hơn thế, C đôi khi được đánh giá như là "có khả năng di động", cho thấy sự khác nhau quan trọng giữa nó với ngôn ngữ bậc thấp như là Assembler, đó là việc mã C có thể được dịch và thi hành trong hầu hết các máy tính, hơn hẳn các ngôn ngữ hiện tại trong khi đó thì Assembler chỉ có thể chạy trong một số máy tính đặc biệt. Vì lý do này C được xem là ngôn ngữ bậc trung.

C đã được tạo ra với một mục tiêu là làm cho nó thuận tiện để viết các chương trình lớn với số lỗi ít hơn trong mẫu hình lập trình thủ tục mà lại không đặt gánh nặng lên vai người viết ratrình dịch C, là những người bề bộn với các đặc tả phức tạp của ngôn ngữ. Cuối cùng C có thêm những chức năng sau:

- Một ngôn ngữ cốt lõi đơn giản, với các chức năng quan trọng chẳng hạn như là những hàm hay việc xử lý tập tin sẽ được cung cấp bởi các bộ thư viện các thủ tục.
- Tập trung trên mẫu hình lập trình thủ tục, với các phương tiện lập trình theo kiểu cấu trúc.
- Một hệ thống kiểu đơn giản nhằm loại bỏ nhiều phép toán không có ý nghĩa thực dụng.
- Dùng ngôn ngữ tiền xử lý, tức là các câu lệnh tiền xử lý C, cho các nhiệm vụ như là định nghĩa các macro và hàm chứa nhiều tập tin mã nguồn (bằng cách dùng câu lệnh tiền xử lý dạng #include chẳng hạn).
- Mức thấp của ngôn ngữ cho phép dùng tới bộ nhớ máy tính qua việc sử dụng kiểu dữ liệupointer.
- Số lượng từ khóa rất nhỏ gọn.
- Các tham số được đưa vào các hàm bằng giá trị, không bằng địa chỉ.
- Hàm các con trỏ cho phép hình thành một nền tảng ban đầu cho tính đóng và tính đa hình.
- Hỗ trợ các bản ghi hay các kiểu dữ liệu kết hợp do người dùng từ khóa định nghĩa struct cho phép các dữ liệu liên hệ nhau có thể được tập hợp lại và được điều chỉnh như là toàn bộ.

Một số chức năng khác mà C không có (hay còn thiếu) nhưng có thể tìm thấy ở các ngôn ngữ khác bao gồm:

- An toàn kiểu,
- Tự động Thu dọn rác,
- Các lớp hay các đối tượng cùng với các ứng xử của chúng (xem thêm OOP),
- Các hàm lồng nhau,
- Lập trình tiêu bản hay Lập trình phổ dụng,
- Quá tải và Quá tải toán tử,
- Các hỗ trợ cho đa luồng, đa nhiệm và mạng.

Mặc dù C còn thiếu nhiều chức năng hữu ích nhưng lý do quan trọng để C được chấp nhận vì nó cho phép các trình dịch mới được tạo ra một cách nhanh chóng trên các nền tảng mới và vì nó cho phép người lập trình dễ kiểm soát được những gì mà chương trình (do họ viết) thực thi. Đây là điểm thường làm cho mã C chạy hiệu quả hơn các ngôn ngữ khác. Thường thì chỉ có ngôn ngữ ASM chỉnh bằng tay chạy nhanh hơn (ngôn ngữ C), bởi vì ASM kiểm soát được toàn bộ máy. Mặc dù vậy, với sự phát triển các trình dịch C, và với sự phức tạp của cácCPU hiện đại có tốc độ cao, C đã dần thu nhỏ khác biệt về tốc độ này.

Một lý do nữa cho việc C được sử dụng rộng rãi và hiệu quả là do các trình dịch, các thư viện và các phần mềm thông dịch của các ngôn ngữ bậc cao khác lại thường được tạo nên từ C.

1.7. BÀI TẬP

Với mỗi bài toán sau, hãy:

- Xác định đầu vào, đầu ra
- Cho một vài ví dụ mẫu
- Xây dựng thuật toán bằng ngôn ngữ tự nhiên
- Dùng lưu đồ để biểu diễn thuật toán

1) Giải phương trình bậc 1: Ax+B=0
2) Giải phương trình bậc 2: Ax2 +Bx+C=0
3) Giải bất phương trình bậc 2: Ax2 +Bx+C>0 (A≠0)
4) Giải hệ 2 phương trình 2 ẩn
5) Giải phương trình trùng phương: Ax4 +Bx2 +C=0
6) Tìm phương trình đường thẳng (dạng y=ax +b) qua 2 điểm A & B trong mặt phẳng Oxy
7) Tính n! (n nguyên dương)
8) Tính S=1+1/2+…+1/n (n nguyên dương)
9) Cho 4 điểm A, B, C & D trong mặt phẳng Oxy. Kiểm tra D□ □ABC
10) Liệt kê các số nguyên tố □n (n nguyên dương)
11) Liệt kê n số nguyên tố đầu tiên (n nguyên dương)
12) Tính ex bằng công thức gần đúng
13) Tính sin(x) bằng công thức gần đúng

BÀI 2: CÁC THÀNH PHẦN CƠ BẢN TRONG NGÔN NGỮ C/C++

Nội dung bài này giới thiệu những thành phần cơ bản của NNLTC đó là tập ký tự, từ khóa, tên… đồng thời cũng nêu lên cấu trúc chung và những quy tắc cơ bản khi viết chương trình C. Bên cạnh đó nội dung bài dành phần lớn nói đến các toán tử toán tử vào ra trong C, các kiểu dữ liệu cơ sở cũng như các phép toán căn bản trên nó.

Kết thúc chương, sinh viên sẽ cài đặt được một chương trình C hoàn chỉnh và biết cách thực hiện chương trình đó trên máy tính.

2.1. CÁC THÀNH PHẦN CƠ BẢN

2.1.1. Bộ Kí Tự Và Từ Khoá

a. Bộ kí tự:

26 chữ cái hoa : A, B, ..., Z

26 chữ cái thường : a, b, ...,c

Chữ số : 0,1,...,9

Các ký hiệu đặc biệt : . , ; : / ? [ ] { } ( ) ‘ “ ! # % & * + - = < > ...

Dấu cách (space bar)

Dấu gạch nối (hyphen) _

Chú ý: ngôn ngữ C phân biệt chữ IN và chữ thường

b. Các từ khoá:

Từ khoá là các từ dùng riêng cho C, mỗi từ có một ý nghĩa hoàn toàn xác định. Ngôn ngữ không cho phép định nghĩa lại các từ khoá của nó. Từ khoá dùng để khai báo các kiểu dữ liệu, viết các toán tử và viết các câu lệnh. Sau đây là một số từ khoá:

Asm	auto	break	case	char	const	continue
for	default	do	doubleelse	enum	extern	far
float	for	goto	huge	if	int	interrupt
long	near	pascal	register	return signed	short	sizeof
static	struct	switch	typedef	union	unsigned	void
volatile	while					

2.1.2. Tên Và Cách Đặt Tên

Tên (hay còn gọi là định danh) do người lập trình tự đặt, là thành phần quan trọng trong lập trình, được dùng để xác định các đại lượng khác nhau trong một chương trình như hằng, hàm, biến, cấu trúc, Macro. Tất cả các tên đều phải được khai báo trước khi sử dụng

Tên được đặt phải thoả 4 nguyên tắc sau:

- Dùng chữ cái, các con số và dấu gạch dưới (không ký tự trắng và ký tự đặc biệt)
- Không bắt đầu bằng số
- Không trùng với từ khoá

- Chỉ 32 ký tự đầu có nghĩa

Khi đặt tên phải thỏa 4 nguyên tắc trên

- Tên trong ngôn ngữ C phân biệt chữ IN và chữ thường
- Tên đặt cần phải gợi nhớ, có ý nghĩa, nói lên mục đích của nó

2.1.3. Lời Chú Thích

/* đoạn chú thích */

// dòng chú thích

Ví dụ:

cout<<"Hello"; //dòng lệnh này in ra màn hình chữ Hello

2.1.4. Cấu trúc chung của một chương trình C

Cấu trúc của chương trình C

```
#include <...>          //khai báo các thư viện chuẩn hoặc các tập tin có sẵn trong C
#define ...             //định nghĩa các Macro hoặc các hằng
const ...               //định nghĩa các hằng
struct ...              //định nghĩa các kiểu dữ liệu có cấu trúc
khai báo các biến toàn cục;
khai báo và định nghĩa các hàm của người sử dụng;
void/int main()         //hàm chính, mọi chương trình C đều thực hiện từ hàm main
{
    Định nghĩa Macro, kiểu cấu trúc, hằng ở đây;
    Khai báo các biến cục bộ trong hàm main;
    Các lệnh;
}
```

Ví dụ: tìm diện tích, chu vi hình thang cân

Giải: Giả sử đáy lớn là a, đáy nhỏ là b, đường cao là h thì:

diện tích = $\frac{a+b}{2}*h$, chu vi = $2*\left(\frac{ab}{2}+h+cạnh\right)$

```
#include<conio.h>
#include<iostream.h>
#include<math.h>
void main()
{
      float fDayLon, fDay_Nho;
      float fChieuCao, fDienTich, fChuVi, fCanhBen;
      clrscr();
      cout << "\nDien tich, chu vi hinh thang can:";
      cout << "\nNhap day lon>0:"; cin>>fDayLon;
      cout << "Nhap day nho>0:";cin>>fDayNho;
```

```
        cout << "Nhap chieu cao>0:";        cin>>fChieuCao;
        fCanhBen= sqrt(pow((fDayLon-fDayNho)/2,2)-pow(fChieuCao,2));
        fDienTich= (fDayLon+fDayNho)/2*fChieuCao;
        fChuVi= 2*fCanhBen+fDayLon+fDayNho;
        cout << " \nDien tich hinh thang =" << fDienTich ;
        cout << "\nChu vi hinh thang=" << fChuVi ;
        getch();
}
```

2.2. CÁC KIỂU DỮ LIỆU CƠ SỞ

2.2.1. Khái Niệm Dữ Liệu

Dữ liệu là những thông tin được lưu trữ trong các biến và được máy tính xử lý thông qua thuật toán. Nó chính là đầu vào của chương trình, được chương trình xử lý làm thay đổi biến thành đầu ra như yêu cầu đặt ra.

Dữ liệu được phân thành các loại như sau:

- Kiểu vô hướng (kiểu đơn giản)
- Kiểu cơ sở (kiểu số, kí tự)
- Kiểu do người lập trình định nghĩa (kiểu liệt kê - enumerated)
- Kiểu có cấu trúc (mảng, kiểu bản ghi, tệp).
- Kiểu con trỏ.

Một kiểu dữ liệu khi được định nghĩa thường có 3 đặc điểm chính như sau:

- Tập giá trị mà biến đó có thể nhận được.
- Các phép toán trên nó.
- Các hàm liên quan tác động trên nó.

2.2.2. Các Kiểu Dữ Liệu Cơ Sở

2.2.2.1. Kiểu ký tự (char)

Khai báo: char c;

Bảng mã của các ký tự biểu diễn được (mã Ascii từ 32..126) :

Ký tự	***Mã ASCII***
Các ký tự diều khiển (enter, esc, tab...)	0..31
Space bar (‘ ‘)	32
! ” # $ % & ’ ( ) * + , - . /	33..47
0...9	48..57
: ; < = > ? @	58..64
A...Z	65..90
[\ ] ^ _ ’	9 .96
a...z	97..122
{ | } ~	123..126

Các ký tự đồ họa 127..

Đoạn chương trình sau cho biết mã ASCII của các ký tự :

```
#include<conio.h>
#include<iostream.h>
void main()
{
	clrscr();
	int iDem;
	for (iDem=32; iDem<=255;iDem++) {
		cout.width(6); cout << iDem << ":" << (char)iDem;
	}
	getch();
}
```

Chú ý: Trong ngôn ngữ C, '5' biểu diễn ký tự 5, còn 5 biểu diễn số 5.

Các ký tự điều khiển không nhìn thấy được, được dùng điều khiển kết quả xuất ra màn hình. Một số ký tự điều khiển cần biết

Kí tự	Ký hiệu sử dụng	**Mã ASCII**
Đỗ chuông(BELL)	\a	7
Xoá trái (back space)	\b	8
Nhảy cách ngang	\t	9
Nhảy cách dọc	\v	11
Xuống dòng mới	\n	13
Về đầu dòng	\r	10
Mã NULL (không)	\0	0

Một số ký tự tham gia vào quá trình soạn thảo chương trình (",',\) thì sử dụng dấu \

Dấu " \"

Dấu ' \'

Dấu \ \\

2.2.2.2. Hằng dãy ký tự (chuỗi ký tự)

Hằng dãy ký tự là chuỗi ký tự được đặt trong cặp dấu nháy kép.

Ví dụ "CS 211" được lưu trữ trong bộ nhớ như sau:

C'	**S'**		**2'**	**1'**	**\0'**

Một số hăm chuẩn xử lý trín ký tự (trong tệp ctype.h):

TÊN HÀM	**Ý NGHĨA**

toupper(chr)	chuyển đổi một ký tự chr thành chữ hoa
tolower(chr)	chuyển đổi một ký tự chr thành chữ thường
isalpha(chr)	Hàm trả lại giá trị đúng (khác không) nếu chr là ký tự alphabe
islower(chr)	Hàm trả lại giá trị đúng (khác không) nếu chr là chữ cái thường
isupper(chr)	Hàm trả lại giá trị đúng (khác không) nếu chr là chữ cái hoa
isdigit(chr)	Hàm trả lại giá trị đúng (khác không) nếu chr là chữ số
iscntrl(chr)	Hàm trả lại giá trị đúng (khác không) nếu chr là ký tự điều khiển (mã từ 0 … 31)
isspace(chr)	Hàm trả lại giá trị đúng (khác không) nếu chr là dấu cách, dấu xuống dòng, dấu về đầu dòng, dấu tab
toascii(chr)	Cho mã ASCII của ký tự chr

2.2.2.3. Kiểu số nguyên

Có hai loại kiểu số nguyên cơ bản, đó là kiểu số nguyên có dấu và kiểu số nguyên không dấu. Bên cạnh đó số nguyên chia ra làm 2 loại là số nguyên và số nguyên dài.

Sau đây là một số dữ liệu kiểu số nguyên:

Từ khoá	***Số byte***	***miền giá trị***
int	2	-215 □ 215 - 1 □(-32768...32767)
short	2	-215 □ 215 - 1
long	4	-231 □ 231 - 1 □(-2.1 tỷ...2.1tỷ)
unsigned int	2	0 □ 216 - 1 □(0...65535)
unsigned short	2	0 □ 216 - 1
unsigned long	4	0 □ 232 - 1□(0...4.2tỷ)

2.2.2.4. Kiểu số thực

Trong ngôn ngữ C cho phép sử dụng 3 loại giá trị dấu phẩy động với các từ khoá float, double, long double.

Phạm vi biểu diễn từng loại được miêu tả qua bảng sau:

Từ khoá	***Số byte***	***Miền giá trị***	***Độ chính xác***
float	4	□ 2*10-38 □ □3.4*1038	7 chữ số

double	8	□2.2*10-308 □ □ 8*10308	15 chữ số
long double	10	□3.4*10-4932 □ □3.4*104932	19 chữ số

Có hai phương pháp biểu diễn số thực:

- Dạng bình thường: 3.14 2.5 ...
- Dạng khoa học: gồm 2 phần (phần định trị và phần mũ được viết sau chữ E)

Ví dụ: Số 3.14 được biểu diễn 0.314E + 01=314E-02

Các phép toán trên số thực: +, -, *, /

Một số hàm chuẩn trên kiểu dữ liệu số

- Chuyển đổi từ số thực sang số nguyên

  Hàm floor(x) □ Kết quả trả về là số nguyên lớn nhất mà □x (cắt phần thập phân).

  Hàm ceil(x) □ Kết quả trả về là số nguyên nhỏ nhất □x

  Ví dụ floor(13.2)=floor(13.8)=13 và ceil(13.2)=ceil(13.8)=14

- Một số hàm khác: với i là số nguyên, d là số thực (trong tệp math.h)

Hàm	**Kiểu giá trị nhận được**	**Tác dụng**
abs(i)	int	trả về giá trị tuyệt đối của i
fabs(d)	double	trả về giá trị tuyệt đối của d
sin(d)	double	trả về hàm sin
cos(d)	double	trả về hàm cos
tan(d)	double	trả về hàm tan
exp(d)	double	trả về hàm e^d
log(d)	double	trả về hàm ln(d)
pow(d1,d2)	double	trả về hàm $d1^{d2}$
fmod(d1,d2)	double	phần dư phép chia d1/d2
sqrt(d)	double	$\sqrt{d}$
random(i)	int	cho ngẫu nhiên giá trị nguyên trong 0..i-1

Chú ý: Các hàm sử dụng trên số thực cũng có thể thực hiện trên số nguyên

2.3. HẰNG, BIẾN, BIỂU THỨC VÀ PHÉP GÁN

2.3.1. Hằng

Hằng là đại lượng mà giá trị không thay đổi trong quá trình thực hiện chương trình

Một số biểu diễn về hằng được sử dụng trong ngôn ngữ C

- Hằng số thực:thêm F vào cuối giá trị (3.14f) hoặc ghi trực tiếp (3.14 hoặc 0.314E+01 hoặc3 4E-01). Đối với kiểu long double thêm L vào sau giá trị (ví dụ 0.543E3L)
- Hằng số nguyên:
  - Biểu diễn theo miền giá trị của số nguyên (int, long hoặc char)
  - Biểu diễn số nguyên dài:thêm L hoặc l vào cuối giá trị -4321L=-4321l=-4321 hoặc 65579L=65579l
  - Biểu diễn số nguyên trong hệ 8: thêm số 0 ở đầu giá trị 0357=3*82+5*81+7*80=239 (hệ 10)
  - Biểu diễn số nguyên trong hệ 16:thêm số 0x ở đầu giá trị 0x357=3*162+5*161+7*160=855(hệ 10) (trong đó A=10, B=11, C=12, D=13, E=14, F=15)
  - Biểu diễn số nguyên không dấu: thêm U vào cuối giá trị bình thường và thêm UL vào giá trị nguyên dài
- Hằng ký tự: được biểu diễn trong cặp dấu nháy đơn.ví dụ ‘5’ , ‘a’...

Cú pháp:

#define <tên_hằng> <giá_trị>

hoặc

const [kiểu_dữ_liệu] <tên_hằng> =<giá_trị>;

Chú ý: [kiểu_dữ_liệu] không có hiểu là kiểu int

ví dụ:

#define PI 3.14

hoặc

const float PI=3.14;

2.3.2. Biến

Biến là đại lượng có thể thay đổi được trong chương trình, mọi biến trước khi sử dụng đều phải được khai báo.

Tên biến theo nguyên tắc đặt tên

Khai báo biến: <kiểu_dữ_liệu> danh_sách_các_biến_cùng_kiểu;

Chú ý: Các biến cách nhau bởi dấu ‘,’ trong danh_sách_các_biến_cùng_kiểu

Ví dụ: khai báo các biến trong giải phương trình bậc 2: AX2+BX+C=0 như sau:

int iHeSoA, iHeSoB, iHeSoC, iDelta;

float fNghiemX1, fNghiemX2;

Trong trường hợp khai báo biến để lưu trữ “dãy các ký tự” thì cú pháp như sau:

char **cST1[n]** hoặc **char *ST21** ; với n là số ký tự cần lưu trữ

Vị trí khai báo biến và phạm vi sử dụng: Các biến có thể khai báo mọi nơi trong chương trình. Biến này được sử dụng từ vị trí khai báo cho đến ngoặc đóng '}' của ngoặc mở '{' tương ứng trong khối chứa biến

Chú ý: Tên biến cần phải gợi nhớ và nói lên mục đích sử dụng của biến (xem phụ lục "Nguyên tắc đặt trên trong lập trình")

2.3.3. Biểu Thức

Biểu thức được dùng để diễn đạt một công thức tính toán.

Biểu thức là sự kết hợp theo nguyên tắc toán học của biến, hằng, hàm, cặp dấu ngoặc đơn và các toán tử (+ , - , * , / , % , && , ||,...) cho ra một kết quả duy nhất.

$$\frac{-b-\sqrt{\Delta}}{2a}$$

Ví dụ: để tính nghiệm thứ nhất của phương trình bậc 2 thì trong lập trình biểu thức được này viết với khai báo biến ở trên như sau:

(-iHeSoB - sqrt(iDelta))/(2*iHeSoA); hoặc (-iHeSoB - sqrt(iDelta))/2/iHeSoA;

Có 3 loại biểu thức trong C: biểu thức số học, biểu thức logic và biểu thức quan hê.

Chú ý: Trong trường hợp đơn giản, biểu thức có thể là chỉ là 1 biến, 1 hằng, 1 hàm.

Phép gán: là gán giá trị của một biểu thức cho một biến có cùng kiểu dữ liệu với biểu thức

biến=biểu_thức;

Ví dụ: gán nghiệm thứ nhất của phương trình bậc 2 cho nghiem_x1 như sau:

fNghiem_x1= (-iHeSoB - sqrt(iDelta))/(2*iHeSoA);

Chú ý:

Ngôn ngữ C cho phép sử dụng phép gán ghép

A = B = C = 1; //gán 1 cho 3 biến a, b, c

Có thể vừa khai báo vừa khởi tạo giá trị ban đầu (gán) cho biến.

Ví dụ: int iA=5, iB=10;

2.4. CÁC TOÁN TỬ TRONG NGÔN NGỮ C

2.4.1. Toán Tử cin>>(Nhập), cout<< (Xuất)

2.4.1.1. Toán tử cin>>

Dùng để nhập dữ liệu từ bàn phím cho biến

Trong trường hợp 1 biến: **cin>>biến;**

Trong trường hợp k biến: **cin>>biến_1>>biến_2>>...>>biến_k;**

Ngoài ra để nhập dữ liệu cho 1 ký tự (giả sử biến c kiểu char) có thể dùng thêm cú pháp **c=cin.get();** hoặc **cin.get(c);** với **c** là biến kiểu **char.**

Riêng nhập dữ liệu cho biến lưu trữ dãy ký tự, nếu sử dụng cin thì không được nhập ký tự trống (ký tự spacebar ' '). Để nhập được ký tự trống, ta có thể dùng **cin.get(biến,n);** với n là một số nguyên dương hoặc **gets(biến);**

2.4.1.2. Toán tử cout<<

Dùng để xuất dữ liệu ra màn hình

Dạng **cout<<biểu thức<<"dãy ký tự"<<...;**

Xuất dữ liệu có định dạng

- Đối với số nguyên, ký tự hoặc dãy ký tự: dùng lệnh **cout.width(n);** (với n là số nguyên) trước lệnh xuất giá trị, để xuất ra giá trị ra màn hình bề ngang có n ký tự và canh lệch phải. Lệnh này chỉ ảnh hưởng đến lệnh xuất ngay sau nó.
- Đối với số thực: dùng lệnh **cout.precision(n);** để xác định số chữ số thập phân xuất ra màn hình. Nếu sử dụng **cout.precision();** thì máy tính sẽ in tất cả các chữ số phần thập phân ra màn hình

2.4.2. Chuyển Đổi Kiểu Dữ Liệu

Trong biểu thức tính toán số học, nếu tồn tại nhiều toán hạng có kiểu khác nhau (nguyên, thực, ...) thì ngôn ngữ quy định kết quả thu được cuối cùng là kiểu cao nhất có trong biểu thức đó. Sau đây là mô hình chuyển đổi từ kiểu dữ liệu đơn giản thành kiểu dữ liệu cao hơn:

int □ long □ float □ double □ long double

Chú ý: hông có sự chuyển đổi kiểu dữ liệu theo chiều ngược lại.

Ví dụ**:** int X, long P; float N;

Muốn tính y = X*P + N thì y phải khai báo kiểu gì? Y kiểu float.

Tuy nhiên người lập trình có thể tự chuyển kiểu dữ liệu để tính toán theo nguyên cú pháp **(kiểu_muốn_chuyển) biểu_thức**

Ví dụ: int iZ;

iZ=9/2;

iZ=(float)9/2;

2.4.3. Phép Toán Logic Và Quan Hệ

2.4.3.1. Phép toán logic

Ngôn ngữ C quy định giá trị đúng là ≠ 0 và giá trị sai là = 0

Có 3 phép toán logic trong ngôn ngữ C. Đó là phép AND (&&), OR(||) và NOT(!)

Ý nghĩa các phép toán logic được cho trong bảng sau:

Toán hạng A	Toán hạng B	AND	OR	NOT
		A&&B	A||B	!A

1	1	1	1	0
1	0	0	1	0
0	1	0	1	1
0	0	0	0	1

2.4.3.2. Phép toán quan hệ

Có 6 phép so sánh được cho dưới bảng dưới đây:

Phép toán so sánh	**Kí hiệu**	**Ví dụ**	**Kết quả**
Lớn hơn	>	3>5, 5>3	0, 1
Lớn hơn hoặc bằng	>=	5>=3, 3>=5	1, 0
Nhỏ hơn	<	3<5, 5<3	1, 0
Nhỏ hơn hoặc bằng	<=	5<=3, 3<=5	0, 1
Bằng	==	5==3, 5==5	0, 1
Khác	!=	5!=3, 5!=5	1, 0

*Chú ý***:** Kết quả của phép toán quan hệ là 0(sai) hoặc 1(đúng).

2.4.4. Phép Toán Trên Bit

Các phép toán theo bit liên quan đến cách biểu diễn nhị phân (biểu diễn bằng các giá trị 0 và 1) trong máy. Chỉ dùng cho kiểu số nguyên.

Phép toán	**Kí hiệu**	**Ghi chú**
AND	&	Phép AND theo từng bit
OR	|	Phép OR theo từng bit
XOR	^	Phép XOR theo từng bit
NOT	~	Phép đảo bit

Toán_hạng_1	Toán_hạng_2	Kết quả			
		&	|	^	~ Toán_hạng_1
1	1	1	1	0	0
1	0	0	1	1	0
0	1	0	1	1	1
0	0	0	0	0	1

Ví dụ:

unsigned int A, B, C, D, E;

A=3737; □ 0000 1110 1001 1001

B=7474; □ 0001 1101 0011 0010

C=A|B; □ 0001 1111 1011 1011 (8123)

D=B&C; □ 0001 1101 0011 0010 (7474)

E=C^D; □ 0000 0010 1000 1001 (649)

Phép dịch bit sang trái (Shift left)<<

Cú pháp biến<<n dịch trái n bit (n số nguyên), thêm số 0 vào các bit bên phải của biến. Còn đối với số nguyên có dấu thì bít dấu (bit đầu tiên) được giữ nguyên.

Đối với số nguyên không dấu

uiA = 21; // = 0000 0000 0001 0101

uiB = uiA << 1; // = 0000 0000 0010 1010=>= 21*2 = 42

uiC = uiA << 2 ; // = 0000 0000 0101 0100 =>= 21*4 = 84

Đối với số nguyên có dấu

iA = -21; // = 1111 1111 1110 1011

iB = iA << 1; // = 1111 1111 1101 0110=> -21*2 = -42

iC = iA << 2; // = 1111 1111 1010 1100=> -21*4 = -84

Chú ý:phép dịch trái n bit là phép nhân 2n.

Phép dịch bit sang phải (Shift right)>>

Cú pháp biến>>n dịch phải n bit (n số nguyên), thêm số 1 vào các bit bên trái của biến. Còn đối với số nguyên có dấu thì bít dấu (bit đầu tiên) được giữ nguyên.

Ví dụ:

Đối với kiểu không dấu

uiA = 84; // = 0000 0000 0101 0100

uiB = uiA >> 1; // = 0000 0000 0010 1010 => 84/2 = 42

uiC = uiA >> 2; // = 0000 0000 0001 0101=>= 84/4 = 21

Đối với kiểu có dấu

iA = -84; // = 1111 1111 1010 1100

iB = iA >> 1; // = 1111 1111 1101 0110 =>-84/2= -42

iC = iA >> 2; // = 1111 1111 1110 1011 => -84/4=-21

Chú ý:phép dịch phải n bit là phép chia 2n.

2.4.5. Phép Gán Mở Rộng

Trong tường hợp phép gán, nếu biến nằm bên trái dấu '=' có tham gia vào biểu thức tính toán bên phải dấu '=' có dạng **biến=biến phép_toán biểu thức;** thì ngôn ngữ cho phép viết gọn lại như sau **biến phép_toán = biểu thức;**

Ví dụ: **S=S+10;** có thể viết lại **S+=10;**

2.4.6. Toán Tử Tăng Giảm

Trong C đưa ra phép toán tăng (giảm) một ngôi cho kiểu nguyên và thực để tăng (giảm) cho biến 1 giá trị nguyên

Toán tử tăng: **++biến** hoặc **biến++** cho kết quả là **biến=biến +1**

Toán tử giảm: **--biến** hoặc **biến--** cho kết quả là **biến=biến -1**

Việc đặt toán tử trước hoặc sau biến đều cho kết quả của biến là như nhau. Tuy nhiên nếu các biến này tham gia vào biểu thức tính toán, thì kết quả biểu thức mà nó tham gia là khác nhau

Trường hợp toán tử nằm trước biến (++biến hoặc --biến) thì biến sẽ thực hiện toán tử xong rồi sau đó mới tham gia vào biểu thức

Trường hợp toán tử nằm sau biến (biến++ hoặc biến--) thì biến sẽ tham giam vào biểu thức tính toán rồi sau đó mới thực hiện toán tử.

Ví dụ:

```
int iA=10, iB=20, iZ;
iZ= iA++ + ++iB;      //iA=11, iB=21, iZ=31
iZ= iA++ + iB++;      //iA=11, iB=21, iZ=30
iZ= ++iA + ++iB;      //iA=11, iB=21, iZ=32
```

2.4.7. Toán Tử Sizeof

Toán tử này được dùng để tính kích thước (số byte) của một biến hoặc một kiểu dữ liệu

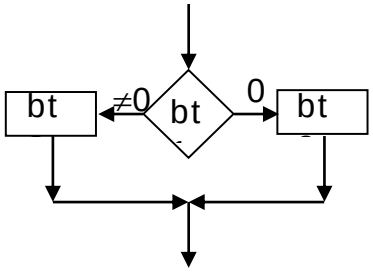
Cú pháp: **sizeof(X)** trong đó X là một biến hoặc tên của một kiểu dữ liệu. Kết quả của nó là một giá trị nguyên.

Ví dụ: sizeof(int) cho kết quả là 2

2.4.8. Toán Tử Điều Kiện

Toán tử điều kiện được dùng để biểu diễn một cấu trúc rẽ nhánh đơn giản.

Cú pháp bt1 ? bt2 : bt3;



Trong đó: bt1, bt2, bt3 là các biểu thức nào đó

Kết quả của toán tử này tuỳ thuộc vào bt Nếu bt1 cho kết quả □0 (đúng) thì kết quả sẽ là bt2. Còn trường hợp ngược lại kết quả sẽ là bt3.

Kiểu dữ liệu của toán tử điều kiện là kiểu lớn nhất của bt2 và bt3

Ví dụ: viết đoạn chương trình nhập vào 2 số nguyên, sau đó tìm số lớn nhất

```
{
int i1, i2;
cout << ”nhap 2 so nguyen:”;
cin >> i1 >> i2;
cout << ”Max(“ << i1 << ’,’ << i2 << ”)=” << (i1>i2) ? i1:i2;
}
```

2.4.9. Toán Tử Dãy (Toán Tử Phẩy)

bt1, bt2,...,btn;

Số các biểu thức lớn hơn 1

Các biểu thức ở trên 1 dòng và cách nhau bởi dấu ‘,’ và kết thúc bằng dấu ‘;’

Nguyên tắc thực hiện: Các biểu thức thực hiện lần lượt từ trái sang phải. Kết quả của biểu thức trước có thể được tham gia vào quá trình tính toán của biểu thức sau (nếu cần). Và kết quả của toán tử là giá trị của biểu thức cuối cùng bên phải.

Ví dụ:

```
int iM, iT, iH;
iM= (iT=2, iH=iT*iT+3, iH+iT);   //sẽ cho iT=2, iH=7 và iM=9
```

2.4.10. Độ Ưu Tiên Toán Tử

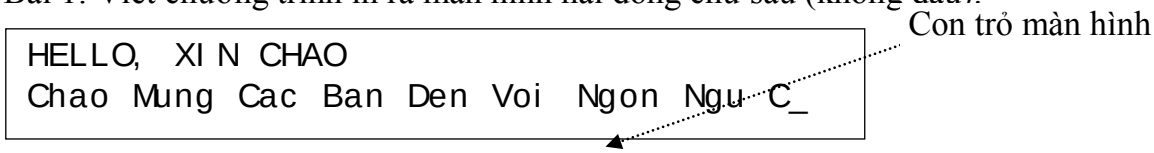
Ưu tiên	Toán tử	Chiều tính toán
1.	(), [], ->	Left Right
2.	! ,~, ++, --, -, sizeof(kiểu), *, &	Right Left
3.	* ,/, %	Left Right
4.	+ ,-	Left Right
5.	<<, >>	Left Right
6.	< ,<=, >, >=	Left Right
7.	== ,!=	Left Right
8.	& ,^, |	Left Right
9.	&&, ||	Left Right
10.	= ,+=, -=, *=, /=, %=, &=, ^=, |=	Right Left

Trong biểu thức có nhiều toán tử tham gia thì thứ tự tính toán của các toán tử phụ thuộc vào độ ưu tiên của nó, độ ưu tiên của các toán tử trong C theo bảng trên. Các toán tử ở trên cùng (vị trí 1) thì ưu tiên nhất và toán tử ở dưới cùng (vi trí 10) có độ ưu tiên thấp nhất. Các toán tử có độ ưu tiên hơn thì sẽ được thực hiện trước. Đối với các toán tử có cùng độ ưu tiên thì thứ tự thực hiện phép toán theo chiều tính toán ở cột 3.

Để viết biểu thức một cách chính xác thì nên dùng dấu ngoặc đơn để xác định độ ưu tiên tính toán.

2.5. BÀI THỰC HÀNH

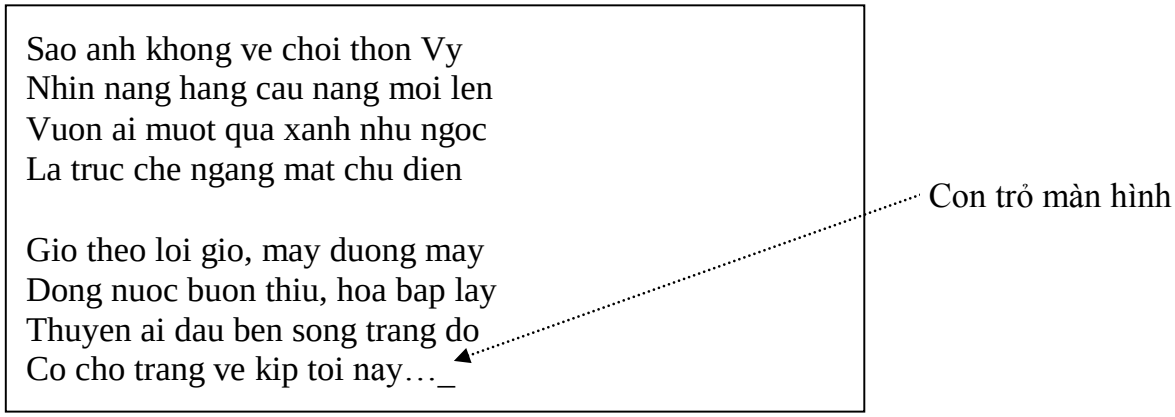
Bài 1: Viết chương trình in ra màn hình hai dòng chữ sau (không dấu):



Chương trình mẫu như sau:

```
#include <conio.h>
#include <iostream.h>
void main()
{
  clrscr();
  cout<<"HELLO, XIN CHAO";
  cout<<"\nChao Mung Cac Ban Den Voi Ngon Ngu C";
  getch();
}
```

Bài 2: Viết chương trình in ra màn hình trích đoạn bài thơ sau:



Bài 3: Viết chương trình in ra màn hình thông tin cá nhân học viên theo mẫu sau:

Bài 4: Viết chương trình hiển thị kết quả trên màn hình như sau:

```
    *
   ***
  *****
 *******
*********_
```

Bài 5: Viết chương trình in ra giá trị của biến số nguyên bằng 30 và biến số thực bằng 5.7 với yêu cầu như sau in ra màn hình

```
So nguyen co gia tri 30
So Thuc co gia tri 5.7_
```

Chương trình mẫu như sau:

```
#include<conio.h>
#include<iostream.h>
void main()
{
  int i;
  float f;
  clrscr();
  i=30;
  f=5.7;
  cout<< "So nguyen co gia tri"<<i;
  cout<< "\nSo Thuc co gia tri"<<f;
  getch();
}
```

Bài 6: Bổ sung vào chương trình của bài 5 để in ra giá trị của biến HANG

Chương trình mẫu như sau:

```
#include<conio.h>
#include<iostream.h>
void main()
{
int i;
float f;
const int HANG=9;
clrscr();
i=30;
f=5.7;
cout<< "So nguyen co gia tri: "<<i;
cout<< "\nSo Thuc co gia tri: "<<f;
cout<< "\nGia tri hang: "<<HANG;
i=8;
```

```
HANG=15;                          //sẽ báo lỗi dòng này
cout<< "So nguyen co gia tri: "<<i;
  cout<< "\nGia tri hang: "<<HANG;
  getch();s
}
```

Sau khi lưu và biên dịch thì chương trình sẽ báo lỗi ở dòng lệnh HANG=15 với câu: "Cannot modify a const object " nghĩa là bạn không thể thay đổi giá trị của hằng được. Như vậy, đối với biến, bạn có thể thay đổi giá trị của biến, còn với hằng thì không được thay đổi giá trị của nó trong chương trình.

Bài 7: Viết chương trình nhập vào 2 số nguyên, sau đó tính tổng và tích của 2 số đó.

Kết quả trên màn hình sẽ là:

```
Chuong trinh tinh tong va tich cua hai
so
Nhap vao so nguyen 1 = 10↵
Nhap vao so nguyen 2 = 20↵
Tong: 10 + 20 = 30
Tich: 10 * 20 = 200
```

Chương trình mẫu như sau:

```
#include <conio.h>
#include <iostream.h> //iostream.h: Thư viện chứa toán tử cin>> (nhập)
void main()
{
int i1, i2, iTong;
clrscr();
cout<<"Chuong trinh tinh tong va tich cua hai so";
cout<<"\nNhap vao so nguyen 1 =";
cin>>i1;
cout<<"Nhap vao so nguyen 2 =";
cin>>i2;
iTong=i1+i2;
cout<<"Tong: "<< i1<<" + "<< i2<<" = "<<iTong;
cout<<"\nTich:"<<i1<<"*"<<i2<<"="<<i1*i2;
getch();
}
```

Dựa vào bài mẫu trên lập trình các bài sau:

Bài 8: Viết chương trình nhập vào bán kính đường tròn. Sau đó tính chu vi, diện tích.

Bài 9: Viết chương trình nhập vào cạnh hình vuông . Sau đó tính chu vi, diện tích và độ dài đường chéo của hình vuông.

Bài 10: Viết chương trình nhập vào cạnh hình chữ nhật. Sau đó tính chu vi, diện tích và độ dài đường chéo của hình chữ nhật.

Bài 11: Viết chương trình khai báo 4 biến nguyên: i1, i2, i3, i4 và iKetQua thuộc kiểu số nguyên. Hãy in mỗi giá trị của biến iKetQua sau khi thực hiện chương trình với mỗi công thức sau:

iKetQua = i1+ i2- i3

iKetQua = i1* i2- i3/7

iKetQua = (i1+ i2)- i3/10

iKetQua = i1*((i2- i3)/4-40)

Đoạn mã của chương trình

```
#include<conio.h>
#include<iostream.h>
void main()
{
int i1, i2, i3, iKetQua;
clrscr();
i1=40;
i2=70;
i3=54;
iKetQua= i1+ i2- i3;
cout << "\nGia tri cua i1+ i2- i3 la " <<iKetQua;
iKetQua = i1* i2- i3/7;
cout << "\nGia tri cua i1* i2- i3/7 la " <<iKetQua;
iKetQua = (i1+ i2)- i3/10;
cout << "\nGia tri cua (i1+ i2)- i3/10 la " <<iKetQua;
iKetQua = i1*(( i2- i3)/4-40);
cout << "\nGia tri cua i1*(( i2- i3)/4-40) la " <<iKetQua;
getch();
}
```

Bài 12: Viết chương trình nhập vào 2 giá trị nguyên So1, So2. Sau đó in các giá trị của So1, So2, So3 của các biểu thức:

So3 = So1++ + So2++

So3 = So1++ + ++So2

So3 = ++So1 + So2++

So3 = ++So1 + ++So2

Kết quả trên màn hình như sau:

```
Chuong trinh mau ve phep toan tang(++),
giam(--)
Nhap vao gia tri so 1= 1↵
Nhap vao gia tri so2 = 2↵
so3 = 1++ + 2++ = 3          so1 = 2
     so2 = 3
so3 = 2++ + ++3 = 6          so1 = 3
     so2 = 4
```

Chương trình mẫu như sau:

```
#include<iostream.h>
#include<conio.h>
void main()
{
 clrscr();
 int i1, i2, i3;
 cout<<"Chuong trinh mau ve phep toan tang(++), giam(--)";
 cout<<"\nNhap vao gia tri so 1=";
 cin>>i1;
 cout<<"Nhap vao gia tri so 2 =";
 cin>>i2;
 cout<<"so3 = "<<i1<<"++ + "<<i2<<"++ = ";
 i3 = i1++ + i2++;
cout<<i3<<"\tso1 = "<<i1<<"\tso2 = "<<i2;
 cout<<"\nso3 = "<<i1<<"++ + ++"<<i2<<" = ";
 i3 = i1++ + ++i2;
 cout<< i3<<"\tso1 = "<<i1<<"\tso2 = "<<i2;
 cout<<"\nso3 = ++"<<i1<<" + "<<i2<<"++ = ";
 i3 =++i1 + i2++;
 cout<<i3<<"\tso1 = "<<i1<<"\tso2 = "<<i2;
 cout<<"\nso3 = ++"<<i1<<" + ++"<<i2<<" = ";
 i3=++i1 + ++i2;
 cout<<i3<<"\tso1 = "<<i1<<"\tso2 = "<<i2;
 getch();
}
```

Dựa vào bài mẫu số 7, hãy lập trình cho bài 8:

Bài 13: Viết chương trình nhập vào số nguyên So Sau đó in ra các giá trị So1, So2, So3 và So4 của các biểu thức sau:

So2 = So1++ + ++ So1

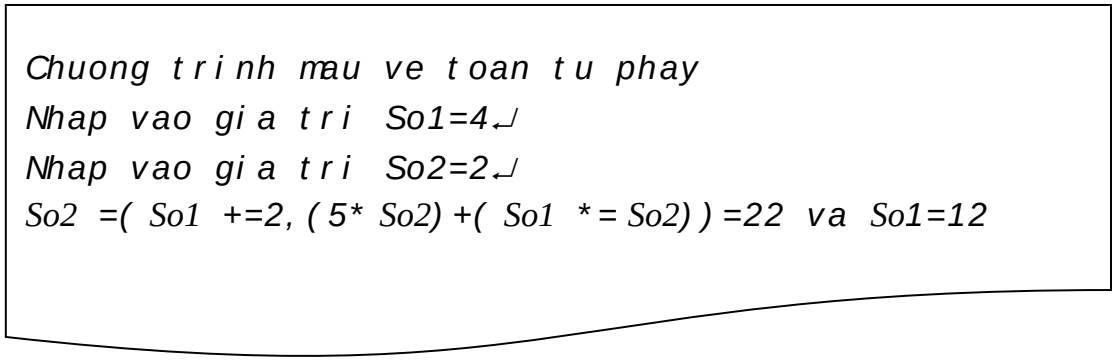
So 1 = So1+ = So1

So3 = So2* = So1/ =2

So4 = --So3 + So 3--

Bài 14: Viết chương trình nhập vào 2 số nguyên So1, So2. Sau đó in ra các giá trị của biểu thức sau:

So2 = (So1+ = 2, (5* So2) + (So1* = So2))



Chương trình mẫu như sau:

```
#include<iostream.h>
#include<conio.h>
void main()
{
 clrscr();
 int i1, i2;
 cout<<"Chuong trinh mau ve toan tu phay";
 cout<<"\nNhap vao gia tri So1 =";
 cin>> i1;
 cout<<"\nNhap vao gia tri So2 =";
 cin>> i2;
 i2=(i1 +=2,(5* i2)+( i1 *= i2));
 cout<<" So2=(So1 +=2,(5* So2)+( So1 *= So2)) = "<< i2<<” va So1=”<< i1;
 getch();
}
```

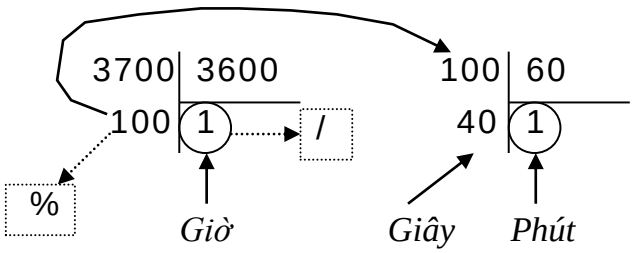
Dựa vào các bài mẫu trên lập trình cho các bài sau

Bài 15: Viết chương trình nhập vào 2 số nguyên a, b. Sau đó in ra các giá trị a, b, c của biểu thức sau:

c = (a- = 1, b/=a, (7*a) + (a+ = b))

Bài 16: Viết chương trình nhập vào số nguyên dương. Sau đó đổi số đó ra thời gian dạng Giờ : Phút : Giây. Ví dụ 3700 thì kết quả là:1h : 1m : 40 s

Hướng dẫn: dùng phép toán chia dư % (mod) và phép toán chia nguyên / (div).



Bài 17:Hãy tính các biểu thức sau:  với x, y nhập từ bàn phím

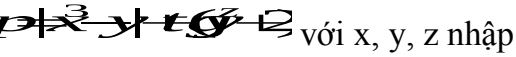
Chú ý: Sin và Cos tính bằng radian, công thức như sau: (góc * 3.14)/180

Bài 18:Hãy tính các biểu thức sau:  với x, y, z nhập từ bàn phím

Chương trình mẫu như sau:

```
#include<iostream.h>
#include<conio.h>
#include<math.h>              //math.h chứa các hàm toán học
void main()
{
 clrscr();
 int  iX, iY, iZ;
 float fP;
 cout<<"\nNhap vao x=";
 cin>>iX;
 cout<<"Nhap vao y=";
 cin>>iY;
 cout<<"Nhap vao z=";
 cin>>iZ;
 fP=pow(pow(iX, iY), 0/iZ)+log(2*pow(iX,2)+5);
 /*hàm pow(x,y) → xy và log(x) → ln(x) có trong thư viện math.h*/
 cout<<"p="<<fP;
 getch();
}
```

Dựa vào bài mẫu trên, hãy lập trình các bài sau

Bài 19: Hãy tính các biểu thức sau:  với x, y, z nhập từ bàn phím

Bài 20: Hãy tính các biểu thức sau:  với x, y, z nhập từ bàn phím

BÀI 3: CẤU TRÚC ĐIỀU KHIỂN

Một chương trình sẽ chạy tuần tự từng lệnh từ trên xuống. Tuy nhiên, để chương trình có thể suy luận như người, hay để chương trình thực hiện lặp lại một khối công việc thì cần có cấu trúc điều khiển (CTĐK). Có 3 loại CTĐK:

- Lệnh nhảy không điều kiện
- Lệnh rẽ nhánh như if, switch
- Lệnh vòng lặp (for, while, do..while)

3.1. LỆNH VÀ KHỐI LỆNH

Lệnh đơn: **cout << "N= " << N;**

Khối lệnh:

```
{
  cout << "N= ";
  cin >> N;
}
```

3.2. CẤU TRÚC RẼ NHÁNH

3.2.1. Cấu Trúc If

Cấu trúc IF giúp chương trình xử lý các trường hợp cần xét.

a) 01 trường hợp:

```
If (Biểu_Thức_Điều_Kiện) {
  Xử lý trong trường hợp biểu thức điều kiện đúng
}
```

Ví dụ 1:

```
If (dtb<5) {
  cout << "Loại yếu";
}
```

b) 02 trường hợp:

```
if (Biểu_Thức_Điều_Kiện) {
  Xử lý 1;
}
else {
  Xử lý 2;
}
```

Ví dụ 2:

```
if (dtb<5) {
  cout << "Loại yếu";
} else {
  cout << "Loại trung bình";
}
```

c) Hơn 02 trường hợp:

```
if (Biểu_Thức_Điều_Kiện_1) {
  Xử lý 1;
} else if (Biểu_Thức_Điều_Kiện_2) {
  Xử lý 2;
} else if (Biểu_Thức_Điều_Kiện_N-1) {
  Xử lý N-1;
} else {
  Xử lý N;
}
```

Ví dụ 3:

```
If (dtb<5) {
   cout "Loại yếu";
} else if (dtb<6.5) {
   cout << "Loại trung bình";
} else if (dtb<8) {
   cout << "Loại khá";
} else if (dtb<10) {
   cout << "Loại giỏi";
} else {
   cout << "Loại xuất sắc";
}
```

Ví dụ 4:vgiải phương trình bậc 2: AX2+BX+C=0

```
#include<conio.h>
#include<iostream.h>
#include<math.h>
void main()
{
      float fHesoA, fHeSoB, fHeSoC, fDelta, fNghiem_X1, fNghiem_X2;
      fDelta=pow(fHeSoB,2)-4*fHeSoA*fHeSoC;
      if(fDelTa>0)
      {
        fNghiem_X1=(-fHeSoB+sqrt(fDelta))/2/fHeSoA;
        fNghiem_X2=(-fHeSoB-sqrt(fDelta))/2/fHeSoA;
        cout<<"X1="<<fNghiem_X1<< " va X2="<<fNghiem_X2;
      }
      else
          if(fDelta==0) cout<<"X1=X2="<<-fHeSoB/2/fHeSoA;
          else cout<<" Phuong trinh vo nghiem";
}
```

3.2.2. Lệnh Switch

Lệnh switch được sử dụng trong trường hợp có hơn 2 lựa chọn.

```
switch (Biến) {
  case Giá_Trị_1:
     Xử lý 1;
     break;
  case Giá_Trị_2:
     Xử lý 2;
     break;
  case Giá_Trị_N-1:
     Xử lý N-1;
     break;
  default:
     Xử lý N;
}
```

Chú ý: nếu không có lệnh break thì swtich tiếp tục làm case tiếp theo

Ví dụ 1: chọn phép tính

```
int a, b, pheptinh;
cout << "a = ";  cin >> a;
cout << "b = ";  cin >> b;
cout << "Phep tinh: ";  cin >> pheptinh;
switch (pheptinh) {
  case 1: //Phép cộng
     cout << "Tong 2 so la " << a+b;
     break;
  case 2: //Phép trừ
     cout << "Hieu 2 so la " << a-b;
     break;
   case 3: //Phép nhân
     cout << "Tich so la " << a*b;
     break;
   case 4: //Phép chia
     cout << "Thuong 2 so la " << a/b;
     break;
   default:
     cout <<"Nhap sai phep tinh";
}
```

Ví dụ 2: để xuất ra các giá trị bằng chữ nhỏ hơn hoặc bằng iSize (0<iSize<5)

```
switch (iSize)
{
case 5: cout<<(" Nam ");
case 4: cout<<(" Bon ");
case 3: cout<<(" Ba ");
case 2: cout<<(" Hai ");
case 1: cout<<(" Mot "); break;
default: cout<<(" No ");
}
```

Ví dụ 3: cho biết vào trị nguyên iSize (0<iSize<10) có phải là số nguyên tố không

```
switch (iSize)
{
case 2: cout<<(" Nguyen to "); break;
case 3: cout<<(" Nguyen to "); break;
case 5: cout<<(" Nguyen to "); break;
case 7: cout<<(" Nguyen to "); break;
case 1: cout<<(" Khong Phai Nguyen to "); break;
case 4: cout<<(" Khong Phai Nguyen to "); break;
case 6: cout<<(" Khong Phai Nguyen to "); break;
case 8: cout<<(" Khong Phai Nguyen to "); break;
case 9: cout<<(" Khong Phai Nguyen to "); break;
default : cout<<(" No ");
}
```

Tuy nhiên có thể viết lại như sau:

```
switch (iSize)
{
case 2:
case 3:
case 5:
case 7: cout << " Nguyen to "; break;
case 1:
case 4:
case 6:
case 8:
case 9: cout <<" Khong Phai Nguyen to "; break;
default : cout << " No ";
}
```

3.3. CÁC LỆNH VÒNG LẶP

Quá trình tính toán có vòng lặp là quá trình tính toán mà có một phần công việc được thực hiện lặp đi lặp lại nhiều lần. Số lần thực hiện lặp lại có thể biết trước hoặc không biết trước

3.3.1. Vòng lặp: for, while và do...while

for (BT1; BTĐiềuKiện; BT3) {
  Lệnh;
}
	while (BTĐiềuKiện) {
	  Lệnh;
	}
		do {
		  Lệnh;
		} while (BTĐiềuKiện);

int n=5; **for (int i=1; i<=n; i++) {** ** cout << i << "\t";** **}** **getch();**	**int n=5;** **int i=1;** **while (i<=n) {** ** cout << i << "\t";** ** i++;** **}** **getch();**	**int n=5;** **int i=1;** **do {** ** cout << i << "\t";** ** i++;** **} while (i<=n);** **getch();**

Cách thực hiện:
(1) i=1
(2) Xét i<=5:
  Nếu đúng:
  (3) cout . . .
  (4) i=i+1
  Quay lại (2)
  Nếu sai:
  (5) getch()
	Cách thực hiện:
	(1) i=1
	(2) Xét i<=5:
	  Nếu đúng:
	  (3) cout . . .
	  (4) i=i+1
	  Quay lại (2)
	  Nếu sai:
	  (5) getch()
		Cách thực hiện:
		(1) i=1
		(2) cout . . .
		(3) i=i+1
		(4) Xét i<=5:
		  Nếu đúng:
		  Quay lại (2)
		  Nếu sai:
		  (5) getch()

Chú ý:

- For ( ) có thể bỏ các biểu thức, nhưng phải dùng break để tránh vòng lặp vô hạn.
- Do … While làm trước rồi mới xét điều kiện để lặp.

Ví dụ 1: nhập một số nguyên x không âm từ bàn phím

```
int x;
do {
      cout << "Nhập x="; cin>>x;
   } while (x<0);
```

3.3.2. Lệnh rẽ nhánh vô điều kiện: break, continue, goto, exit

a) Lệnh break:

Lệnh break được dùng để kết thúc vòng lặp (ngoài việc sử dụng trong lệnh switch).

Khi chương trình gặp lệnh break, thì chương trình sẽ dừng ngay lập tức vòng lặp trong nhất chứa no,. và đi thực hiện lệnh tiếp theo sau lệnh vòng lặp đó. Do đó lệnh break thông thường được sử dụng khi điều kiện dừng vòng lặp xãy ra.

b) Lệnh continue:

Trái với lệnh break (được dùng để dừng vòng lặp), lệnh continue được dùng để thực hiện ngay lập tức lần lặp tiếp theo của vòng lặp trong nhất chứa nó. Nghĩa là khi chương trình gặp lệnh continue thì chương trình sẽ bỏ qua không thực hiện các lệnh sau nó của vòng lặp trong nhất chứa nó mà thực hiện tiếp lần lặp tiếp theo của vòng lặp này.

Ví dụ 1: Tính trung bình cộng các số chẵn trong các số nguyên dương nhập từ bàn phím, quá trình nhập kết thúc khi nhập giá trị âm.

```
#include<conio.h>
#include<iostream.h>
void main()
{
  int iSum =0, iCount=0, iX;
  for (iDem =1; 1 ;iDem++) {
     cout<<"Nhap so thu "<<iDem<<" nhap <0 de dung:"; cin>>iX;
     if (iX<0) break;    //dừng vòng lặp
     if (iX%2!=0) continue; // thực hiện lần lặp tiếp theo
     iSum = iSum + iX;
     iCount++;
}
  if (iCount==0) cout<<"Trung binh cong so chan =0";
  else cout<<"Trung binh cong so chan ="<<(float) iSum /iCount;
 }
```

c) Lệnh goto:

Cú pháp: goto nhãn;

Mục đích: cho con trỏ chương trình (con trỏ lệnh) trỏ tới vị nhãn được đặt. Với nhãn phải tuân thủ theo nguyên tắc đặt tên và sau nhãn phải có dấu ':'

Nguyên tắc sử dụng:

- Nhãn và lệnh goto phải cùng nằm trong một hàm
- Nhãn và goto có thể nằm ngang cấp với nhau. Nghĩa là dùng goto để nhảy ngang hàng
- Không được phép nhảy từ bên ngoài khối (hoặc vòng lặp) vào bên trong khối (hoặc vòng lặp). Chỉ được phép nhảy từ bên trong khối(hoặc vòng lặp) ra bên ngoài khối (hoặc vòng lặp)

Nhận xét:

- Có thể dùng lệnh goto thay thế cho các cấu trúc lặp
- Có thể dùng lệnh goto để dừng vòng lặp (ngoài lệnh break)

Ví dụ 2: Tính trung bình cộng các số chẵn trong các số nguyên dương nhập từ bàn phím, quá trình nhập kết thúc khi nhập giá trị âm.

```
#include<conio.h>
#include<iostream.h>
void main()
{
  int iSum=0, iCount=0, iX;
  LAP:
  iDem++;
  cout<<"Nhap so thu "<<iDem<<" nhap <0 de dung:"; cin>>iX;
  if (iX<0) goto KET_THUC_LAP;
  if(iX%2!=0) goto LAP;
  iSum = iSum + iX;
  iCount++;
  goto LAP;
  KET_THUC_LAP:
  if (iCount==0) cout<<"Trung binh cong so chan =0";
  else cout<<"Trung binh cong so chan ="<<(float) iSum /iCount;
 }
```

d) Lệnh exit

Cú pháp exit(0);

Lệnh này được dùng để kết thúc chương trình khi gặp trường hợp muốn dừng ngay chương trình

3.4. BÀI TẬP

Bài 1: Viết chương trình nhập vào một số có kiểu dữ liệu nguyên. Kiểm tra số đó có lớn hơn hoặc bằng 0 hay không?. Nếu đúng thì in ra thông báo "So nhap vao la so duong". Nếu không thì in ra thông báo "So nhap vao la so am".

Đoạn mã chương trình

```
#include<conio.h>
#include<iostream.h>
void main()
{
  int  i;
  clrscr();
  cout<<"Nhap so nguyen:";
  cin>>i;
  if(i>=0)
        cout<<"\n So nhap vao la so duong";
  if(i<0)
        cout<<"\n So nhap vao la so am";
  getch();
```

```
}
```

Giải thích chương trình:

Trong ví dụ trên thì nếu giá trị biến i nhập vào là 25, phép so sánh i>=0 (25≥0) cho kết quả True nên lệnh cout bên trong lệnh if thứ 1 sẽ được thực hiện. Kết quả chương trình là in ra câu thông báo: "So nhap vao la so duong"

Nếu giá trị i nhập vào là -5, phép so sánh i>=0 (-5<0) cho kết quả False nên lệnh cout bên trong lệnh if thứ 2 sẽ được thực hiện. Kết quả chương trình là in ra câu thông báo: "So nhap vao la so am"

Bài 2: Viết chương trình nhập vào 2 số nguyên So1, So2. Kiểm tra So1 có chia hết cho So2 ? và hiện thông báo tương ứng

Gợi ý:

Nhập vào giá trị của 2 số

Lấy số dư của phép chia So1 cho So2 (dùng phép toán %)

Nếu số dư bằng 0 thì in thông báo "So1 chia hết cho So2"

Nếu số dư khác 0 thì in thông báo "So1 khong chia het cho So2"

Bài 3: Viết chương trình nhập vào 2 số nguyên. Hãy so sánh và in ra giá trị lớn nhất của 2 số nếu 2 số khác nhau. Nếu 2 số bằng nhau thì in thông báo hai số bằng nhau

Gợi ý: Sử dụng 3 lệnh if đơn theo thuật toán sau:

Nếu a>b thì in thông báo "So lon nhat la" và in giá trị của a

Nếu a<b thì in thông báo "So lon nhat la" và in giá trị của b

Nếu a=b thì in thông báo "Hai so bang nhau"

Bài 4: Viết chương trình giải phương trình bậc 1: ax+b=0

Bài 5: Hãy sửa chương trình ở bài 1 để có thể thay 2 lệnh if đơn bằng 1 lệnh if..else

Đoạn mã chương trình như sau

```
#include<conio.h>
#include<iostream.h>
void main()
{
  int  i;
  clrscr();
  cout<<"Nhap so nguyen:";
  cin>>i;
  if(i>=0)
        cout<<"\n So nhap vao la so duong";
else
        cout<<"\n So nhap vao la so am";
  getch();
}
```

Giải thích:

Thay vì chúng ta dùng 2 lệnh if đơn thì ta thay bằng 1 lệnh if..else. Chương trình trở nên gọn hơn.

Bài 6: Viết chương trình nhập vào 2 mốc thời gian t1 , t2 (với mỗi mốc thời gian nhập theo giờ, phút, giây) số nguyên dương. Sau đó cộng hai mốc thời gian này lại và xuất kết quả ra màn hình.

Kết quả trên màn hình như sau:

```
Nhap moc thoi gian t1
Nhap gio: 17↵
Nhap phut: 38↵
Nhap giay: 45↵
Nhap moc thoi gian t2
Nhap gio: 9↵
Nhap phut: 43↵
Nhap giay: 30↵
Moc t1:  17h : 38m : 45s
Moc t2:   9h : 43m : 30s
```

Chương trình mẫu như sau:

```
#include<conio.h>
#include<iostream.h>
void main()
{
 int iGio1,iPhut1,iGiay1;
 int iGio2,iPhut2,iGiay2;
 int iGio3,iPhut3,iGiay3;
 int iNho;
 clrscr();
 cout<<"Nhap vao moc thoi gian t1";
 cout<<"\nNhap gio:";
 cin>>iGio1;
 cout<<"Nhap phut:";
 cin>>iPhut1;
 cout<<"Nhap giay:";
 cin>>iGiay1;
 cout<<"\n\nNhap vao moc thoi gian t2";
 cout<<"\nNhap gio:";
 cin>>iGio2;
 cout<<"Nhap phut:";
 cin>>iPhut2;
 cout<<"Nhap giay:";
```

```
  cin>>iGiay2;
  iGiay3=(iGiay1+iGiay2)%60;
  iNho=(iGiay1+iGiay2)/60;
  iPhut3=(iPhut1+iPhut2+iNho)%60;
  iNho=(iPhut1+iPhut2+iNho)/60;
  iGio3=(iGio1+iGio2+iNho)%24;
  cout<<"Moc t1:    "<<iGio1<<"h : "<<iPhut1<<"m : "<<iGiay1<<"s";
  cout<<"\nMoc t2:    "<<iGio2<<"h : "<<iPhut2<<"m : "<<iGiay2<<"s";
  cout<<"\n         ------------------";
  cout<<"\nKet qua:   "<<iGio3<<"h : "<<iPhut3<<"m : "<<iGiay3<<"s";
  getch();
}
```

Bài 7: Nhập vào 3 số nguyên dương a, b và c. Kiểm tra xem a, b và c có tạo thành một tam giác không. Nếu là tam giác thì xuất ra màn hình diện tích, chu vi và loại tam giác(cân, đều, vuông…).

Hướng dẫn:

Xác định loại tam giác:

tam giác đều: a=b=c

tam giác cân: a=b hoặc a=c hoặc b=c

tam giác vuông: bình phương cạnh huyền bằng tổng bình phương hai cạnh góc vuông.

Bài 8: Giải phương trình trùng phương: AX4+BX2 +C=0

Hướng dẫn: Đặt y = x2 ≥ 0, phương trình trùng phươngAX4+BX2 +C=0 trở thành bậc hai Ay2+By +C=0. Sau đó giải phương trình bậc hai Ay2+By +C=0 theo nghiệm y (tham khảo bài giảng). Sau khi có nghiệm y, ta suy ra nghiệm x.

Bài 9: Giải bất phương trình bậc 2: f(x)=Ax2+Bx +C trong các trường hợp ≥0,>0,<0, □0

Hướng dẫn:

Dấu của tam thức bậc hai

f(x) = ax2 + bx + c (a ≠ 0)

Đặt Δ = b2 – 4ac. Khi đó

Δ < 0

x	-∞		+∞
f(x)	Cùng dấu với a		

Δ = 0

x	-∞		-b/(2a)		+∞
f(x)	cùng dấu với a		0		

cùng dấu với a

$\Delta > 0$

x-∞	x1	x2	+∞
f(x)cùng dấu với a	0	trái dấu a	0 cùng dấu với a

Bài 10: Nhập vào 3 số nguyên, sau đó tìm số lớn nhất với 3 đoạn lệnh khác nhau

Bài 11: Viết chương trình nhập năm từ bàn phím và kiểm tra đó phải là năm nhuận không.

Hướng dẫn:

Năm nhuận là năm chia hết cho 4 trừ những năm đầu thế kỷ.

Ví dụ:

1900 không phải là năm nhuận vì 1900 chia hết cho 4 nhưng là năm đầu thế kỷ.

2004 là năm nhuận vì 2004 chia hết cho 4 và không phải năm đầu thế kỷ.

Bài 12: Nhập vào một ngày bất kỳ trong năm (dương lịch từ 2000). Xuất ra ngày tiếp theo sau đó.

Tháng 2 của Năm nhuận.

Ngày 31 tháng 12 hằng năm.

Ví dụ:

1. Nhap vao nam: 2000

Nhap vao thang bat ky trong nam 2000: 12

Nhap vao ngay bat ky trong nam 2000: 31

Ngay tiep theo se la: ngay 1 thang 1nam 2001_

2. Nhap vao nam: 2004

Nhap vao thang bat ky trong nam 2004: 2

Nhap vao ngay bat ky trong nam 2004: 28

Ngay tiep theo se la: 29/1/2004

Bài 13: Viết chương trình nhập vào 5 số nguyên, tìm số lớn nhất trong 5 số đó (dùng 4 lệnh if)

Bài 14: Viết chương trình giải hệ phương trình sau:

$$\begin{cases} Ax + By = C \\ Dx + Ey = F \end{cases}$$

Hướng dẫn: Dùng phương pháp tính định thức

Công thức như sau:

$$D = \begin{vmatrix} A & B \\ D & E \end{vmatrix} \quad Dx = \begin{vmatrix} C & B \\ F & E \end{vmatrix} \quad Dy = \begin{vmatrix} A & C \\ D & F \end{vmatrix}$$

* $D \neq 0$: nghiệm duy nhất

  x = Dx/D; y = Dy/D;

* D = 0: Nếu Dx= 0 và Dy= 0 thì hệ có vô số nghiệm

  Nếu Dx ≠0 hoặc Dy≠ 0 thì hệ vô nghiệm

Bài 15: Viết chương trình nhập vào số nguyên dương X(0 <=X<=9). Sau đó xuất ra màn hình giá trị chữ của nó.

Kết quả trên màn hình sẽ là:

```
Nhap vao 1 gia tri trong khoang
0..9 : 8↵
8→ Tam_
```

Chương trình mẫu như sau:

```
#include<conio.h>
#include<iostream.h>
void main()
{
  int iX;
  clrscr();
  cout<<"Nhap vao 1 gia tri trong khoang 0..9";
  cin>>iX;
  cout<<iX<<"□";
  switch(iX){
        case 0: cout<<"Khong";break;
        case 1: cout<<"Mot";break;
        case 2: cout<<"Hai";break;
        case 3: cout<<"Ba";break;
        case 4: cout<<"Bon";break;
        case 5: cout<<"Nam";break;
        case 6: cout<<"Sau";break;
        case 7: cout<<"Bay";break;
        case 8: cout<<"Tam";break;
        case 9: cout<<"Chin";break;
        default: cout<<"gia tri nay nam ngoai khoang 0..9";
  }
}
```

Bài 16: Viết chương trình nhập vào số nguyên dương X(0 <=X<=9). Sau đó kiểm tra X có phải số nguyên tố không (chỉ sử dụng lệnh if hoặc switch).

Hướng dẫn: X là số nguyên tố khi và chỉ khi x>1 và chỉ chia hết cho 1& chính nó

Bài 17: Viết chương trình nhập vào số nguyên dương n(0<n<8) từ bàn phím. Sau đó xuất thứ theo giá trị n vừa nhập.

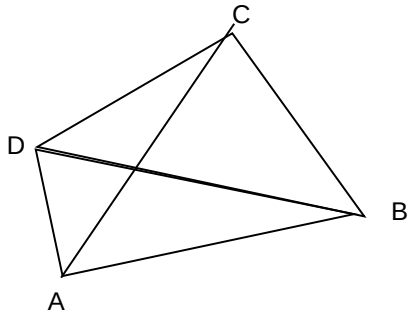
Ví dụ: n = 1-> "Chu nhat",n = 2-> "Thu hai"…, nếu n<1 hoặc n>7 thì kết quả "Khong phai"

Bài 18: Viết chương trình sử dụng như một máy tính điện tử đó là: nhập vào 2 số a, b. Sau đó nhập vào các phép toán (+,-,*,/, ^) và xuất kết quả ra màn hình theo phép toán vừa nhập.

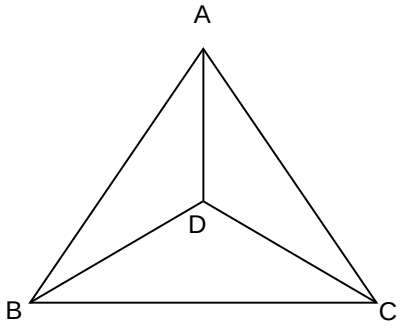
Mô phỏng màn hình kết quả :

```
Chuong trinh may tinh dien tu
Nhap vao a: 4↵
Nhap vao b: 3↵
Nhap vao phep toan (+, -, *, /, ^) : *↵
4 * 3 = 12_
```

Bài 19: Nhập vào toạ độ 4 điểm A, B, C và D trong mặt phẳng Oxy. Hãy kiểm tra xem điểm D có thuộc tam giác ABC.



D không thuộc $\triangle ABC$



D thuộc $\triangle ABC$

Bài 20: Nhập vào toạ độ 3 điểm A, B và C trong mặt phẳng Oxy. Sau đó tính diện tích, chu vi và xác định loại của tam giác này (vuông, cân, đều, …)

Hướng dẫn:

Tính khoảng cách giữa các cạnh

Áp dụng bài 2.7.

Bài 21: Tính tiền điện hàng tháng. Nhập vào số đầu kỳ và số cuối kỳ và các đơn giá m1, m2 và m3. Tiền điện tính theo lũy tiến như sau: 100 số đầu tiền tính đơn giá m1/1kw, 100 số tiếp theo tính đơn giá m2/1kw, còn lại tính đơn giá m3/1kw.

Bài 22: Viết chương trình tính tổng của số nguyên lớn nhất và số nguyên nhỏ nhất của 3 số nguyên a, b và c

Bài 23: Viết chương trình tính tổng dãy số: S=12 + 22 + 32 + … + 102

Thuật toán:

Bước 0: gán S=0; {gán giá trị ban đầu cho S}

Bước 1: gán S=S+1*1; {được S=12}

Bước 2: gán S=S+2*2; {được S=12+22}

Bước 3: gán S=S+3*3; {được S=12+22+32}

v.v.

Bước 10: gán S=S+10*10; {được S=12+22 + 32 +…+102}

Quá trình từ bước 1 đến bước 10 được gọi là phép cộng dồn vào biến S. Tại bước thứ i, lấy giá trị của biến S cộng với i2, kết quả lại được gán cho biến S, do đó giá trị của biến S được tăng thêm một lượng bằng i2. Khi i thay đổi từ 1 đến 10 thì các số 12, 22, 32, … , 102 đều được cộng dồn vào S, kết quả là sau bước thứ 10 giá trị của S đúng bằng tổng 12 + 22 + 32 + … + 102

Viết chương trình

```
#include<conio.h>
#include<iostream.h>
void main()
{
  int  iDem;     //khai bao bien nguyen iBienDem
  int  iTong;            //khai bao bien iTong thuoc kieu so nguyen
  iTong=0;
  for(iDem=1; iDem<=10; iDem++)
          iTong=iTong+iDem*iDem;
  cout<<"\n Tong binh phuong day so la: "<<iTong;
  getch();
}
```

Bài 24: Viết chương trình tính tích dãy số S=1*2*3*…*10 (S=10!) và in kết quả ra màn hình.

Hướng dẫn bằng thuật toán:

Bước 0: gán S=1; {gán giá trị ban đầu cho S}

Bước 1: gán S=S*1; {được S=1}

Bước 2: gán S=S*2; {được S=1*2}

Bước 3: gán S=S*3; {được S=1*2*3}

v.v.

Bước 10: gán S=S*10; {được S=1*2 * 3*…*10 = 10!}

Nếu trong bài trên ta phải cộng dồn vào biến S thì trong bài này ta phải nhân dồn vào biến S. Tại bước thứ i, lấy giá trị của biến S nhân với i, rồi lại gán kết quả cho biến S. Khi i thay đổi từ 1 đến 10 thì S sẽ tích lũy đủ các thừa số 1, 2, 3,…, 10 và giá trị của S sau bước thứ 10 đúng bằng 1*2*3*…*10 = 10!

Quá trình thực hiện từ bước 1 đến bước thứ 10 được mô tả bằng câu lệnh for:

```
for(iDem=1; iDem<=10; iDem++)
    S=S*iDem;
```

Một cách tổng quát, để tính tích: S=1*2*…*n, trong đó n là số nguyên dương bất kỳ, ta dùng hai lệnh:

```
S=1;
for(iDem=1; iDem<=n; iDem++)
    S=S*iDem;
```

Lưu ý phải khởi tạo giá trị ban đầu cho S=1;

Bài 25: Viết chương trình tính $S = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n}$ tổng với n□N được nhập từ bàn phím

Chương trình mẫu như sau:

```
#include<conio.h>
#include<iostream.h>
void main()
{
    int  iN, iDem;
   float fTong=0;       /*khai bao va gán giá trị đầu cho một số thực*/
  clrscr();
  cout<<"Nhap vao gia tri de tinh tong:";
  cin>>iN;
  /*Vòng lặp FOR*/
     for(iDem=1; iDem<=iN; iDem++)
fTong=fTong+1.0/iDem;
  /*Vòng lặp WHILE
  iDem=1;
  while(iDem<= iN)
  {
 fTong=fTong + 1.0/iDem;  //fTong+=1.0/iDem;
 iDem++;                          //iDem=iDem+1;
  }
  */
  /*Vòng lặp DO..WHILE
  iDem=1;
 do{
fTong=fTong + 1.0/iDem;   //fTong+=1.0/iDem;
 iDem++;                          //iDem=iDem+1;
  }while(iDem<=iN);
  */
  cout<<"Tong nghich dao cua “<<iN<<” so tu nhien dau tien la:"<<fTong;
  getch();
```

}

Bài 26: Viết chương trình tính tổng S = 1 + 3 + … + (2k + 1) với (2k+1)≤n□N được nhập từ bàn phím

Bài 27: Viết chương trình tính tổng  với n□N được nhập từ bàn phím

Hướng dẫn: để tính ii hàm pow(i,i) và chú ý đến biến i chẵn hoặc lẻ.

Bài 28: Viết chương trình tính tổng  với n□N được nhập từ bàn phím

Hướng dẫn: Tính i! ở trong thân vòng lặp.

Bài 29: Viết chương trình tính  với độ chính xác Epsilon nhập từ bàn phím

```
Nhap vao x: 1↵
Nhap vao do chi nh xac
epsi l on: 0. 0001↵
e^1=2. 7182_
```

Chương trình mẫu như sau:

```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
#include<graphics.h>
void main()
 {
  int iDem=1,iX;
  float fEx,fEpsilon, fMu, fGt;
  clrscr();
  cout<<"\nNhap vao x:"; cin>>iX;
  cout<<"Nhap vao do chinh xac epsilon:";cin>>fEpsilon;
  fMu=1;
  fGt=1;//tính giai thừa
  fEx=1;
  while((fMu/gt)>fEpsilon)
  { fEx=fEx+fMu/fGt;
```

```
  fMu=fMu*iX;
  fGt=fGt*i;
  iDem+=1;
 }
 cout<<"e^"<<iX<<" = "<<fEx;
 getch();
}
```

Bài 30: Viết chương trình tính Sin(x) theo công thức = $x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \ldots$ … với độ chính xác Epsilon. Epsilon và x được nhập từ bàn phím (x tính bằng radian)

Bài 31: Viết chương trình tính cos(x) theo công thức = $1 - \frac{x^2}{2} + \frac{x^4}{4} - \frac{x^6}{6} + \frac{x^8}{8} \ldots$ với độ chính xác Epsilon. Epsilon và x được nhập từ bàn phím (x tính bằng radian)

Bài 32: Tính lãi suất vay ngân hàng.

Vay số tiền là a, lãi suất là k%, số tháng vay là n. Tính tổng số tiền phải trả sau n tháng vay.

Có số tiền cần trả, số tháng cần vay. Tìm số tiền cần vay ban đầu

Có số tiền cần vay và số tiền phải trả. Tính số tháng vay.

Bài 33:Tính $n!! = \begin{cases} 2*4*6\ldots \text{ nếu n chẵn} \\ 1*3*5*7\ldots \text{nếu n lẻ} \end{cases}$

Bài 34:Hãy tính số hạng thứ k của dãy số sau: 1 1 2 3 5 8 13 21

Ví dụ: nếu n =7 thì kết quả là 13

Bài 35:Kiểm tra xem số nguyên n có thể phân tích thành tổng 2 số chính phương không.

Ví dụ: 25=9+16 (Yes) , 24 (No)

Bài 36:Tìm k nhỏ nhất để 2k □n. với n nhập từ bàn phím (Yêu cầu sử dụng vòng lặp)

Ví dụ : n=15 thì k=4 vì 24=16□15

Bài 37:Tìm k lớn nhất để 4k□n. với n nhập từ bàn phím (Yêu cầu sử dụng vòng lặp)

Ví dụ: n=65 thì k=3 vì 43=64□65

Bài 38: Viết chương trình nhập vào 1 số nguyên n và in giá trị bằng chữ của số đó.

Ví dụ: n = 123 in ra "Một trăm hai mươi ba"

Bài 39: Viết chương trình tìm các số tương ứng với các ký tự theo đề sau (các ký tự có giá trị khác nhau)

```
    B  I  T
X
          8
-----------
B  Y  T  E
```

Ví dụ:

```
    3  9  7
X
          8
-----------
3  1  7  6
```

Bài 40: Viết chương trình tìm số có 3 chữ số sao cho tổng các lập phương của chúng bằng chính nó

Ví dụ: 153 = 13 + 53 + 33

Bài 41: Viết chương trình nhập n số nguyên và tính trung bình cộng các số nguyên tố trong n số nguyên đó (n nhập từ bàn phím)

Mô phỏng màn hình kết quả như sau:

```
Nhap n = 5↵
So 1 = 7↵
So 2 = 3↵
So 3 = 4↵
So 4 = 2↵
So 5 = 1↵
Trung binh cac so nguyen to la: 4.00_
```

$$C_n^k = \frac{n!}{k!(n-k)!}$$

Bài 42: Viết chương trình tính tổ hợp chập k của n phần tử.

Bài 43: Viết hàm liệt kê các số nguyên tố ≤ n với n□N

Ví dụ: n=7 □ các số nguyên tố ≤ 7 là: 2 3 5 7

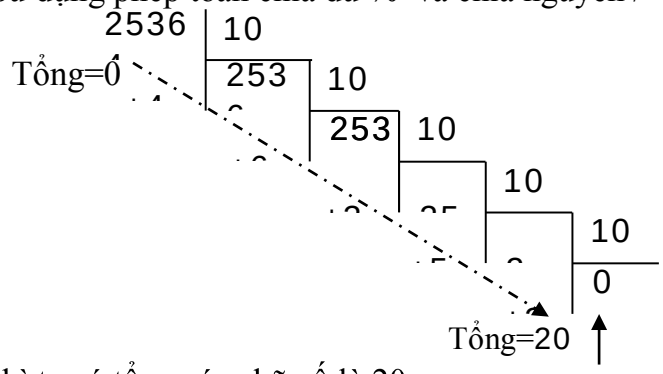
Bài 44: Viết hàm liệt kê n số nguyên tố đầu tiên với n□N

Ví dụ: n=7 □ 7 số nguyên tố đầu tiên là: 2 3 5 7 11 13 17

Bài 45:Nhập một số nguyên dương n, sau đó tính tổng các chữ số của số nguyên đó

Ví dụ: Cho n=25364 gồm 5 chữ số 2, 5, 3, 6, 4. Vậy tổng các chữ số của n là: 20

Hướng dẫn: Sử dụng phép toán chia dư % và chia nguyên /

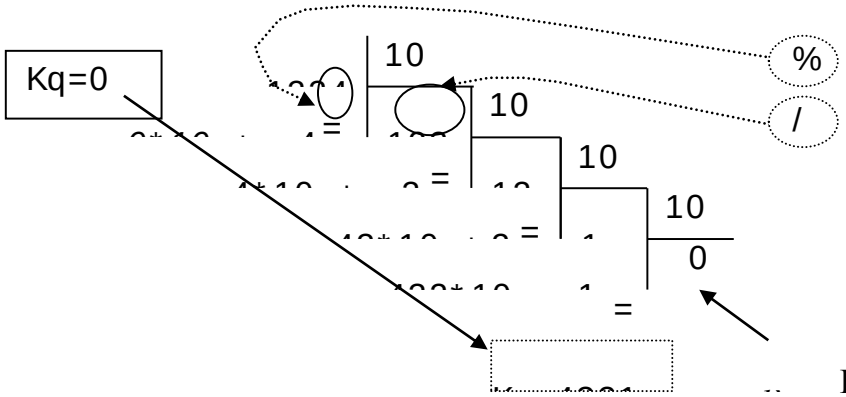


 Khi n = 0 thì ta có tổng các chữ số là 20.

Bài 46: Nhập vào một số nguyên dương n, sau đó đảo ngược số nguyên đó

Ví dụ:

n = 1234 sau khi đảo ngược ta có số m = 4321



Hướng dẫn:

Bài 47: Nhập vào số nguyên dương n, sau đó bỏ đi số 0 và 5 trong số đó. Xuất kết quả sau khi xử lý

Ví dụ:

N = 157604

Bước 1: 4671

Bước 2: 1764

Bài 48: Nhập vào một số nguyên n, kiểm tra số đó có đối xứng không

Bài 49: Viết hàm liệt kê các số hoàn thiện ≤ n với n□N

Hướng dẫn: số hoàn thiện là số mà tổng các ước thật sự của nó bằng chính nó.

Ví dụ: Cho x = 6, các ước của 6 (trừ 6) là 1, 2, 3. Vậy 1+2+3 = 6. Do đó số 6 là số hoàn thiện.

BÀI 4: HÀM

Hàm giúp tạo một chương trình con xử lý một việc nào đó để main( ) sử dụng nhiều lần và còn giúp tổ chức và gỡ rối main( ) dễ dàng.

4.1. XÂY DỰNG HÀM

Hàm là chương trình con thực hiện một công việc nào đó, có hoặc không có dữ liệu đầu vào và có hoặc không có trả về một giá trị cụ thể. Hàm có sẵn trong các thư viện chuẩn stdio.h, conio.h, iostream.h v.v hoặc hàm do người lập trình tự định nghĩa.

4.1.1. Hàm Không Trả Về Giá Trị Và Không Nhận Tham Số Đầu Vào

```
void Tong1( ) {
  int a, b, tong;
  cout << "a = ";  cin >> a;
  cout << "b = ";  cin >> b;
  tong = a + b;
  cout << "Tong 2 so la ” << tong;
}
```

```
void main( ) {
  Tong1( );
  getch( );
}
```

➔ main( ) không thể truy xuất giá trị của a và b.

4.1.2. Hàm Nhận Tham Số Đầu Vào Và Không Trả Về Giá Trị

```
void Tong2(int a, int b) {
  int tong;
  tong = a + b;
  cout << "Tong 2 so la " << tong;
}
```

```
void main( ) {
  Tong2(10,20);
  getch( );
}
```

4.1.3. Hàm Trả Về Giá Trị Và Không Nhận Tham Số Đầu Vào

```
int Tong3( ) {
  int a, b, tong;
  cout << "a = ";  cin >> a;
  cout << "b = ";  cin >> b;
  tong = a + b;
  return tong;
}
```

```
void main( ) {
  int x;
  x= Tong3( );
  cout << "Tong 2 so la " << x;
  getch( );
}
```

➔ trong Tong3( ), kiểu của biến tong phải cúng kiểu của hàm là int.

➔ trong main( ), kiểu của biến x phải cúng kiểu của hàm Tong3 là int.

4.1.4. Hàm Trả Về Giá Trị Và Nhận Tham Số Đầu Vào

```
int Tong4(int a, int b) {
  int tong;
  tong = a + b;
  return tong;
}
```

```
void main( ) {
  int tong;
  tong= Tong4(10,20);
  cout << "Tong 2 so la " << tong;
  getch();
}
```

4.1.5. Tham Trị

```
void HoanVi(int a, int b) {
  int tam;
  tam=a; a=b; b=tam;
}
```

```
void main( ) {
  int a=10, b=20;
  HoanVi(a,b);
  cout << "a=" << a << ", b=" << b;
  getch( );
}
```

➔ Sau khi main( ) gọi HoanVi(a,b) thì kết quả a=10, b=20

4.1.6. Tham Chiếu

```
void HoanVi(int &a, int &b) {
  int tam;
  tam=a; a=b; b=tam;
}
```

```
void main( ) {
  int a=10, b=20;
  HoanVi(a,b);
  cout << "a=" << a << ", b=" << b;
  getch( );
}
```

➔ Sau khi main( ) gọi HoanVi(a,b) thì kết quả a=20, b=10

4.1.7. Biến toàn cục

```
int N=5, Y=10;
void main( ) {
  int N=15, X=20;
  cout << "\n N=" << N;
  cout << "\n X=" << X;
  cout << "\n Y=" << Y;
  getch( );
}
```

➔ N=15, X=20, Y=10

```
int N=5, Y=10;
void main( ) {
  int N=15, X=20;
  cout << "\n N=" << ::N;
  cout << "\n X=" << X;
  cout << "\n Y=" << Y;
  getch( );
}
```

➔ N=5, X=20, Y=10

Ví Dụ: Viết hàm tính tổ hợp chập k của n phần tử

Đầu vào : n, k □N , k□n; Đầu ra: $C_n^k = \frac{n!}{k! * (n-k)!}$

// Viết hàm tính giai thừa của 1 số nguyên dương. Đầu vào n□N , đầu ra n!

```
unsigned long Giai_Thua(unsigned int uiN) {
  if (uiN==0) return 1;
  unsigned int uiDem;
  unsigned long ulKetQua=1;
  for(uiDem=1;uiDem<=uiN; uiDem++)  ulKetQua=ulKetQua*uiDem;
  return uiKetQua;
}
```

```
//Hàm tính tổ hợp
unsigned int To_Hop(unsigned int uiN, unsigned int uiK) {
  return Giai_Thua(uiN)/ Giai_Thua (uiK)/ Giai_Thua (uiN- uiK);
// gọi hàm tính giai thừa
}
//Hàm main sử dụng hàm tính tổ hợp chập k của n phần tử
void main() {
  unsigned int uiN, uiK;
  cout << "Tinh to hop C(n,k) \n";
  cout << "nhap n>=k>=0:"; cin >> uiN >> uiK;
  cout << "C(" << uiN << "," << GiaTriK << ")=" << To_Hop(uiN, uiK);
}
```

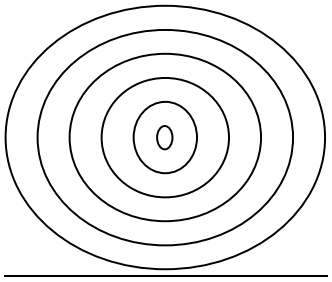
4.2. GIẢI THUẬT ĐỆ QUY

4.2.1. Khái Niệm

Một đối tượng được gọi là đệ qui nếu nó bao gồm chính nó như một bộ phận hoặc nó được định nghĩa dưới dạng chính nó nhưng có kích thước nhỏ hơn

Ví dụ 1: Thư mục được gọi là đệ quy bởi vì thư mục chứa các thư mục con và các tập tin.

Ví dụ 2: Tập hợp một chuỗi các hình tròn đồng tâm, khác bán kính (bán kính nhỏ dần) xếp chồng lên nhau như hình dưới đây:



Ví dụ 3: Định nghĩa đệ qui về tên như sau:

- Mỗi chữ cái là 1 tên
- Nếu t là 1 tên thì t +<chữ cái> cũng là 1 tên

Ví dụ 4: Định nghĩa đệ qui n! như sau:

- 0!=1
- n!=(n-1)! * n

Ví dụ 5: Giả sử hàm f được định nghĩa bằng đệ quy sau:

$$F(n) = \begin{cases} 0 & \text{nếu } n = 2 \\ 2F(n-1)+3 & \text{nếu} \end{cases}$$

Ví dụ 6: Dãy số Fibonaci được định nghĩa theo đệ quy sau:

$$F(n) = \begin{cases} 1 & \text{nếu } n \le 2 \\ F(n-1)+F(n-2) & \text{nếu } n > 2 \end{cases}$$

Ví dụ 7:

$$F(n) = \begin{cases} 0 & \text{nếu } n = 0 \\ F(n-1) & \text{nếu n lẻ} \\ n + F(n-2) & \text{nếu n chẵn} \end{cases}$$

4.2.2. Giải Thuật Đệ Quy

4.2.2.1. Khái niệm

Nếu một lời giải của bài toán P được thực hiện bằng lời giải của 1 hay nhiều của bài toán P'nhỏ hơn có dạng giống như P thì đó là một lời giải đệ quy. Giải thuật tương ứng với lời giải đệ quy gọi là giải thuật đệ quy.

4.2.2.2. Cấu trúc của giải thuật đệ quy

Một giải thuật đệ quy bao giờ cũng gồm 2 phần

- Phần neo: Xác định điểm kết thúc của một giải thuật đệ quy. Trường hợp này còn được gọi là trường hợp suy biến. Nếu một giải thuật đệ quy không có trường hợp suy biến thì sẽ dẫn đến lặp vô hạn và sinh lỗi khi thi hành
- Phần đệ quy: Phân tích và xây dựng trường hợp chung của bài toán (có nghĩa là đưa bài toán về bài toán cùng loại nhưng với dữ liệu nhỏ hơn).

4.2.2.3. Xây dựng giải thuật đệ quy

Khi cài đặt thuật toán đệ quy ta tiến hành những bước sau:

Bước 1: Xác định mục đích, đầu vào và đầu ra để từ đó xác định tên tiêu đề hàm (tên hàm và tham số hình thức của nó)

Bước 2:Xác định trường hợp suy biến (neo)

Bước 3: Phân tích và xây dựng trường hợp chung của bài toán (phần đệ quy). Có nghĩa là đưa bài toán về bài toán cùng loại nhưng với dữ liệu nhỏ hơn.

Từ 3 bước trên, áp dụng nguyên tắc viết ngôn ngữ ta xây dựng được hàm đệ qui (thông thường sử dụng toán tử điều kiện (if…else) để viết hàm đệ quy.

Ví dụ 1: Tính x^y với x^y được định nghĩa như sau: $x^y = \begin{cases} 1 & khi\ y=0 \\ x \cdot x^{y-1} & khi\ y>0 \end{cases}$

1. Bước 1: Xác định tham số đầu vào là x và y, tham số đầu ra là x^y. Ta luôn nhận được giá trị x^y duy nhất, do đó ta sử dụng hàm có kiểu trả về.
2. Bước 2: Phần neo: y=0 thì tổng $x^0 = 1$

3. Bước 3: Phần đệ quy: là trường hợp thực hiện lại bài toán với giá trị nhỏ hơn y = y - 1. Tức là ứng với trường hợp thực hiện lại lời gọi cũng đều có chung tham số đầu vào là y nhưng với giá trị nhỏ hơn y = y – 1 và đều có khuynh hướng đến trường hợp suy biến y = 0.

Giải thuật

```
float x_mu_y (float x, int y)
{       if(y==0) return 1;
        else return x*x_mu_y(x,y-1);      //gọi đệ quy x_mu_y(x,y-1) để tính x^(y-1)
        }
```

Ví dụ 2: Tính tổng n số nguyên dương đầu tiên (n≥0).

$$S = 1 + 2 + \ldots + n$$

4. Bước 1: Xác định tham số đầu vào là n, tham số đầu ra là S. Ta luôn nhận được giá trị tổng S duy nhất, do đó ta sử dụng hàm có kiểu trả về.

5. Bước 2: Phần neo: n=0 thì tổng S = 0

6. Bước 3: Phần đệ quy: là trường hợp thực hiện lại bài toán với giá trị nhỏ hơn n = n - 1. Tức là ứng với trường hợp thực hiện lại lời gọi cũng đều có chung tham số đầu vào là n nhưng với giá trị nhỏ hơn n = n – 1 và đều có khuynh hướng đến trường hợp suy biến n = 0.

$$tong(n) = \begin{cases} 0 & \text{nếu } n = 0 \\ n+tong(n-1) & \text{nếu } n > 0 \end{cases}$$

Giải thuật hàm đệ quy được viết như sau:

```
int     tong(int n)
{
if(n==0)      return 0;
else    return (n + tong(n-1));
}                                   //gọi đệ quy
```

Ví dụ 3: Đếm số chữ số của số nguyên N. Ví dụ N = 123 có 3 chữ số

7. Bước 1: Xác đinh tham số đầu vào là n, tham số đầu ra là số chữ số của n. Sử dụng hàm có kiểu trả về vì bài toán này trả về một giá trị duy nhất là số chữ số của n.

8. Bước 2: Phần neo: n = 0 thì số chữ số bằng 0.

9. Bước 3: Phần đệ quy:

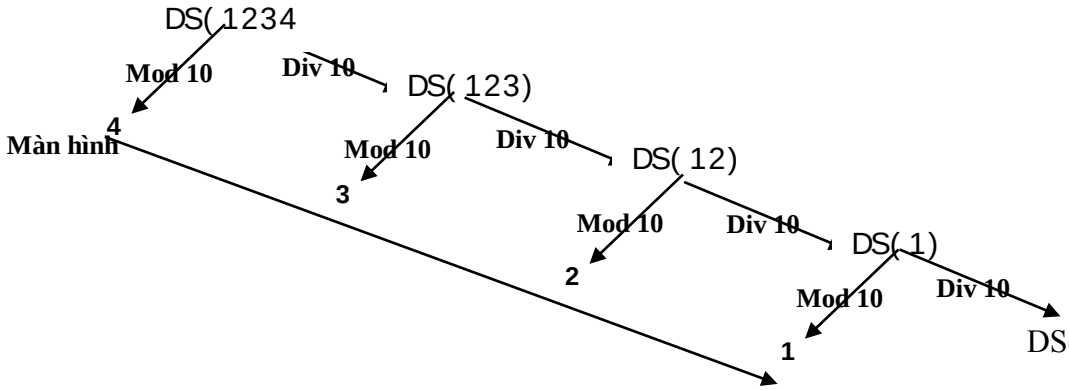
$$dem(n) = \begin{cases} 0 & \text{nếu } n = 0 \\ 1+dem(n/10) & \text{nếu } n > 0 \end{cases}$$

Hàm đệ quy như sau:

```
int  dem(long n)
{
 if(n==0)  return 0;
 else    return (1+ dem(n/10));
```

} ***//gọi đệ quy dem(n/10) để bớt đi số hàng đơn vị***

Ví dụ 4: Xuất đảo ngược một số nguyên dương ra màn hình



suy biến

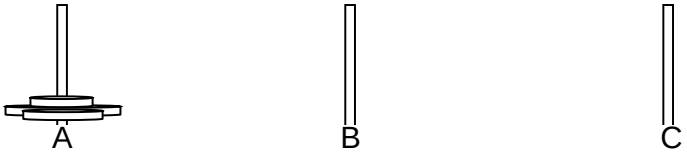
Giải thuật:

```
void xuat_dao_so(int n)
{ if (n >0)
  {cout<< n % 10;
   xuat_dao_so(n / 10);       //gọi đệ quy xuat_dao_so(n/10) để bớt đi số  hàng đơn vị
  }
}
```

Ví dụ 5: Bài toán tháp Hà Nội

Bài toán được phát biểu như sau:

Có ba cọc A,B,C. Khởi đầu cọc A có một số đĩa xếp theo thứ tự nhỏ dần lên trên đỉnh. Bài toán đặt ra là phải chuyển toàn bộ chồng đĩa từ A sang C. Mỗi lần thực hiện chuyển một đĩa từ một cọc sang một cọc khác và không được đặt đĩa lớn nằm trên đĩa nhỏ



Phân tích bài toán:

Trường hợp 1 đĩa: Chuyển thẳng từ A sang C. Đây là trường hợp suy biến

Trường hợp 2 đĩa: Chuyển 1 đĩa từ A sang B

Chuyển 1 đĩa từ A sang C

Chuyển 1 đĩa từ B sang C

Trường hợp chung n>1 đĩa; coi n-1 đĩa trên như là 1 đĩa và áp dụng trong trường hợp 2 đĩa

- Chuyển n-1 đĩa từ A sang B(dùng cọc C làm trung gian)
- Chuyển 1 đĩa từ A sang C

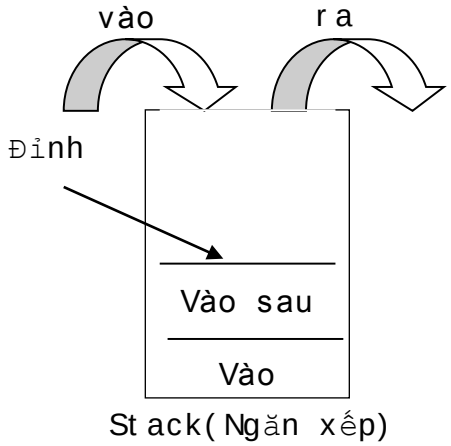
- Chuyển n-1 đĩa từ B sang C (dùng cọc A làm trung gian)

Giải thuật:

```
void ha_noi(int n, char A, char C, char B) // Chuyển n đĩa từ cọc A sang cọc C
{ if (n>0)
{       ha_noi(n-1, A, B, C);           //chuyển n-1 đĩa từ A  B
        //chuyển 1 đĩa từ A  C
        ha_noi(n-1, B, C, A);           //chuyển n-1 đĩa từ B  C
  }
}
```

4.2.2.4. Cơ chế làm việc khi gọi đệ qui

Như chúng ta đã biết, sự thực hiện của đệ quy chính là sự thực hiện lại chính bản thân nó với dữ liệu nhỏ hơn, nên khi gọi hàm đệ quy thì máy sẽ cấp phát một vùng nhớ có cơ chế hoạt động như Stack (ngăn xếp) cho hàm đệ quy đó. Cơ chế này hoạt động như kiểu LIFO(last in first out – vào sau ra trước), tức là việc thêm vào hay lấy ra đều thực hiện thông qua một đầu ra duy nhất



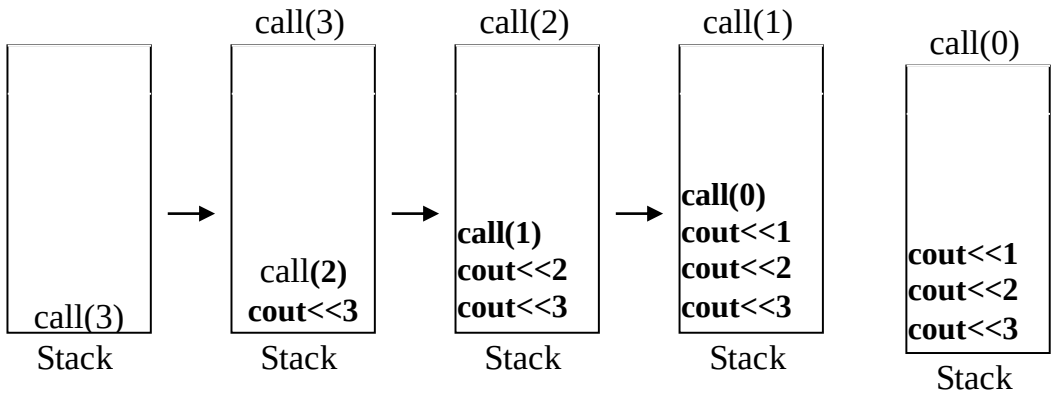
Khi hàm đệ quy được gọi lại thì những lệnh sau hàm đệ quy chưa thực hiện sẽ được đưa vào Stack, cứ như thế các lệnh của các lời gọi tiếp theo cũng được đưa vào Stack cho đến khi gặp trường hợp suy biến thì lần lượt các câu lệnh trong Stack được lấy ra thực hiện theo cơ chế LIFO cho đến khi Stack rỗng

Cho đoạn chương trình sau:

```
void call(int n)
{
  if(n>0)
  {
        call(n-1); //lời gọi đệ quy
        cout<<n<<"\n";
}
}
void    main()
```

```
{
  int n = 3;
  clrscr();
  call(n);
  getch();
}
```

Cơ chế làm việc khi gọi đệ quy được minh hoạ với hình ảnh dưới đây:



Kết quả trên màn hình như sau:

```
1
2
3
```

Như vậy với cơ chế hoạt động theo kiểu stack thì rõ ràng câu lệnh cout<<1 nằm trên đỉnh của stack nên được lấy ra thực hiện trước rồi lần lượt đến câu lệnh cout<<2 và cuối cùng là câu lệnh cout<<3 được thực hiện cho đến khi stack rỗng như kết quả trên màn hình trên.

4.2.2.5. Phân loại đệ quy

Có 2 loại đệ quy: đệ quy trực tiếp và đệ quy hổ tương

Đệ qui trực tiếp: P gọi P'

Ví dụ 1:

```
long GiaiThua(int n)
{
if (n==0) return 1;
else return n*GiaiThua(n-1);
```

```
}
```

Đệ qui gián tiếp: P gọi Q và Q gọi P'

Ví dụ 2:

```
void  Ong(int n);
void Ba(int n)
{
cout<<n<<"  Ba";
if (n>0) Ong(n-1);          //gọi hàm Ong(n-1)
}
void  Ong(int n)
{
cout<<n<<" Ong';
if (n>0) Ba(n-1);           //gọi hàm Ba(n-1)
}
void main()                 //chương trình chính
{  Ong(3);                  //gọi hàm Ong(3);
}
```

Màn hình kết quả:

```
3  Ong
2  Ba
1  Ong
0  Ba
```

4.3. MACRO

Macro là một tập hợp văn bản, câu lệnh và biểu thức được đại diện bằng một tên. Macro còn được gọi là lệnh gộp.Định nghĩa và sử dụng Macro làm cho chương trình ngắn gọn, rõ ràng và sáng sủa như việc định nghĩa và sử dụng hàm.

#define MAX 100 **void main( ) {** **int x, y=5;** **x= y*MAX;** **cout << "x=" << x;** **getch( );** **}**	**#define F(X) ((X)*(X))** **void main( ) {** **float x=2, y=3;** **cout << F(x) << ",“ << F(x) + F(y);** **getch( );** **}** ➔ Thì bộ tiền xử lý sẽ biến đổi thành: cout << (2*2) << ",“ << (2*2) + (3*3);

4.4. BÀI TẬP

Làm lại các bài tập của bài 2 và bài 3 dưới dạng hàm.

BÀI 5: KIỂU DỮ LIỆU CÓ CẤU TRÚC & KIỂU CON TRỎ

Chúng ta đã có các kiểu int, float, char để lưu số, ký tự v.v Tuy nhiên, những ứng dụng thực tế như lưu trữ thông tin học sinh, lưu trữ một danh sách học sinh, lưu trữ một ma trận, lưu trữ cây nhị phân v.v thì C chưa cung cấp kiểu, vì vậy lập trình viên phải tự xây dựng ra các kiểu mới này, đó chính là kiểu có cấu trúc.

MẢNG

Mảng 1 Chiều

Kiểu Tên[Số phần tử];	**int m[3]**; //khai báo mảng m kiểu int gồm có 3 phần tử
	int m[3]= {1,2,3}; //m[0]=1, m[1]=2, m[2]=3
	cout << m[0] << ",' << m[1] << "," m[2]
	char m[3]= {'a', 'b'}; //m[0]='a', m[1]='b', m[2] rỗng

Một số thao trên mảng 1 chiều

Nhập mảng:

```
int m[100], n;
cout << "n= "; cin >> n;
//phần tử mảng do người dùng tự nhập
for (int i=0; i<n; i++) {
  cout << "m[" << i << "]= "; cin >> m[i];
}
//phần tử mảng do tạo ngẫu nhiên
include <stdlib.h>
for (int i=0; i<n; i++) m[i]= random(1000);
```

Ví dụ: Nhập các giá trị nguyên dương cho một mảng nguyên M. Quá trình nhập kết thúc khi nhập giá trị âm hoặc bằng 0.

```
void Nhap_Mang(int iM[], int &iSize) {
  iSize=0;
  do {
          int iX;
          cout<<" nhap gia tri thu "<<iSize<<" :";  cin>>iX;
          if(iX>0) {
            iSize++;
   iM[iSize-1]=iX;
}
  } while (iX>0);
}
```

Duyệt mảng:

```
//in ra từng phần tử trong mảng
for (int i=0; i<n; i++) cout << "\t" << m[i]
```

Xoá một phần tử khỏi mảng:

Xoá phần tử thứ k trong mảng M có Size phần tử (0□k□Size-1).

Được minh hoạ qua hình vẽ sau:

Trước khi xoá: Size=6

15	21	4	13	8	11	...	...	...	...

Sau khi xoá (giả sử k=2) Size =5

15	21	13	8	11	1	...	...	...	...

Đầu vào: Mảng M có Size phần tử và vị trí xoá k.

Đầu ra: Mảng M có Size phần tử.

Thuật toán:

+B1: dời tất cả các phần tử từ vị trí k cho đến n-1 lên 1 vị trí (dời về bên trái)

+B2: giảm kích thước mảng đi 1

```
void Xoa_Vi_Tri_K(int iM[], int &iSize, int iViTriK) {
  int iDem;
  for(iDem=iViTriK; iDem<iSize-1; iDem++) iM[iDem]=iM[iDem+1];
  iSize--;
}
```

Thêm một giá trị x vào nảng

Thêm một giá trị x vào vị trí k trong mảng M có Size phần tử (0□k□ Size). Được minh hoạ qua hình vẽ sau:

Trước khi chèn: với Size =6

15	21	4	13	8	11	...	...	...	...

Sau khi chèn (giả sử k=2, x=100) Size =7

15	21	100	4	13	8	11	...	...	...

Đầu vào: Mảng M có Size phần tử , vị trí chèn k, giá trị chèn x.

Đầu ra: Mảng M có Size +1 phần tử với giá trị chèn x nằm đúng vị trí k.

Thuật toán:

+B1: Dời tất cả các phần tử từ vị trí k cho đến Size -1 ra sau 1 phần tử (dời về bên phải)

+B2: Đặt giá trị x vào vị trí k

+B2: Tăng kích thước mảng đi 1

```
void Chen_Vao_Vi_Tri_K(int iM[], int &iSize, int iX, int iViTriK) {
  int iDem;
  for(iDem=iSize; iDem>iViTriK; iDem--) iM[iDem]=iM[iDem-1];
  iM[iViTriK]=iX;
  iSize++;
}
```

Sắp xếp mảng

Sử dụng 2 vòng lặp sau để xắp xếp mảng theo điều kiện nào đó

```
int iDem1, iDem2;
for (iDem1=0; iDem1<iSize-1; iDem1++)
  for(iDem2=iDem1+1; iDem2<iSize; iDem2++)
  // phần tử iDem1 trước phần tử iDem2
     if(<vi phạm điều kiện sau sắp xếp giữa iM[iDem1] và iM[iDem2]>)
        Hoan_Vi(iM[iDem1], iM[iDem2]);
```

Tìm kiếm trên mảng

Tìm kiếm 1 giá trị x trên mảng số nguyên M có Size phần tử

Đầu vào: mảng M, Size phần tử và giá trị x.

Đầu ra: nếu có x trong mảng thì kết quả là vị trí đầu tiên của nó. Nếu không có thì kết quả là -

```
int Tim_X(int iM[], int iSize, int iX) {
  int iDem;
  for(iDem=0; iDem<iSize; iDem++)
  if (iM[iDem]==iX) return iDem;
  return -1;
}
```

Chú ý: Các thao tác trên sử dụng trên mảng số nguyên. Tuỳ theo từng yêu cầu bài toán cụ thể mà người sử dụng phải thay đổi kiểu dữ liệu và các yêu cầu cho phù hợp.

Mảng Nhiều Chiều

Kiểu Tên[Sốdòng][Sốcột];	// ma trận m kiểu int gồm 2 dòng, 3 cột, có 2x3 phần tử
	int m[2][3];
	//khai báo ma trận m có sẵn phần tử
	iMaTran [2][3]={{1,2,3},{4,5,6}};
	// m[0][0]=1, m[0][1]=2, m[0][2]=3
	// m[1][0]=4, m[1][1]=5, m[1][2]=6

Ví dụ : Viết chương trình nhập một ma trận Am*n từ bàn phím, sau đó xuất ma trận đó ra màn hình.

```
#include<stdio.h>
#include<iostream.h>
void main()
{
  int iCot, iHang, iDem1, iDem2, iMaTran[10][10];
```

```
    cout<<"Nhap so hang cua ma tran :";
    cin>>iHang;
    cout<<"Nhap so cot cua ma tran :";
    cin>>iCot;
    for(iDem1 = 0 ; iDem1<iHang ; iDem1 ++)
        for(iDem2 = 0 ; iDem2<iCot ; iDem2 ++)
        {
            cout<<"Nhap so phan tu [ "<<iDem1+1<<","<<iDem2+1<<" ]:";
            cin>>iMaTran[iDem1][iDem2];
        }
    cout<<"Noi dung ma tran \n";
    for(iDem1 = 0 ; iDem1<iHang ; iDem1 ++)
    {
        for(iDem2 = 0 ; iDem2<iCot ; iDem2 ++) {
            cout.width(4);
            cout<<iMaTran[iDem1][iDem2];
           }
        cout<<"\n";
    }
}
```

5.1. XÂU KÍ TỰ (HẰNG DÃY KÝ TỰ)

Kiểu Tên[độ dài xâu;	//khai báo xâu ký tự có độ dài 6 ký tự
	char strChuoi[6];
	//khai báo xâu ký tự
	strChuoi[6]=”HELLO”;
	char *strChuoi=”HELLO”;
	char strChuoi[]=”HELLO”;

Bộ nhớ máy tính sẽ cung cấp 6 byte để lưu trữ xâu strChuoi

‘ H’	‘ E’	‘ L’	‘ L’	‘ L’	‘ \ 0’

H
SF

Thông thường, do chưa xác định trước độ dài của xâu kí tự, cho nên người ta thường hay sử dụng mảng động (cách 2) để khai báo xâu kí tự.

Các thao tác nhập xuất trên xâu ký tự:

Nhập xâu ký tự từ bàn phím

Cách 1: sử dụng toán tử cin>>. Cú pháp cin>>biến_xâu; . Trong trường hợp này không nhập ký tự trắng

Cách 2: sử dụng hàm gets. Cú pháp gets(biến_xâu);

Cách 3: sử dụng cin.getline. cú pháp cin.getline(biến_xâu,n); với n số ký tự tối đa cần nhập

Xuất xâu ký tự được lưu trữ trong biến_xâu ra màn hình tại vị trí con trỏ màn hình

Cách 1: sử dụng toán tử cout<<: cú pháp cout<<biến_xâu ;

Cách 2: sử dụng hàm puts: cú pháp puts(biến_xâu);

Cả 2 hàm gets và puts đều nằm trong thư viện <stdio.h>

Một số hàm chuẩn thường sử dụng xử lý xâu ký tự trong C (thuộc thư viện <string.h>).

Hàm Ý nghĩa

strcpy(S1, S2) Chép xâu S2 vào xâu S1

strcat(S1, S2) Trả về xâu S1 mới có độ dài bằng S1 cũ nối thêm S2

strlen(S1) Trả về chiều dài xâu S1

strcmp(S1, S2) So sánh hai xâu S1 với S2, kết quả trả về:

- Bằng 0 nếu S1 bằng S2.
- Lớn hơn 0 nếu S1>S2.
- Nhỏ hơn 0 nếu S1<S2

strchr(S1, ch) Trả về vị trí lần gặp thứ nhất của kí tự ch trong xâu S1

strstr(S1, S2) Trả về vị trí lần gặp thứ nhất của xâu S2 trong xâu S1

5.2. CÁC KIỂU DỮ LIỆU TỰ ĐỊNH NGHĨA (ENUM, UNION, STRUCT)

5.2.1. Enum

Kiểu này được dùng để liệt kê các hằng có thể được sử dụng.

Xây dựng: cú pháp enum <tên_kiểu { pt_1[=gt_1], pt_2[=gt_2],...,pt_n=[gt_n]};>

Trong đó:

Tên_kiểu là kiểu dữ liệu mới được dùng để khai báo biến

pt_i (i=1,n) là các tên khác nhau từng đôi một (theo nguyên tắc đặt tên)

gt_i (i=1,n) phải là các giá trị nguyên. Nêu không gán giá trị khi định nghĩa kiểu, thì giá trị của phần tử đó sẽ là giá trị của phần tử trước nó + Phần tử đầu tiên mặc định giá trị 0.

Ví dụ: Nếu ta quy định chủ nhật là ngày thứ nhất trong tuần, thì có thể định nghĩa kiểu THU (thứ) như sau:

enum Thu{CHU_NHAT=1, THU_HAI, THU_BA, THU_TU, THU_NAM, THU_SAU, THU_BAY};

hoặc

enum Thu{CHUNHAT=1, THU_HAI=2, THU_BA=3, THU_TU=4, THU_NAM=5, THU_SAU=6, THU_BAY=7};

Sử dụng kiểu enum

Khai báo biến bình thường cho kiểu enum. Ví dụ khai báo Thu thuNgay; thì thuNgay sẽ có giá trị từ .7. ví dụ lệnh thuNgay=THU_TU thì thứ ngày có giá trị là 4

5.2.2. Union

Cấu trúc của Union gồm nhiều thành phần trong đó các thành phần có cùng chung một vùng nhớ. Kích thước của kiểu Union bằng chính kích thước của kiểu dữ liệu lớn nhất trong đó.

Định nghĩa kiểu union:

union Tên_kiểu{

[Khai báo các thành phần của union]

};

Ví dụ: định nghĩa kiểu KieuChung có thể lưu trữ số nguyên, số thực và ký tự tuỳ theo thời điểm sử dụng.

```
union KieuChungType{
  int i;
  float f;
  char c;
};
```

Kiểu dữ liệu KieuChung trên có kích thước là 4 byte của số thực.

Khai báo biến kiểu union: bình thường như mọi kiểu dữ liệu khác

Truy cập từng thành của union: tên_biến.tên_thành_phần. Tuỳ theo từng thành phần của union mà trình dịch đối xử thành phần đó theo kiểu dữ liệu tương ứng như khi định nghĩa.

Nội dung biến này luôn giữ giá trị cuối cùng nhập (hoặc gán) vào cho biến.

Ví dụ: với khai báo kiểu union như trên, ta có thể khai báo biến x và sử dụng như sau:

```
KieuChungType x;
cout<<" nhap vao so thuc:"; cin>>x.f;
cout<<" nhap ky tu :";  cin>>x.c;
cout<<" nhap vao so nguyen:"; cin>>x.i;
cout<<x.f;              //Không xác định
cout<<x.i +10;          //Kết quả là giá trị nhập cuối cùng + 10
```

5.2.3. Struct

Từ những kiểu dữ liệu đã có (cơ sở hoặc kiểu có cấu trúc) ngôn ngữ cho phép xây dựng nên một kiểu dữ liệu mới cho phù hợp với mỗi bài toán cụ thể. Kiểu này được gọi là kiểu struct (hay kiểu bản ghi)

Định nghĩa kiểu struct

struct kiểu_struct{

  kiểu_đã_có_1: danh sách các trường cùng kiểu;

  kiểu_đã_có_2: danh sách các trường cùng kiểu;

  ...

  kiểu_đã_có_n: danh sách các trường cùng kiểu;

};

Ví dụ: định nghĩa kiểu HocSinhType gồm các thông tin sau: họ tên, ngày tháng năm sinh, giới tính, điểm.

```
struct NgayType{ int iday, imonth, iyear;};
struct HocSinhType {
  char cHoTen[30];
  NgayType ngaySinh;
  int iGioiTinh;                //0:nam, 1:nữ
  float fDiem;
};
```

Khai báo biến kiểu struct: kiểu_struct biến;

Ví dụ khai báo biến hs1, hs2 kiểu HocSinhType: HocSinhType hs1, hs2;

Truy cập vào từng trường của cấu trúc: biến_cấu_trúc.tên_trường

Ví dụ: viết chương trình nhập vào 3 điểm A, B và C trong mặt phẳng Oxy. Sau đó tính diện tích và chu vi của tam giác ABC

```
#include<conio.h>
#include<iostream.h>
#include<math.h>
//khai báo kiểu dữ liệu điểm
struct DiemType {
  int iHoanhDo, iTungDo;
};
//Hàm nhập dữ liệu cho 1 điểm
void Nhap_Diem(DiemType & p)  {
  cout<<"\n\tNhap hoanh do:"; cin>>p.iHoanhDo;
  cout<<"\tNhap tung do:"; cin>>p.iTungDo;
}
//Hàm tính khoảng cách giữa 2 điểm
float Khoang_Cach(DiemType a, DiemType b) {
 return sqrt(pow(a.iHoanhDo-b.iHoanhDo,2)+pow(a.iTungDo-b.iTungDo,2));} //
```

$\sqrt{(x_A-x_B)^2+(y_A-y_B)^2}$

```
//Hàm tính chu vi tam giác
float Chu_Vi(DiemType x, DiemType y, DiemType z) {
  float fKhoangCach1, fKhoangCach2, fKhoangCach3;
  fKhoangCach1=Khoang_Cach(x,y);
  fKhoangCach2=Khoang_Cach(x,z);
  fKhoangCach3=Khoang_Cach(z,y);
  return fKhoangCach1+fKhoangCach2+fKhoangCach3;
}
//Hàm tính diện tích tam giác
float Dien_Tich(DiemType x, DiemType y, DiemType z) {
  float fKhoangCach1, fKhoangCach2, fKhoangCach3, fChuVi
```

```
  fKhoangCach1=Khoang_Cach(X,Y);
  fKhoangCach2=Khoang_Cach(X,Z);
  fKhoangCach3=Khoang_Cach(Z,Y);
  fChuVi=(fA+fB+fC)/2;
 return sqrt(fChuVi*(fChuVi-fKhoangCach1)*(fChuVi-fKhoangCach2)*(fChuVi-
fKhoangCach3)); //√P(P-a)(P-b)(P-c)
 }
//Hàm main
 void main() {
  DiemType a, b, c;
  cout<<" Nhap diem a:";
  Nhap_Diem(a);      //gọi hàm Nhap_Diem để nhập điểm a
  cout<<" Nhap diem b:";
  Nhap_Diem(b);
  cout<<" Nhap diem c:";
  Nhap_Diem(c);
  cout<<"Chu vi tam giac ABC:"<<Chu_Vi(a,b,c);     //gọi hàm Chu_Vi
  cout<<"\nDien tich tam giac ABC :"<<Dien_Tich(a,b,c); //gọi hàm Dien_Tich
 }
```

Màn hình kết quả của chương trình như sau:

```
Nhap diem A:
     Nhap hoanh do: 1 ↵
     Nhap Tung do: 1↵
Nhap diem B:
     Nhap hoanh do: 1 ↵
     Nhap Tung do: 4↵
Nhap diem C:
     Nhap hoanh do: 5 ↵
     Nhap Tung do: 1↵
Chu vi tam giac ABC: 12
Dien tich tam giac ABC: 6
```

5.3. Kiểu con trỏ

5.3.1. Khái Niệm

Một biến con trỏ là một biến mà nội dung chỉ chứa địa chỉ của một biến khác. Nhờ có biến con trỏ mà ta có thể sử dụng biến thông qua địa chỉ của nó trong bộ nhớ

5.3.2. Khai Báo Biến Con Trỏ

Cú pháp: kiểu_dữ_liệu *biến_con_trỏ;

Ví dụ:

```
int *iptr;      // khai báo iptr là biến con trỏ kiểu số nguyên
float *fptr;    // khai báo fptr là biến con trỏ kiểu số thực
```

Khi khai báo biến con trỏ chứa giá trị NULL. NULL là một hằng đặc biệt của kiểu con trỏ. Khi con trỏ được gán giá trị NULL thì nó không trỏ đến một biến nào cả.

5.3.3. Các Phép Toán Trên Biến Con Trỏ

- Lấy địa chỉ của biến khác: cú pháp biến_con_trỏ=&biến_cùng_kiểu;

Ví dụ:

```
int iX, *iptr, iY;
float fF, fR,*fptr;
iptr=&iX;  //con trỏ iptr lấy địa chỉ của biến iX. Ta nói con trỏ nguyên iptr chứa địa chỉ biến iX
fptr=&fF; //con trỏ fptr lấy địa chỉ của biến fF. ta nói con trỏ thực fptr chứa địa chỉ biến fF
```

- Truy cập vào vùng nhớ do biến con trỏ trỏ đến : cú pháp *biến_con_trỏ

Ví dụ : từ ví dụ trên ta có thể gán giá trị 100 cho vùng nhớ do biến con trỏ iptr trỏ đến là ***iptr=100;** lệnh này tương đương với iX=100;

- Phép so sánh trên biến con trỏ:

Phép so sánh bằng (biến_con_trỏ==biến_con_trỏ) để xem 2 biến con trỏ có cùng chứa địa chỉ của một biến khác.

Phép so sánh khác (biến_con_trỏ != biến_con_trỏ) để xem 2 biến con trỏ chứa địa chỉ của hai biến khác.

- Phép cộng một số nguyên i vào biến con trỏ tức là trỏ tới địa chỉ của ô nhớ cách i phần tử tính từ địa chỉ biến con trỏ đang trỏ

Ví dụ:

```
int *iptr;
++iptr;
```

iptr là biến con trỏ kiểu số nguyên , khi iptr tăng lên 1, nghĩa là sẽ tăng 2 byte. Chẳng hạn, nếu iptr đang trỏ địa chỉ 100 thì sau khi thực hiện ++ iptr thì iptr sẽ trỏ đến địa chỉ 102

```
char *cptr;
cptr=cptr+4;
```

cptr là biến con trỏ kiểu char, khi cptr tăng lên 4, nghĩa là sẽ tăng 4 byte. Chẳng hạn, nếu cptr đang trỏ địa chỉ 100 thì sau khi thực hiện lệnh cộng thì cptr sẽ trỏ đến địa chỉ 104

5.3.4. Con trỏ kiểu void

Con trỏ kiểu void là con trỏ đặc biệt, nó có thể chứa bất kỳ địa chỉ của kiểu dữ liệu nào. Nó được khai báo như sau: void *biến_con_trỏ;

Ví du:

```
int iX, iY;
float fF, fR;
void *vptr;
vptr=&iX; hoặc vptr=&fF;
```

5.3.5. Con Trỏ Và Biến Động

Các biến mà ta sử dụng trên được gọi là biến tĩnh, vì chúng được xác định một cách rõ ràng trong các câu lệnh khai báo biến. Sau đó chúng được truy cập thông qua tên của chúng. Thời gian tồn tại của biến này là thời gian tồn tại của khối lệnh trong chương trình chứa câu lệnh khai báo của biến đó.

Tuy nhiên có loại biến có thể được tạo ra và xoá đi trong khi đang thực hiện chương trình, tuỳ theo nhu cầu. Các biến này hoàn toàn không xác định được từ trước, những biến này được gọi là biến động. Các biến động không có tên, việc tạo ra, huỹ và sử dụng chúng thông qua biến con trỏ với toán tử new, delete và thao tác trên biến con trỏ.

Cấp phát bộ nhớ cho biến con trỏ: biến_con_trỏ=new kiểu; với new là từ khoá, kiểu_dữ_liệu là kiểu mà biến_con_trỏ sẽ trỏ đến (kiểu dữ liệu trỏ đến khi khai báo). Toán tử new cấp phát một vùng ở bộ nhớ bằng chính kích thước của kiểu_dữ_liệu đứng sau từ khoá new, vùng nhớ này nằm ở vùng nhớ heap và địa chỉ của vùng nhớ này được gán cho biến_con_trỏ

Huỹ (giải phóng) vùng nhớ đã được cấp phát cho biến con trỏ: delete biến_con_trỏ;

5.3.6. Mối Liên Hệ Giữa Con Trỏ Và Mảng

Nhằm tránh gây lãng phí bộ nhớ do thông thường khai báo mảng tĩnh lớn nhưng thực tế sử dụng ít hơn nhiều, nên thông thường ta thường sử dụng con trỏ để khai báo mảng, mảng này được gọi là mảng động. Việc sử dụng mảng động làm cho việc quản lý bộ nhớ được tối ưu hơn do khi cần thì xin cấp phát vùng nhớ để sử dụng, khi không cần sử dụng thì trả lại bộ nhớ cho chương trình.

- Cú pháp khai báo mảng động: Kiểu_dữ _liệu *Tên_mảng;
- Khi đã xác định được số phần tử cần sử dụng thì xin cấp phát vùng nhớ cho biến mảng theo cú pháp : tên_mảng= new Kiểu_dữ_liệu [Số_phần_tử]; với *new* là toán tử cấp phát bộ nhớ động.
- Khi không cần sử dụng mảng và trả lại vùng nhớ cho chương trình thì sử dụng cú pháp sau: delete(tên_mảng); hoặc free(tên _mảng);

Ví dụ: khai báo mảng số nguyên bằng con trỏ và cấp phát cho mảng 6 phần tử

int *iM; iM=new int[6];

hoặc int *iM=new int[6];

Truy cập từng phần tử mảng thông qua biến con trỏ

Giả sử có mảng số nguyên được khai báo và cấp phát như sau : int *iM=new int[iSize]; (iSize là một hằng đã có trước)

Nhờ phép cộng một giá trị nguyên vào biến con trỏ nên ta có thể truy cập từng phần tử của mảng như sau:

*iM iM[0]

*(iM +1)□ iM[1]

...

*(iM+iDem) □ iM[iDem] (0□iDem□iSize-1)

Ví dụ: Viết hàm xuất nội dung các số chẵn trong mảng số nguyên M ra màn hình:

```
void Xuat_Mang(int *iM, int iSize)
{ int iDem;
cout<<"\n Cac so chan trong mang la:";
for(iDem=0; iDem<iSize ; iDem++)
  if (*(iM+iDem)%2==0) cout<<*(iM+iDem);
}
```

Chú ý:

▪ Trong trường hợp sử dụng con trỏ để khai báo xâu ký tự thì char *biến_xâu; Hoặc vừa khai báo vừa khởi tạo char *biến_xâu="nội dung chuỗi";

▪ Trong trường hợp khai báo mảng 2 chiều thì kiểu_dữ_liệu **Tên_ma_trận ; Còn cấp phát bộ nhớ cho mảng 2 chiều được thể hiện qua ví dụ sau: Viết chương trình nhập một ma trận A_{m*n} từ bàn phím, sau đó xuất ma trận đó ra màn hình

```
#include<iostream.h>
void main()
{ int iCot, iHang, iDem1, iDem2, **iMaTran;
  cout<<"Nhap so hang cua ma tran :";      cin>>iHang;
  cout<<"Nhap so cot cua ma tran :";       cin>>iCot;
  // xin cấp phát ma trận
  *iMaTran=new int[iHang];
  for(iDem1=0; iDem1<iHang; iDem1++) iMaTran[iDem]=new int[iCot];
  …// nhập và xuất ma trận như ví dụ phần 5.2.3 trang 59
  }
```

BÀI TẬP

Bài 1: Viết chương trình nhập một mảng số nguyên có n phần tử với n nhập từ bàn phím. Sau đó xuất nội dung của mảng ra màn hình.

```
Nhap vao so phan tu cua mang: 5↵
Mang[1] = 4↵
Mang[2] = 3↵
Mang[3] = 8↵
Mang[4] = 5↵
Mang [5] = 1↵
Mang vua nhap la: 4   3   8   5   1_
```

Chương trình mẫu như sau:

```
#include<conio.h>
#include<iostream.h>
//Hàm nhập mảng số nguyên có k phần tử từ bàn phím
void NhapMang(int iA[],  int &iK)
{ cout<<"Nhap vao so phan tu cua mang";
  cin>>iK;
  for(int iDem=1; iDem<=iK; iDem++)
  {       cout<<"Mang["<<iDem<<"]=";
          cin>>iA[iDem];
  }
}
// Xuất nội dung mảng số nguyên A ra màn hình
void XuatMang(int iA[],  int iK)
{  cout<<"\n';
   for(int iDem=1; iDem<=iK; iDem++)
          cout<<iA[iDem]<<"   ";
}
void main()
{    int   iMang[100];
  int  iN;                        //iN là số phần tử của mảng
  clrscr();
  //Nhập mảng từ bàn phím
     NhapMang(iMang,iN);
  //Xuat cac gia tri trong mang ra man hinh
   cout<<"\nMang vua nhap la:";
   XuatMang(iMang,iN);
  getch();
}
```

Dựa vào bài mẫu có sẵn trên, hãy viết tiếp chương trình để thực hiện các yêu cầu sau:

1. Xuất các giá trị chẵn ra màn hình.
2. Tính tổng các số chia hết cho 2 và cho 5 trong mảng.
3. Tính trung bình cộng các số lẻ trong mảng.

4. Loại phần tử ở vị trí bất kỳ trong mảng.

Bài 2: Viết chương trình nhập một mảng số nguyên có n phần tử với n nhập từ bàn phím. Sau đó xuất nội dung của mảng ra màn hình. Thực hiện các yêu cầu sau:

1. Chèn một giá trị x và vị trí bất kỳ trong mảng.
2. Sắp xếp mảng theo thứ tự tăng dần.
3. Loại tất cả các giá trị âm trong mảng.
4. Tìm số nguyên tố lớn nhất trong dãy
5. Tìm 2 phần tử trong mảng có tổng lớn nhất.

Ví dụ:

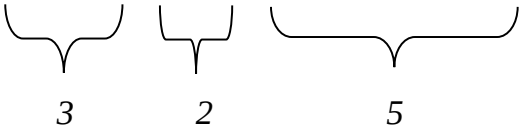
Mảng có các phần tử: 6 2 7 9 4 9

 2 phần tử trong mảng có tổng lớn nhất là 18

Bài 3: Viết chương trình nhập một mảng số nguyên có n phần tử với n nhập từ bàn phím. Hãy tìm dãy con tăng dần dài nhất

Ví dụ:

Mảng : 2 5 7 1 4 3 8 9 10 23 20



3 2 5

 Mảng có 3 mảng con tăng dần, nhưng mảng con tăng dần dài nhất là:

3 8 9 10 23 có độ dài là 5

Bài 4: Viết chương trình nhập một mảng số nguyên có n phần tử (với n nhập từ bàn phím). Hãy tìm mảng con lập thành cấp số cộng dài nhất.

Ví dụ:

Mảng : 2 5 8 1 3 5 7 9 4 10

3 5 2

 Mảng có 3 mảng con lập thành cấp số cộng. Mảng con 2 5 8 lập thành cấp số cộng với công sai là 3 và có độ dài là 3. Mảng con 1 3 5 7 9 lập thành cấp số cộng với công sai là là 2 và có độ dài là 5. Mảng con 4 6 lập thành cấp số cộngvới công sai là 2 và có độ dài là 2 Vậy mảng con lập thành cấp số cộng dài nhất là 1 3 5 7 9

Bài 5: Viết chương trình nhập một mảng số nguyên có n phần tử (với n nhập từ bàn phím). Hãy loại các phần tử trùng nhau trong mảng.

Hướng dẫn: tại vị trí i, tìm xem trong mảng từ vị trí 0 đến i-1 có giá trị tại vị trí i không ? nếu có thì xoá giá trị ở vị trí i ra khỏi mảng.

Bài 6: Viết chương trình nhập một mảng số nguyên M có n phần tử với n nhập từ bàn phím. Hãy sắp xếp mảng tăng dần. Sau đó chèn tăng một giá trị vào mảng (không sắp xếp lại mảng).

Bài 7: Viết chương trình nhập một mảng số nguyên M có n phần tử (với n nhập từ bàn phím). Hãy loại các giá trị x trong mảng ra khỏi mảng

Bài 8: Viết chương trình nhập một mảng số nguyên M có n phần tử (với n nhập từ bàn phím). Hãy tìm số nguyên nhỏ nhất không có trong mảng

Bài 9: Tạo một mảng A có n số thực, tìm trung bình cộng các số âm trong mảng.

Nhập vào mảng n số thực A, hãy tính tổng $a_1-a_2+a_3-a_4+\ldots$

Bài 10: Viết chương trình nhập 2 mảng số nguyên M_1, M_2. Mỗi mảng có n phần tử, với n nhập từ bàn phím. Sau đó thực hiện các yêu cầu sau:

1. Tính $M_3 = M_1 + M_2$ (Bằng cách cộng các phần tử tương ứng của 2 mảng với nhau và xuất M_3 ra màn hình)
2. Xuất M_3 ra màn hình chẵn trước lẻ sau.
3. sắp xếp M_1, M_2 tăng dần
4. Trộn tăng M_1 và M_2 thành M_3. Xuất M_3 ra màn hình

Bài 11: Nhập vào một xâu ký tự, đếm xem có bao nhiêu chữ số, bao nhiêu nguyên âm, bao nhiêu phụ âm, bao nhiêu ký tự đặc biệt (các ký tự còn lại)

Bài 12: Nhập vào xâu ký tự, đưa xâu ký tự về dạng tiêu đề (ví dụ: "Đại Học Duy Tân")

Bài 13: Viết chương trình nhập một xâu ký tự bất kỳ, sau đó loại các ký tự trắng thừa trong xâu

Ví dụ: chuỗi vào:" Dai hoc Duy tan "

chuỗi kết quả: "Dai hoc Duy tan"

Bài 14: Nhập vào xâu ký tự, thay thế các ký tự giống nhau và đi liền nhau bằng 1 ký tự (abcbbbcca->abcbca)

Bài 15: Viết chương trình nhập một xâu ký tự và xuất ra màn hình theo thứ tự ngược lại.

Bài 16: Nhập vào xâu ký tự, đếm xem trong xâu có bao nhiêu từ

Bài 17: Để lưu trữ và xử lý phân số, người ta thường định nghĩa kiểu dữ liệu *phân số.* Hãy viết chương trình thực hiện các phép toán trên đó. Yêu cầu:

a. Định nghĩa kiểu dữ liệu phân số.
b. Viết chương trình con nhập phân số.
c. Viết chương trình con xuất phân số.
d. Viết các chương trình con thực hiện các phép tính trên phân số.

Sau đó viết chương trình chính để được màn hình mô phỏng như sau:

```
Nhap phan so thu 1
Nhap tu=1↵
Nhap mau<>0: 2↵
Nhap phan so thu 2
Nhap tu=3↵
Nhap mau<>0: 4↵
Nhap phep toan(+,-,*,/):+↵
1/2 + 3/4=5/4
```

Chương trình mẫu như sau:

```
#include<conio.h>
#include<iostream.h>
struct   PhanSoType
  {
   int  iTu, iMau;
  };
void   NhapPhanSo(PhanSoType &P)
{
  cout<<"\nNhap tu=";
  cin>>P.iTu;
  do
  {
    cout<<"Nhap mau<>0:";
    cin>>P.iMau;
  }while(P.iMau==0);
}
void   XuatPhanSo(PhanSoType P)
{  if(P.iTu*P.iMau<0) cout<<"-";
  cout<<abs(P.iTu)<<"/"<<abs(P.iMau);
}
/*Tim uoc chung lon nhat cua 2 so nguyen a,b*/
int   TimUocChungLonNhat(int  iA, int iB)
{
iA=abs(iA);
iB=abs(iB);
   if(iA==0) return   iB;
 else
if(iB==0) return   iA;
else
{
```

```
                while(iA != iB)
                  if(iA > iB)
                        iA = iA - iB;
                  else  iB = iB - iA;
                return iA;
}
}
/*Hàm rút gọn một phân số*/
void   RutGonPhanSo(PhanSoType &P)
{
 int  iUocChungLonNhat = TimUocChungLonNhat(P.iTu,P.iMau);
 P.iTu=P.iTu/ iUocChungLonNhat;
 P.iMau=P.iMau/ iUocChungLonNhat;
}
PhanSoType   CongPhanSo(PhanSoType  P1, PhanSoType  P2)
{
 PhanSoType  P;
 P.iTu=P1.iTu*P2.iMau+P1.iMau*P2.iTu;
 P.iMau=P1.iMau*P2.iMau;
 RutGonPhanSo (P);
 return P;
}

PhanSoType   TruPhanSo (PhanSoType P1, PhanSoType P2)
{
 PhanSoType  P;
 P.iTu=P1. iTu*P2. iMau-P1. iMau*P2. iTu;
 P. iMau=P1. iMau*P2. iMau;
 RutGonPhanSo (P);
 return P;
}
PhanSoType NhanPhanSo (PhanSoType P1, PhanSoType P2)
{
 PhanSoType P;
 P. iTu=P1. iTu*P2. iTu;
 P. iMau=P1. iMau*P2. iMau;
 RutGonPhanSo(P);
 return  P;
}
PhanSoType ChiaPhanSo (PhanSoType P1, PhanSoType P2)
{
 PhanSoType P;
 if(P2. iTu==0) cout<<"\nKhong chia duoc";
 else
```

```
  {
    P. iTu=P1. iTu*P2. iMau;
    P. iMau=P1. iMau*P2. iTu;
    RutGonPhanSo (P);
   return P;}
}
void main()
{
  clrscr();
  PhanSoType   P1,P2,P3;
 char  cPhepToan;
  cout<<"Nhap phan so thu 1";
  NhapPhanSo(P1);
  cout<<"Nhap phan so thu 2";
  NhapPhanSo (P2);
 cout<<"Nhap phep toan(+, -, *, /):";
 cin>>chrPhepToan;
switch(cPhepToan)
{  case '+':      P3=CongPhanSo(P1,P2); break;
   case '-':      P3=TruPhanSo(P1,P2); break;
   case '*':      P3=NhanPhanSo(P1,P2); break;
   case '/':      if (P2.iTu==0)
                        {cout<<" Khong chia cho 0"; exit(0);}
                  else  {P3=ChiaPhanSo(P1,P2); break;}
  Default:        {cout<<"Khong phải phep toan"; exit(0); }
}
XuatPhanSo(P1); cout<<cPhepToan; XuatPhanSo(P2); cout<<"="; XuatPhanSo(P3);
  getch();
}
```

Bài 18:Nhập vào n phân số (hoặc tạo ngẫu nhiên). Sau đó tìm 2 phân số có tổng lớn nhất

Bài 19:Nhập vào n phân số (hoặc tạo ngẫu nhiên). Sau đó sắp xếp các phân số tăng dần

Bài 20: Số phức là số có dạng như sau A + iB. Với A là phần thực và B la phần áo và i được tính như sau : i^2=-1; i^3= -i; i^4=1

Ví dụ: Cho 2 số phức X = a +ib và Y = c +id

Z = X + Y = (a +ib) + (c +id) = (a +c) + i(b +d)

Z = X - Y = (a +ib) - (c +id) = (a -c) + i(b - d)

Z = X * Y = (a +ib) * (c +id) = ac + iad +ibc +i^2bd = (ac - bd) + i(ad + bc)

Z = X / Y = (a +ib) / (c +id) = [(a + ib) * (c - id)] / [(c + id) * (c - id)] = [(ac + bd) + i(bc - ad)] / (c^2 + d^2)

Yêu cầu:

10. Định nghĩa kiểu dữ liệu số phức.
11. Viết chương trình con nhập số phức.
12. Viết chương trình con xuất số phức dưới dạng A + iB hoặc A – iB.
13. Viết các thuật toán tính các phép tính trên (+, -, *, /) trên 2 số phức.

Sau đó viết chương trình chính nhập vào 2 số phức. Sau đó thực hiện các phép tính trên 2 số đó và xuất kết quả ra màn hình

Bài 21: Để xử lý một điểm trên mặt phẳng Oxy, người ta thường định nghĩa kiểu dữ liệu Diem *(điểm)*gồm 2 thành thần hoành độ và tung độ. Sau đó dựa vào kiểu dữ liệu mới này để nhập các thông tin cho một tam giác.

Yêu cầu:

1. Định nghĩa kiểu dữ liệu Diem có 2 phần hoành độ và tung độ.
2. Viết chương trình con nhập một điểm.
3. Viết chương trình con xuất một điểm.
4. Viết chương trình con tính khoảng cách giữa 2 điểm.
5. Viết chương trình con tính diện tích của một tam giác (có 3 điểm).

Sau đó viết chương trình nhập vào 4 điểm A, B, C và D. Hãy kiểm tra xem điểm D có nằm trong tam giác ABC hay không.

Bài 22: Viết chương trình theo các yêu cầu sau:

1. Định nghĩa kiểu dữ liệu Diem và khai báo một mảng chứa các điểm.
2. Viết chương trình con nhập 1 mảng điểm.
3. Viết chương trình con tìm khoảng cách giữa 2 điểm.
4. Viết chương trình con tìm khoảng cách lớn nhất giữa 2 điểm trong mảng điểm.
5. Viết chương trình con tìm tam giác chu vi lớn nhất được lấy từ trong mảng điểm.
6. Viết chương trình con tìm tam giác có diện tích lớn nhất trong mảng điểm.

Sau đó viết chương trình chính gọi các chương trình con trên.

Bài 23: Để xử lý một đa thức 1 biến $f(x) = a_nx^n + a_{n-1}x^{n-1} + \ldots + a_1x^1 + a_0$ thì ta xây dựng một cấu trúc có hệ số, số mũ để lưu trữ 1 phần tử của đa thức và dùng mảng để lưu trữ

Yêu cầu:

1. Xây dựng một cấu trúc đa thức và khai báo một mảng chứa đa thức.
2. Nhập đa thức cho mảng.
3. Đưa đa thức về dạng chuẩn (sắp xếp theo số mũ giảm dần).
4. Tính giá trị của đa thức $f(x_0)$ với x_0 được nhập từ bàn phím
5. Tính giá trị đạo hàm ($f'(x_0)$)của đa thức $f(x_0)$ với x_0 được nhập từ bàn phím

Bài 24: Tìm ước số chung lớn nhất của 2 số nguyên dương bằng phân tích thừa số nguyên tố

Bài 25: Tìm bội số chung nhỏ nhất của 2 số nguyên dương bằng phân tích thừa số nguyên tố.

BÀI 6: KIỂU DỮ LIỆU FILE (FILE)

File giúp lập trình viên lưu lại dữ liệu của chương trình. File lưu trên bộ nhớ ngoài (HDD, SSD v.v). Thư viện stdio.h cung cấp các hàm xử lý file.

6.1. KHÁI NIỆM VỀ FILE

Có 02 loại file: văn bản và nhị phân.

Các bước xử lý file: khai báo con trỏ file, mở file để đọc hoặc ghi, ghi hoặc đọc trên file và đóng file.

Khai báo file:

FILE *con_trỏ_file	**FILE *f;**

6.2. FILE VĂN BẢN.

Ví dụ có file văn bản có nội dung sau:

TRUONG DAI HOC DUY TAN
KHOA CONG NGHE THONG TIN

File được lưu:

TRUONG DAI HOC DUY TAN	**CR LF**	KHOA CONG NGHE THONG TIN	**EOF**

- CR (ascii 13): xuống dòng.
- LF (ascii 10): về đầu dòng
- EOF (ascii 26): kết thúc file

6.2.1. Các Thao Tác Trên File (fopen, fclose, fcloseall, fllush, fllushall, ferror, feof).

Tất cả các hàm này dùng chung cho cả hai file văn bản và file nhị phân.

a) *Mở file:* fileptr= fopen (tên_file, kiểu_xử_lý);

Trong đó:

- Tên_file phải đúng theo nguyên tắc đặt tên của DOS, có thể chứa cả đường dẫn.
- Nếu mở file thành công thì fptr ≠NULL, còn mở file không được thì fptr=NULL.
- Kiểu_xử_lý là kiểu xử lý file sau khi mở:

Kiểu_xử_lý	**Ý NGHĨA**
"r" "rt"	Mở file văn bản đã tồn tại để ***đọc***
"w" "wt"	Mở file văn bản mới để ***ghi***. Nếu file đã tồn tại thì nó sẽ bị xoá để thay vào đó một file mới
"a" "at"	Mở file văn bản đã có để ***ghi*** thêm dữ liệu nối vào cuối file. Nếu file này chưa có thì file mới sẽ tạo ra
"r+" "r+t"	Mở file văn bản đã tồn tại cho phép cả ***đọc***, cả ***ghi***
"w+" "w+t"	Mở file văn bản mới cho cả ***ghi*** và ***đọc***. Nếu đã có file với tên file này thì nó sẽ bị huỷ để thay vào đó là một file mới
"a+" "a+t"	Mở file văn bản đã tồn tại hoặc tạo file mới để ***đọc*** và ***ghi*** thêm dữ liệu vào cuối file
"rb"	Mở nhị phân đã tồn tại để ***đọc***

"wb"	Mở file nhị phân mới để ghi. Nếu file đã tồn tại thì nó sẽ bị xoá để thay vào đó một file mới
"ab"	Mở file nhị phân đã có ghi thêm dữ liệu nối vào cuối file. Nếu file này chưa có thì file mới sẽ tạo ra
"r+b"	Mở file nhị phân đã tồn tại cho phép cả đọc, cả ghi
"w+b"	Mở file nhị phân mới cho cả ghi và đọc. Nếu đã có file với tên file này thì nó sẽ bị huỷ để thay vào đó là một file mới
"a+b"	Mở file nhị phân đã tồn tại hoặc tạo file mới để đọc và ghi thêm dữ liệu vào cuối file

Chú ý: Trong các kiểu đọc ghi, ta nên làm sạch vùng đệm trước khi chuyển từ đọc sang ghi hoặc ngược lại. Ta sẽ đề cập đến các hàm với tính năng xóa sau này

Ví dụ:

fileptr = fopen("C:\\TEPVB.txt", "wt");

b) Đóng file: fclose(fptr);

 Hàm kiểm tra con trỏ file đã ở vị trí kết thúc file chưa: **feof(fileptr)** hàm này kiểm tra cho kết quả là 1 nếu con trỏ file ở vị trí kết thúc file và bằng 0 nếu file chưa ở vị trí kết thúc file

Ví dụ: Tạo một file văn bản mới có tên là "THI_DU.TXT".

```
#include<stdio.h>
void main()
{
  FILE *fileptr; // khai báo con trỏ file
  fileptr = fopen ("THI_DU.TXT" , "w"); //mở file để ghi
  if (fileptr = = NULL)
        cout<<"ERROR: Khong mo duoc tep";
  else
  {
        // các lệnh ghi nội dung lên file
        fclose(fileptr); //đóng file
  }
}
```

c) Đóng tất cả các file đang mở: fcloseall();

Hàm dùng để đóng tất cả các file đang mở. Nếu lệnh thành công, hàm sẽ cho giá trị bằng số là số file được đóng, ngược lại nó cho hàm EOF.

d) Làm sạch vùng đệm: fflush(fileptr);

Hàm dùng làm sạch vùng đệm của file. Nếu lệnh thành công, hàm sẽ cho giá trị 0, ngược lại nó cho hàm EOF.

e) Làm sạch vùng đệm của các file đang mở: fflushall();

Hàm dùng để làm sạch vùng đệm của tất cả các file đang mở. Nếu lệnh thành công, hàm sẽ cho giá trị bằng số là số file được mở, ngược lại nó cho hàm EOF.

f) Kiểm tra lỗi file: ferror(fileptr);

Hàm dùng để kiểm tra lỗi khi thao tác trên file. Hàm cho giá trị 0 nếu không có lỗi, ngược lại hàm cho giá trị khác 0.

6.2.2. Các Thao Tác Ghi

Việc ghi dữ liệu vào file văn bản được thực hiện thông qua bởi các hàm:

putc *(ch, fileptr);*để ghi 1 ký tự ch vào file văn bản được trỏ bởi con trỏ fileptr

fputs *(str , fileptr); để g*hi 1 xâu ký tự *str* vào file văn bản được trỏ bởi con trỏ *fileptr*

fprintf*(fileptr,“ nội dung+[n ký_tự_đại_diện]+[ký_tự_ điều_khiển]”,[bt1, bt2,…,btn]);*

Trong đó:

fileptr: con trỏ trỏ đến file cần ghi

ký_tự_ điều_khiển :\t \a \n….

nội dung: là nội dung văn bản cần ghi lên file

ký_tự_đại_diện: các ký tự đại diện cho các kiểu dữ liệu của các giá trị cần ghi lên file thông qua các biểu thức bt1, bt2, …,btn. Các ký tự đại diện được cho trong bảng sau:

Kt đại diện	Ý nghĩa
%[n]c	Ghi ra một ký tự kiểu char với chiều dài tối thiểu là n (n là kiểu int).
%[n]d	Ghi ra một số nguyên có dấu: kiểu int, kiểu char với chiều dài tối thiểu là n.
%[n]u	Ghi ra một số nguyên không dấu: unsigned int, unsigned char, unsigned short với chiều dài tối thiểu là n.
%[n]ld	Ghi ra một số nguyên có dấu: kiểu long với chiều dài tối thiểu là n.
%[n]lu	Ghi ra một số nguyên không dấu: unsigned long với độ dài tối thiểu là n.
%[n.m]f	Ghi ra một số thực kiểu float hay double với chiều dài n và lấy m chữ số thập phân.
%[n]s	Ghi ra một chuỗi ký tự.

Ví dụ 1: Viết chương trình nhập một chuỗi ký tự và đổi thành chữ hoa và ghi vào file THI_DU.TXT, việc nhập kết thúc khi bấm phím enter.

```
#include<stdio.h>
#include<ctype.h>
void main()
{
  FILE *fileptr;
  char c;
```

```
  fileptr = fopen("THI_DU.TXT","w");
  do{
        putc(toupper(c=getchar()),fileptr);
        // hàm getchar() để nhập 1 ký tự
   }while(c != '\n');
  fclose(fileptr);
}
```

Ví dụ 2: Tạo file có tên là "THU.TXT" có nội dung như sau:

```
Thu ghi file:
So thu 1: 7
So thu 2: 9
Tong 7 + 9 =16
```

với giá trị 7 & 9 được nhập từ bàn phím trong chương trình tạo file "THU.TXT"

```
#include<stdio.h>
void main()
{
  FILE *fileptr;
  fileptr = fopen("THU.TXT","w");
  if (fileptr!=NULL)
  {     fprintf(fileptr,"Thu ghi file:");
        int i1, i2;
        cout<<"Nhap 2 so nguyen :";
        cin>>i1>>i2;
        fprintf(fileptr,"\nSo thu 1: %d",i1);
        fprintf(fileptr,"\nSo thu 2: %d",i2);
        fprintf(fileptr,"\nTong %d + %d = %d",i1,i2,i1+i2);
        fclose(fileptr);
  }
  }
```

Các thao tác đọc:

getc *(fileptr);* Đọc một ký tự từ file văn bản được trỏ bởi con trỏ file *fileptr*

fgets *(str , n, fileptr);* Đọc một xâu *có n(số nguyên) ký tự hoặc 1 dòng văn bản* từ file văn bản được trỏ bởi con trỏ file *fileptr lưu vào biến str. Nếu chiều dài của dòng văn bản tại con trỏ file□n thì chỉ đọc được nguyên dòng văn bản đó, nếu chiều dài dòng văn bản>n thì chỉ đọc được n ký tự*

fscanf*(fileptr," n ký_tự_đại_diện",danh sách địa chỉ n biến);*

Trong đó:

fileptr: con trỏ trỏ đến file cần đọc

danh sách địa chỉ n biến là các biến tương ứng với n_ký_tự_đại_diện, để sử dụng địa chỉ của biến thì ta thêm ký tự '&' trước biến. (ví dụ: int x; địa chỉ biến x : &x)

ký_tự_đại_diện: các ký tự đại diện cho các kiểu dữ liệu cần đọc từ file. Các ký tự đại diện được cho trong bảng sau:

Kt đại diện	Ý nghĩa
%c	Đọc một ký tự kiểu char.
%d	Đọc một số nguyên có dấu
%u	Đọc một số nguyên không dấu: unsigned int, unsigned char, unsigned short.
%ld	Đọc một số nguyên dài có dấu.
%lu	Đọc một số nguyên dài.
%f	Đọc một số thực kiểu float
%lf	Đọc một số thực kiểu double
%s	Đọc một chuỗi ký tự (không sử dụng ký tự & trước biến xâu)

Chú ý: Trong trường hợp lệnh fscanf để đọc xâu ký tự thì chỉ đọc từ vị trí con trỏ file cho đến khi gặp ký tự trắng đầu tiên hoặc dấu kết thúc chuỗi

Ví dụ: In nội dung file đang soạn ra màn hình (giả sử file có tên là "THUFILE.CPP")

```
#include<stdio.h>
#include<iostream.h>
void main()
{
  FILE *fileptr;
  fileptr = fopen("THUFILE.CPP","r");
  if (fileptr!= NULL)
        {     char *str;
              while (!feof(fileptr))
              {     fgets(str,80,fileptr); // đọc 1 dòng văn bản từ file
                    cout<<str;
              }
              fclose(fileptr);
        }
  else
        cout<<"Co loi, khong mo duoc tep";
}
```

6.3. FILE NHỊ PHÂN

File nhị phân là file mà các phần tử của nó chỉ biểu diễn 1 kiểu dữ liệu. Mỗi phần tử trong file đúng bằng kích thước của kiểu dữ liệu mà nó lưu trữ. Một file tin dù được xây dựng bằng cách nào thì bản chất của nó đơn giản cũng chỉ là dãy các byte (*có giá trị từ 0 ... 255*) ghi trên đĩa, với cách quan niệm này người ta gọi là file nhị phân. Số byte trên đĩa là độ dài của file, dấu hiệu kết thúc file là EOF (-1).

File nhị phân không xem được bằng lệnh Type của DOS hoặc các trình soạn thảo văn bản

6.3.1. Các Thao Tác Đọc Ghi Trên File Nhị Phân (Fread, Fwrite)

Thao tác ghi

fwrite *(&biến, kích thước, số phần tử , fileptr);*

Trong đó:

&biến : địa chỉ của biến cần ghi dữ liệu lên file, trong trường hợp ghi một khối (mảng) thì không có dấu &

Kích thước: kích thước của 1 phần tử cần ghi vào file. Thông thường sử dụng toán tử ***sizeof(biến| kiểu)*** để xác định kích thước.

số_phần_tử: số phần tử cần ghi lên file. Như vậy kích thước mỗi lần ghi lên file là *kích thước* số phần tử*

Ví dụ: tạo 1 file nhị phân có tên "SONGUYEN.DAT" chỉ chứa n giá trị nguyên ngẫu nhiên trong khoảng 0..100 với n nhập từ bàn phím

```
#include<stdio.h>
#include<iostream.h>
#include<stdlib.h>
void main()
{
  FILE *fileptr;
  fileptr= fopen("SONGUYEN.DAT","wb");
if ( fileptr! = NULL)
  {       int iN, iX, i;
          cout<<"Nhap so phan tu cho file:";
          cin>>iN;
          for (i=1; i<=iN; i++)
                  {       iX=random(101); //tạo ngẫu nhiên giá trị nguyên
                          fwrite(&iX,sizeof(iX),1,fileptr);
                  }
          fclose(fileptr);
  }
else
          printf("Co loi, khong tao duoc tep");
  }
```

Thao tác đọc

Việc đọc một phần tử của file cần phải có điều kiện, phải xem con trỏ file đã trỏ đến EOF hay chưa.

fread *(&biến , kích thước, số phần tử, fileptr);*

Trong đó:

&biến : địa chỉ của biến cần đọc dữ liệu từ file, trong trường hợp đọc một khối (mảng) thì không có dấu &

Kích thước: kích thước của 1 phần tử cần đọc từ file. Thông thường sử dụng toán tử ***sizeof(biến or kiểu)*** để xác định kích thước.

số_phần_tử: số phần tử cần đọc từ file. Như vậy kích thước mỗi lần đọc từ file là *kích thước* số phần tử*

chú ý: Nếu con trỏ file ở cuối file thì hàm **fread** cho kết quả là 0.

Ví dụ : Đọc file "SONGUYEN.DAT" ở ví dụ trên và hiển thị kết quả ra màn hình.

```
#include<conio.h>
#include<stdio.h>
#include<iostream.h>
void main()
{ clrscr();
 Cout<<"Noi dung file:";
 FILE*fileptr=fopen("SONGUYEN.DAT","rb");
 if (fileptr!=NULL)
 {
 while(!feof(fileptr))
  { int iX;
   if(fread(&iX,sizeof(int),1,fileptr)!=0)
       cout<<iX<<" ";
  }
 fclose(fileptr);
 }
 getch();
}
```

6.3.2. Các Thao Tác Trên File Nhị Phân(Fseek, Ftell, Rewind)

Chuyển con trỏ file về đầu file **rewind(fileptr)**

Di chuyển con trỏ file đến một vị trí nào đó trong file

fseek(fileptr, số_byte, hướng)

trong đó:

fileptr là con trỏ file

số_byte là số byte con trỏ file sẽ di chuyển tính từ ***hướng*** xác định

hướng là một trong 3 giá trị sau:

hướng = SEEK_SET hay 0: Xuất phát từ đầu file,

hướng = SEEK_CUR hay 1: Xuất phát tại vị trí con trỏ hiện tại

hướng = SEEK_END hay 2: Xuất phát từ cuối file.

Hàm di chuyển con trỏ file từ vị trí xác định bởi **hướng** qua một số byte bằng giá trị tuyệt đối của **số_byte**. Chiều di chuyển là về cuối file nếu **số_byte** dương, ngược lại sẽ di chuyển về phía đầu file. Khi thành công hàm trả về giá trị 0. Khi có lỗi hàm trả về giá trị khác không.

Cho biết vị trí hiện tại của con trỏ file: **ftell(fileptr)**

Khi thành công hàm cho biết vị trí hiện tại của con trỏ file (byte thứ mấy trên file). Số thứ tự của byte được tính từ 0. Khi có lỗi hàm trả về -1L.

Ví dụ sau sẽ giải thích thêm cách làm việc của hàm fseek và ftell

Giả sử file fptr có 3 ký tự lần lượt là A, B và C.

Sau câu lệnh

fseek(fileptr,0,SEEK_END);

thì con trỏ chỉ vị ở cuối file và ftell(fileptr) = 3;

Sau câu lệnh

fseek(fileptr,-1,SEEK_END);

thì con trỏ chỉ vị đặt tại C và ftell(fileptr) = 2;

Sau câu lệnh

fseek(fileptr,0,SEEK_SET);

thì con trỏ chỉ vị đặt tại A và ftell(fileptr) = 0;

1BÀI TẬP

Bài 1: Viết chương trình đọc một ma trận từ file văn bản (text) có tên "matran.txt" ra màn hình. Cấu trúc file ma trận như sau:

Dòng đầu tiên lưu 2 số nguyên m và n là hàng cột ma trân.

m dòng tiếp theo, mỗi dòng lưu n số nguyên.

Các giá trị trên một dòng được cách nhau bởi ký tự trắng.

Dữ liệu trên file Màn hình

Bài mẫu

```
void main()
{ FILE *fileptr;
int  iHang, iCot, iDem1, iDem2, iX;
fileptr=fopen("matran.txt","r");
if (fileptr!=NULL)
{ fscanf(fileptr,"%d%d",&iHang,&iCot);
// đọc kích thước ma trận
for (iDem1=1; iDem1<=iHang;  iDem1++)
{
for(iDem2=1; iDem2<=iCot; iDem2++)
{ fscanf(fileptr,"%d",&iX);
  cout.width(4);
  cout<<iX;
}
cout<<"\n";
}
            fclose(fileptr);
  }
  else cout<<"loi khong mo file duoc";
  getch();
}
```

Bài 2: Cho một file văn bản (text) lưu trữ các số nguyên, các giá trị trên file cách nhau bởi ký tự trắng. Hãy viết chương trình tách file này thành 2 file, một file chỉ lưu các số nguyên tố còn file kia lưu các giá trị còn lại, tên của cả 3 file nhập từ bàn phím.

Bài 3: Cho một file văn bản (text) lưu trữ các số nguyên, các giá trị trên file cách nhau bởi ký tự trắng. Hãy viết chương trình loại các giá trị âm ra khỏi file. Sau đó sắp xếp các giá trị trên file tăng dần.

Hướng dẫn:

Bước 1:Đọc dữ liệu từ file vào mảng, nếu giá trị đọc từ file □0 thì mới đưa vào mảng

Bước 2:Sắp xếp các giá trị trên mảng tăng dần

Bước 3:ghi dữ liệu từ mảng vào file

Bài 4: Cho một file văn bản (text) chứa các toạ độ trong mặt phẳng Oxy. Hãy tìm tam giác được tạo từ các điểm này có diện tích lớn nhất.

Cấu trúc file như sau:

Dòng đầu tiên lưu số nguyên n là số điểm trong mặt phẳng.

n dòng tiếp theo, mỗi dòng lưu 2 số nguyên x, y là toạ độ của một điểm.

Bài 5: Cho một file văn bản (text) chứa các toạ độ trong mặt phẳng Oxy. Hãy tìm tọa độ góc trên bên trái và góc dưới bên phải của hình chử nhật nhỏ nhất có các cạnh song song với các trục tọa độ chứa tất cả các điểm này (các điểm có thể nằm trên cạnh của hình chử nhật).

Cấu trúc file như sau:

Dòng đầu tiên lưu số nguyên n là số điểm trong mặt phẳng.

n dòng tiếp theo, mỗi dòng lưu 2 số nguyên x, y là toạ độ của một điểm.

Bài 6: Cho một file văn bản, hãy viết chương trình xuất nội dung file ra màn hình theo từng trang (mỗi trang 40 dòng) với tên file được nhập từ bàn phím.

Bài 7: Cho một file văn bản, hãy viết chương trình đếm xem file có bao nhiêu từ

Bài 8: Cho một file văn bản, hãy viết chương trình tìm số ký tự xuất hiện nhiều nhất trong file.

Hướng dẫn: sử dụng mảng có chỉ số là các ký tự có mã Ascii từ 32 -> 128 để đếm mỗi ký tự xuất hiện bao nhiêu lần. Sau đó tìm số lớn nhất trong mảng, từ đó sẽ xác định được ký tự xuất hiện nhiều nhất trong file.

Bài 9:Dùng File để lưu trữ hồ sơ thí sinh dự thi, thông tin của mỗi thí sinh bao gồm {Mã sinh viên, họ tên, ngày tháng năm sinh, giới tính, điểm trung binh}.

Yêu cầu:

Định nghĩa cấu trúc dữ liệu phù hợp để lưu trữ và xử lý hồ sơ

Nhập danh sách thí sinh từ bàn phím và lưu trữ trên bộ nhớ ngoài (chú ý Mã thí sinh là duy nhất) bằng 1 trong 2 cách sau:

Nhập n nguyên dương, sau đó nhập n thí sinh

Nhập các thí sinh, quá trình nhập kết thúc khi nhập 1 mã thi sinh rỗng

Giả sử đã có file chứa hồ sơ thí sinh, hãy thực hiện các thao tác sau:

Xuất danh sách ra màn hình với mỗi thí sinh trên 1 dòng

Cộng them 0.5 điểm cho những sinh viên sinh ngày 30 tháng 4 năm 2005

Xuất ra màn hình các thí sinh có điểm trung bình ≥ điểm chuẩn

Tạo file có tên là "Tr_Tuyen.Dat" để chứa các thí sinh trúng tuyển

Sắp xếp các thí sinh giảm dần theo điểm và ghi lại vào file

Bài 10: Cho 2 file nhị phân lưu trữ số nguyên, giá trị trong các file đã được sắp xếp tăng. Hãy viết chương trình tạo file nhị phân chứa các số nguyên từ 2 file trên với nội dung là dãy số tăng (không được dùng thuật toán sắp xếp).

PHỤ LỤC 1
MỘT SỐ LỖI THƯỜNG GẶP
TRONG KHI SOẠN THẢO CHƯƠNG TRÌNH NGUỒN

Nội dung	Giải thích
) expected	Biểu thức thiếu dấu ngoặc ‘)’. Kiểm tra lại biểu thức
Code has no effect	Kiểm tra lại cú pháp của lệnh(ví dụ: có thể thiếu 1 biểu thức trong vòng lặp for…)
Compound statement missing	Thiếu dấu ngoặc ‘}’ của hàm
Cannot convert ‘kiểu *’ to ‘kiểu’	Xem lại khai báo biến, đặc biệt là biến mảng và biến đơn
Declaration missing ;	Do xây dựng một hàm trong một hàm khác, nghĩa là hàm trên chưa có dấu ngoặc ‘}’
Declaration Syntax error	+Dòng khai báo biến trên lệnh này thiếu dấu ‘;’, hoặc giữa các biếêntrong cùng 1 kểu thiếu dấu ‘,’ +Hoặc một hàm tự định nghĩa chưa có ngoặc đóng ‘}’ kết thúc hàm
Expression Syntax	Kiểm tra lại cú pháp của lệnh
For statement missing ;	Thiếu dấu ‘;’ ở cú pháp lệnh for (lệnh for có 2 dấu ‘;’)
Function ‘X’ should have a prototype	Thiếu nguyên mẫu hàm, kiểm tra lại tên của hàm ‘X’ đã đúng hay chưa. Hàm này do mình định nghĩa hoặc nó có trong file *.h nào đó
If statement missing (	Thiếu cặp dấu ngoặc đơn ‘()’bọc biểu thức của lệnh if
Lvalue required	lệnh gán bị sai
Misplaced Break (Continue)	Lệnh Break (continue) không thuộc vòng lặp. Xem lại cú pháp của vòng lặp, đặc biệt là dấu ‘;’ ở cuối dòng lệnh
Misplaced else	Do lệnh if tương ứng đã bị kết thúc vì thiếu cặp ngoặc nhọn ‘{…}’ của khối lệnh trong thân lệnh if. Hoặc thiếu dấu ngoặc ‘}’ của ngoặc ‘{‘ ngay sau if. Hoặc sau biểu thức của if có dấu ‘;’
Multiple declaration for ‘I’	Khai báo trùng tên biến ‘I’
Parameter ‘A’ is never used	(cảnh báo)Biến A có khai báo nhưng không sử dụng
Statement missing ;	Dòng lệnh trên thiếu dấu ‘;’, hoặc có thể biểu thức thiếu dấu ngoặc đơn đóng ‘)’
Too few parameter in call to “tên_hàm(???)”	Sai số đối số khi gọi hàm “tên_ham(???)”
Too many decimal points	Số thực có quá nhiều dấu chấm phần thập phân
Too many initializers	Giá trị khởi tạo cho mảng nhiều hơn số phần tử mảng
Undefined symbol ‘X’	Biến ‘X’ chưa khai báo , kiểm tra lại khai báo biến, chú ý ký tự HOA và thường
Unterminated string or char constant	Dãy xâu ký tự thiếu dấu nháy kép ở cuối chuỗi
Unexpected }	Thiếu dấu ngoặc ‘{‘ của hàm
while statement missing)	Thiếu biểu thức trong lệnh while (xem lại cú pháp)

Unable to open include file ‘….h’=> Tìm các file đường dẫn (path) tới các file “*.H” và file “*.Lib” đang nằm ở thư mục nào, rồi sau đó vào Option->Directories…

PHỤ LỤC 2
MỘT SỐ NGUYÊN TẮC TRONG LẬP TRÌNH

1. Nguyên tắc đặt tên trong lập trình

Tên hằng viết chữ hoa và có dấu gạch dưới phân cách giữa các từ. Tên hằng là một danh từ.

Ví dụ: MIN_HEIGHT, PI

Tên kiểu enum: Viết hoa ký tự đầu của mỗi từ, viết chữ hoa cho các nhãn và có gạch dưới phân cách giữa các nhãn. Tên kiểu enum là một danh từ.

Ví dụ:
```
enum PinSateType {
  int PIN_ON;
  int PIN_OFF;
};
```

Tên kiểu union, Tên kiểu struct: Viết hoa ký tự đầu của mỗi từ, thêm từ *Type* vào cuối tên. Tên kiểu union và tên kiểu struct là một danh từ.

Ví dụ:
```
typedef union KieuChungType{
  int i;
  float f;
  char c;
};
```

```
typedef struct HocSinhType{
  char chrHoTen[30];
  int iGioiTinh;
  float fDiem;
};
```

Tên hàm: Viết hoa các ký tự đầu của mỗi từ, tên hàm là một động - danh từ.

Ví dụ: **GiaiPhuongTrinhBac2(...)**
KiemTraSoNguyenTo(...)

Tên biến: Ký tự đầu tiên viết bằng chữ thường và ký tự này là tiếp đầu ngữ của kiểu dữ liệu. Các ký tự đầu của mỗi từ tiếp theo viết chữ hoa.

Ví dụ:
```
iDelta      //i là tiếp đầu ngữ của kiểu số nguyên.
chrKyTu     //chr là tiếp đầu ngữ của kiểu ký tự
strString   //str là tiếp đầu ngữ của kiểu chuỗi
```

Sau đây là một số quy ước cho các tiếp đầu ngữ của các kiểu dữ liệu

Kiểu dữ liệu	**Tiếp đầu ngữ**
Int	i
short	s
Long	l
unsigned int	ui
unsigned short	us
unsigned long	ul
Char	chr

Float	f
Double	d
long double	ld
char*	str
FILE	file
Pointer	ptr

2. Nguyên tắc viết chương trình

a. Chú thích chương trình

Khi viết chương trình, thường sử dung các chú thích của ngôn ngữ để giải thích từng đoạn chương trình; để cho người đọc chương trình có thể hiểu rõ mục đích của mỗi đoạn chương trình làm gì và đồng thời giúp người lập trình nhớ lại thuật toán được dùng để giải quyết bài toán phức tạp. Bên cạnh đó, trước mỗi Hàm được định nghĩa, người ta thường chú thích mục của hàm, xác định đầu vào, đầu ra của hàm đó. Có thể tham khảo qua ví dụ sau:

Ví dụ:

```
//Khai bao cac thu vien co cac ham su dung
#include<conio.h>
#include<stdio.h>
#include<iostream.h>
/*Ham xuat tam giac Pascal ra man hinh
Dau vao: la mang chua noi dung cac he so cua tam giac pascal
Dau ra : Tam giac pascal hien len man hinh*/
void xuat1(int iA[100],int iN)
{ int 11,i2;
  for(int i1=1;i1<=n; i1++)
  { for(i2=n; i2>=i1; i2--)cout<<"  ";
   i2=i1*(i1-1)/2;
   for( i2=1; i2<=i; i2++)
    {
     cout<<"  ";
     cout.width(3);
     cout<<A[i1+i2];
    }
   cout<<"\n\n";
  }
}

void main()
{  int iA[100],iN;
  int i1, i2, i3;
  cout<<"Nhap n:";
```

```
  cin>>iN;
  iN=iN+1;
  iA[1]=1;
//Vong lap tao cac he so cua tam giac Pascal de luu vao mang 1 chieu A
  for (i1=2;i1<=n;i1++)
   { i3=i1*(i1-1)/2;          //xac dinh can duoi cua doan chua cac he so
    iA[i3+1]=1;               //Phan tư dau tien cua he thuc
    for(i2=2 ;i2<=i1-1;i2++)
          A[i3+i2]=A[i3-(i1-i2)]+A[i3-(i1-i2)+1];
//xac dinh cac he so
    A[i3+i1]=1;               //Phan tu cuoi cung cua he thuc
   }
  cout<<"\n";
  xuat(A,n);
  getch();
}
```

b. Cách trình bày một văn bản chương trình

Khi trình bày văn bản chương trình, nên xác định những lệnh thuộc cú pháp của một cấu trúc nào đó (if, for, while...) để xuống dòng đưa vào một khoảng trống so với cấu trúc của lệnh đó làm cho chương trình dễ đọc và dễ hiểu

Ví dụ: với đoạn chương trình trên, nếu viết không định dạng thì sẽ rất khó hình dung như sau:

```
#include<conio.h>
#include<stdio.h>
#include<iostream.h>
void xuat1(int iA[100],int iN)
{int 11,i2;
for(int i1=1;i1<=n; i1++)
{for(i2=n; i2>=i1; i2--)cout<<"  ";
i2=i1*(i1-1)/2;
for( i2=1; i2<=i; i2++)
{cout<<"  ";
cout.width(3);
cout<<A[i1+i2];}
cout<<"\n\n";}}
void main()
{int iA[100],iN;
int i1, i2, i3;
cout<<"Nhap n:";
cin>>iN;
iN=iN+1;
iA[1]=1;
for (i1=2;i1<=n;i1++)
{i3=i1*(i1-1)/2;
```

```
iA[i3+1]=1;
for(i2=2 ;i2<=i1-1;i2++)
A[i3+i2]=A[i3-(i1-i2)]+A[i3-(i1-i2)+1];
A[i3+i1]=1;}
cout<<"\n";
xuat(A,n);
getch();}
```