BỘ GIÁO DỤC VÀ ĐÀO TẠO

TIN HỌC

DÀNH CHO TRUNG HỌC CƠ SỞ

Quyển 3

NHÀ XUẤT BẢN GIÁO DỤC VIỆT NAM

BỘ GIÁO DỤC VÀ ĐÀO TẠO

PHẠM THẾ LONG (Chủ biên)
BÙI VIỆT HÀ - QUÁCH TẤT KIÊN - BÙI VĂN THANH

TIN HỌC

DÀNH CHO TRUNG HỌC CƠ SỞ

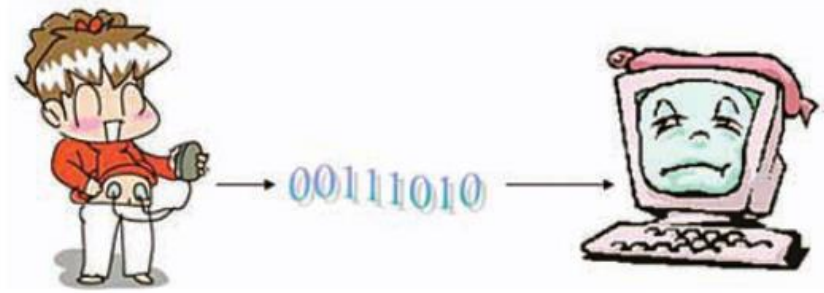
Quyển 3

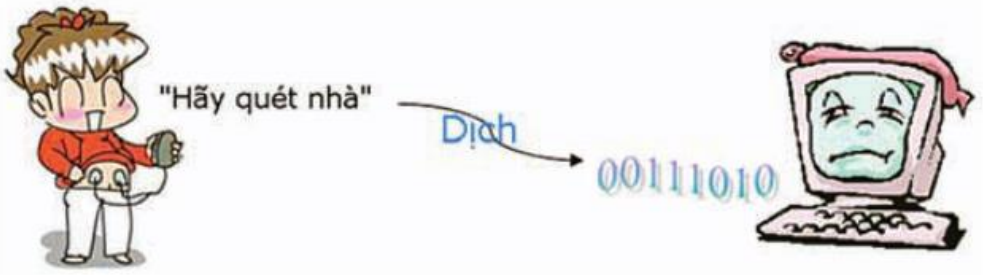
(Tái bản lần thứ sáu, có chỉnh lí, bổ sung)

NHÀ XUẤT BẢN GIÁO DỤC VIỆT NAM

Phần 1

LẬP TRÌNH ĐƠN GIẢN





BÀI 1

MÁY TÍNH VÀ CHƯƠNG TRÌNH MÁY TÍNH

1. Con người ra lệnh cho máy tính như thế nào?

Máy tính là công cụ trợ giúp con người để xử lí thông tin một cách hiệu quả. Tuy nhiên, để máy tính có thể thực hiện một công việc cụ thể, con người phải đưa ra những chỉ dẫn thích hợp cho máy tính.

Khi *nháy đúp chuột* lên biểu tượng của một phần mềm trên màn hình nền, phần mềm sẽ được khởi động. Bằng cách đó ta đã cho máy tính những *chỉ dẫn*, nói cách khác, đã *ra lệnh* cho máy tính khởi động phần mềm.

Khi soạn thảo văn bản, ta *gõ một phím* chữ (chẳng hạn phím chữ A), chữ tương ứng sẽ xuất hiện trên màn hình. Như vậy, ta cũng đã ra lệnh cho máy tính (hiện chữ lên màn hình).

Khi thực hiện lệnh *sao chép* một phần văn bản từ vị trí này sang vị trí khác, thực chất ta đã yêu cầu máy tính thực hiện liên tiếp hai lệnh, đó là lệnh sao chép nội dung phần văn bản vào bộ nhớ của máy tính và lệnh sao chép nội dung đó từ bộ nhớ vào vị trí mới trên văn bản.

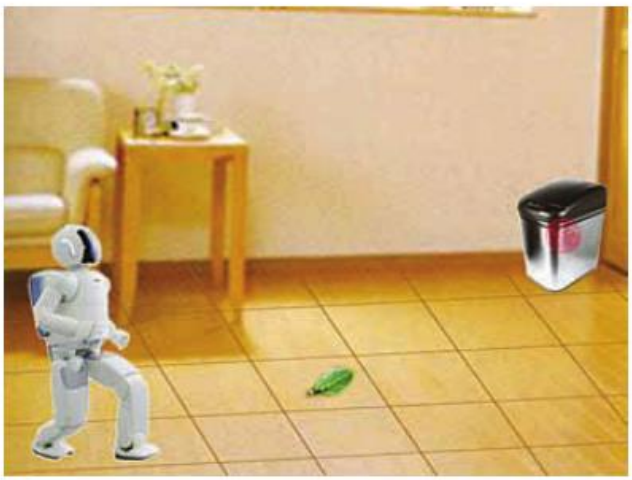
Như vậy, để chỉ dẫn máy tính thực hiện một công việc nào đó, con người ra lệnh cho máy tính một hoặc nhiều lệnh, máy tính sẽ *lần lượt thực hiện* các lệnh đó.

2. Ví dụ: rô-bốt nhặt rác

Rô-bốt (hay người máy) là một loại máy có thể tự động thực hiện được một số công việc thông qua sự điều khiển của con người. Chúng ta sẽ tìm hiểu cách ra lệnh cho máy tính thông qua một ví dụ về rô-bốt.

Giả sử ta có một rô-bốt có thể thực hiện được các thao tác cơ bản như tiến một bước, quay phải, quay trái, nhặt rác và bỏ rác vào thùng. Hình 1 sau đây mô tả vị trí của rô-bốt, rác và thùng rác. Có nhiều cách để chỉ dẫn rô-bốt di chuyển từ vị trí hiện thời, nhặt rác và bỏ vào thùng rác để ở nơi quy định. Dưới đây là một trong các cách đó:

1. Tiến 2 bước;
2. Quay trái, tiến 1 bước;
3. Nhặt rác;
4. Quay phải, tiến 3 bước;
5. Quay trái, tiến 2 bước;
6. Bỏ rác vào thùng;



Hình 1. Rô-bốt "nhặt rác"

Giả sử các lệnh trên được viết và lưu trong rô-bốt với tên *"Hãy nhặt rác"*. Khi đó ta chỉ cần ra lệnh *"Hãy nhặt rác"*, rô-bốt sẽ tự động thực hiện lần lượt các lệnh nói trên.

3. Viết chương trình - ra lệnh cho máy tính làm việc

Về thực chất, việc viết các lệnh để điều khiển rô-bốt trong ví dụ nói trên chính là viết *chương trình*. Tương tự, để điều khiển máy tính làm việc, chúng ta cũng viết chương trình máy tính.

Theo nghĩa đó, *chương trình máy tính là một dãy các lệnh mà máy tính có thể hiểu và thực hiện được*. Khi thực hiện chương trình, máy tính sẽ thực hiện các lệnh có trong chương trình một cách *tuần tự*, nghĩa là thực hiện xong một lệnh sẽ thực hiện lệnh tiếp theo, từ lệnh đầu tiên đến lệnh cuối cùng.

Trở lại ví dụ về rô-bốt nhặt rác, chương trình có thể có các lệnh như hình 2.

```
Bắt đầu
  Tiến 2 bước;
  Quay trái, tiến 1 bước;
  Nhặt rác;
  Quay phải, tiến 3 bước;
  Quay trái, tiến 2 bước;
  Bỏ rác vào thùng;
Kết thúc.
```

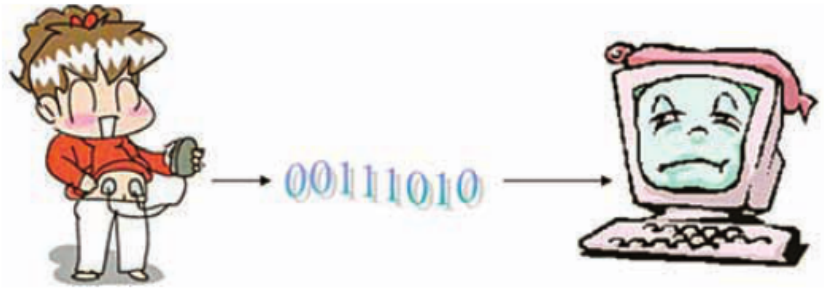
Hình 2. Ví dụ về chương trình

Tại sao cần viết chương trình?

Khi gõ một phím hoặc nháy chuột, thực chất ta đã "ra lệnh" cho máy tính. Tuy nhiên, trong thực tế các công việc con người muốn máy tính thực hiện rất đa dạng và phức tạp. Một lệnh đơn giản không đủ để chỉ dẫn cho máy tính. Vì thế việc viết nhiều lệnh và tập hợp lại trong một chương trình giúp con người điều khiển máy tính một cách dễ dàng và hiệu quả hơn.

4. Chương trình và ngôn ngữ lập trình

Chúng ta đã biết rằng, để máy tính có thể xử lí, thông tin đưa vào máy tính phải được chuyển đổi thành dạng dãy bit (dãy các số chỉ gồm 0 và 1). Các dãy bit là cơ sở để tạo ra ngôn ngữ dành cho máy tính, được gọi là *ngôn ngữ máy*. Những chương trình máy tính đầu tiên khi máy tính mới xuất hiện được viết chính bằng ngôn ngữ này.



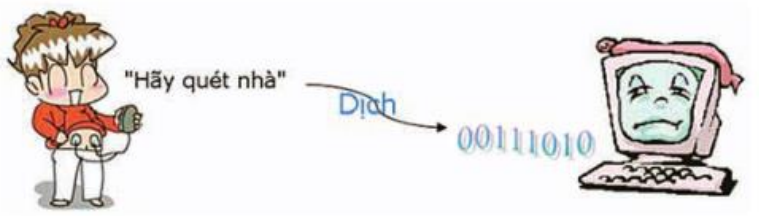
Hình 3

Tuy nhiên, việc viết chương trình bằng ngôn ngữ máy rất khó khăn và mất nhiều thời gian, công sức. Bởi lẽ, về mặt trực quan, các câu lệnh được viết dưới dạng các dãy bit khác xa với ngôn ngữ tự nhiên nên khó nhớ, khó sử dụng. Vì vậy người ta mong muốn có thể sử dụng được các từ có nghĩa, dễ hiểu và dễ nhớ để viết các câu lệnh thay cho các dãy bit khô khan. Các ngôn ngữ lập trình đã ra đời để phục vụ mục đích đó.

Ngôn ngữ lập trình là ngôn ngữ dùng để viết các chương trình máy tính.

Như vậy, để tạo chương trình máy tính, chúng ta phải viết chương trình theo một ngôn ngữ lập trình nào đó. Nói cách khác, ngôn ngữ lập trình là công cụ giúp để tạo ra các chương trình máy tính.

Tuy nhiên, máy tính vẫn chưa thể hiểu được trực tiếp các chương trình được viết bằng ngôn ngữ lập trình. Chương trình còn cần được chuyển đổi sang ngôn ngữ máy bằng một *chương trình dịch* tương ứng:



Hình 4

Tóm lại, việc tạo ra chương trình máy tính thực chất gồm hai bước sau:

(1) *Viết* chương trình bằng một ngôn ngữ lập trình;

(2) *Dịch* chương trình thành ngôn ngữ máy để máy tính hiểu được.

Hình 5

Kết quả nhận được sau bước (1) là danh sách các lệnh được lưu thành một tệp văn bản trong máy tính; còn kết quả của bước (2) là một tệp có thể thực hiện trên máy tính. Các tệp đó được gọi chung là chương trình.

Người ta thường viết chương trình bằng một chương trình soạn thảo (tương tự như chương trình soạn thảo văn bản). Chương trình soạn thảo và chương trình dịch cùng với các công cụ trợ giúp tìm kiếm, sửa lỗi và thực hiện chương trình thường được kết hợp vào một phần mềm, được gọi là *môi trường lập trình*. Ví dụ, với ngôn ngữ lập trình Pascal có hai môi trường làm việc phổ biến là Turbo Pascal và Free Pascal.

Có rất nhiều ngôn ngữ lập trình khác nhau. Có thể kể tên một số ngôn ngữ lập trình phổ biến hiện nay như C, Java, Basic, Pascal,... Mỗi ngôn ngữ lập trình được tạo ra với định hướng sử dụng trong một số lĩnh vực cụ thể và có điểm mạnh cũng như điểm yếu riêng.

GHI NHỚ

1. Con người chỉ dẫn cho máy tính thực hiện công việc thông qua các lệnh.

2. Viết chương trình là hướng dẫn máy tính thực hiện các công việc hay giải một bài toán cụ thể.

3. Ngôn ngữ dùng để viết các chương trình máy tính được gọi là ngôn ngữ lập trình.

Câu hỏi và bài tập

1. Trong ví dụ về rô-bốt, nếu thay đổi thứ tự của lệnh 1 và lệnh 2 trong chương trình, rô-bốt có thực hiện được công việc nhặt rác không? Hãy xác định vị trí mới của rô-bốt sau khi thực hiện xong chương trình. Em hãy bổ sung hai lệnh để rô-bốt trở lại vị trí ban đầu.

2. Hãy cho biết lí do cần phải viết chương trình để điều khiển máy tính.

3. Tại sao người ta phải tạo ra các ngôn ngữ lập trình trong khi có thể điều khiển máy tính bằng ngôn ngữ máy?

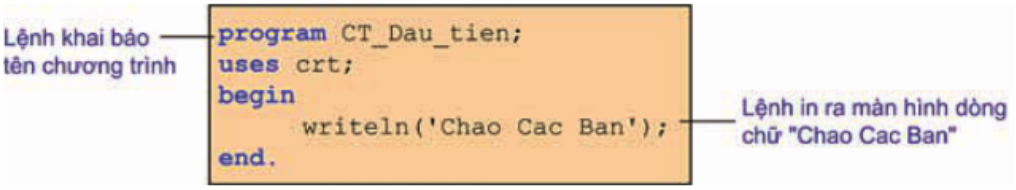
4. Chương trình dịch làm gì?

BÀI 2

LÀM QUEN VỚI CHƯƠNG TRÌNH VÀ NGÔN NGỮ LẬP TRÌNH

1. Ví dụ về chương trình

Ví dụ 1. Hình 6 dưới đây minh hoạ một chương trình đơn giản được viết bằng ngôn ngữ lập trình Pascal. Sau khi dịch, kết quả chạy chương trình là dòng chữ "Chao Cac Ban" được in ra trên màn hình.



Hình 6

Chương trình trên chỉ có năm dòng lệnh. Mỗi lệnh gồm các cụm từ khác nhau được tạo từ các chữ cái. Trong thực tế có những chương trình có đến hàng nghìn hoặc thậm chí hàng triệu dòng lệnh.

Trong các phần tiếp theo chúng ta sẽ tìm hiểu các câu lệnh trong chương trình được viết như thế nào.

2. Ngôn ngữ lập trình gồm những gì?

Chúng ta đã biết chương trình có thể có nhiều câu lệnh. Các câu lệnh được viết từ những kí tự nhất định. Tập kí tự này tạo thành bảng chữ cái của ngôn ngữ lập trình.

Giống như ngôn ngữ tự nhiên, mọi ngôn ngữ lập trình đều có bảng chữ cái riêng. Các câu lệnh chỉ được viết từ các chữ cái của bảng chữ cái đó.

Bảng chữ cái của các ngôn ngữ lập trình thường gồm các chữ cái tiếng Anh và một số kí hiệu khác như dấu phép toán (+, –, *, /,...), dấu đóng mở ngoặc, dấu nháy,... Nói chung, hầu hết các kí tự có trên bàn phím máy tính đều có mặt trong bảng chữ cái của mọi ngôn ngữ lập trình.

Mỗi câu lệnh trong chương trình trên gồm các từ và các kí hiệu được viết theo một quy tắc nhất định. Các quy tắc này quy định cách viết các từ và thứ tự của chúng. Chẳng hạn, trong ví dụ trên các từ được cách nhau bởi một hoặc nhiều dấu cách, một số câu lệnh được kết thúc bằng dấu chấm phẩy (;), dòng lệnh thứ tư có cụm từ nằm trong cặp dấu ngoặc đơn,... Nếu câu lệnh bị viết sai quy tắc, chương trình dịch sẽ nhận biết và thông báo lỗi.

Mặt khác, mỗi câu lệnh đều có một ý nghĩa riêng xác định các thao tác mà máy tính cần thực hiện. Dòng lệnh đầu tiên trong ví dụ trên là câu lệnh đặt tên (khai báo) cho chương trình, dòng lệnh thứ tư chỉ thị cho máy tính in ra màn hình dòng chữ "*Chao Cac Ban*",...

Tóm lại, về cơ bản ngôn ngữ lập trình gồm *bảng chữ cái* và *các quy tắc* để viết các câu lệnh có ý nghĩa xác định, cách bố trí các câu lệnh,... sao cho có thể tạo thành một chương trình hoàn chỉnh và thực hiện được trên máy tính.

3. Từ khoá và tên

Trong chương trình trên, ta thấy có các từ như **program**, **uses**, **begin**, **end**,... Đó là những *từ khoá* được quy định tuỳ theo mỗi ngôn ngữ lập trình. Từ khoá của một ngôn ngữ lập trình là những *từ dành riêng*, không được dùng các từ khoá này cho bất kì mục đích nào khác ngoài mục đích sử dụng do ngôn ngữ lập trình quy định. Theo quy định, **program** là từ khoá dùng để khai báo tên chương trình, **uses** là từ khoá khai báo các thư viện. Các từ khoá **begin** và **end** luôn đi thành cặp dùng để thông báo các điểm bắt đầu và kết thúc phần thân chương trình.

Ngoài các từ khoá, chương trình trong ví dụ 1 còn có các từ như `CT_Dau_tien`, `crt`,... Đó là các *tên* được dùng trong chương trình. Khi viết chương trình để giải các bài toán, ta thường thực hiện tính toán với những đại lượng (ví dụ như so sánh chiều cao, tính điểm trung bình,...) hoặc xử lí các đối tượng khác nhau. Các đại lượng và đối tượng này đều phải được đặt tên. Ví dụ tên `CT_Dau_tien` là tên của chương trình.

Tên do người lập trình đặt phải tuân thủ các quy tắc của ngôn ngữ lập trình cũng như của chương trình dịch và thoả mãn:

– *Tên khác nhau* tương ứng với những *đại lượng khác nhau*.

– Tên *không được trùng với các từ khoá*.

Tên trong chương trình được dùng để phân biệt và nhận biết các đại lượng khác nhau. Do vậy, tuy có thể đặt tên tuỳ ý, nhưng để dễ sử dụng nên đặt tên sao cho *ngắn gọn*, *dễ nhớ* và *dễ hiểu*.

Ví dụ 2. Tên hợp lệ trong ngôn ngữ lập trình Pascal không được bắt đầu bằng chữ số và không được chứa dấu cách (kí tự trống). Do vậy chúng ta có thể đặt tên *STamgiac* để chỉ diện tích hình tam giác, hoặc đặt tên *ban_kinh* cho bán kính của hình tròn,.... Các tên đó là những *tên hợp lệ*, còn các tên *Lop em*, *10A*,... là những tên không hợp lệ.

Chúng ta sẽ dần làm quen với cách đặt tên và sử dụng tên trong các bài sau.

4. Cấu trúc chung của chương trình

Cấu trúc chung của mọi chương trình gồm:

• *Phần khai báo* thường gồm các câu lệnh dùng để:

- Khai báo tên chương trình;

- Khai báo các thư viện (chứa các lệnh viết sẵn có thể sử dụng trong chương trình) và một số khai báo khác.

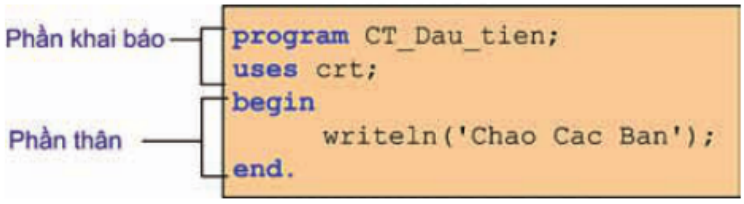
• *Phần thân* của chương trình gồm các câu lệnh mà máy tính cần thực hiện. Đây là *phần bắt buộc phải có*.

Phần khai báo có thể có hoặc không. Tuy nhiên, nếu có *phần khai báo thì nó phải được đặt trước phần thân chương trình*.

Trở lại với chương trình trong hình 6, ta có thể thấy:

– Phần khai báo gồm hai lệnh: khai báo tên chương trình là `CT_dau_tien` với từ khoá **`program`** và khai báo thư viện `crt` với từ khoá **`uses`**.

– Phần thân chỉ gồm các từ khoá **`begin`** và **`end`** cho biết điểm bắt đầu, điểm kết thúc phần thân và một câu lệnh là `writeln('Chao Cac Ban')` để in ra màn hình dòng chữ "*Chao Cac Ban*".



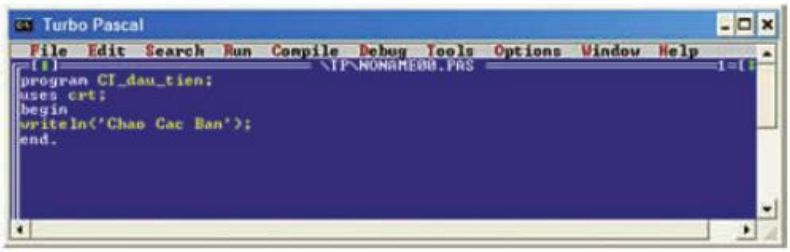
Hình 7

5. Ví dụ về ngôn ngữ lập trình

Trong phần này chúng ta sẽ làm quen với một ngôn ngữ lập trình cụ thể, ngôn ngữ Pascal. Để lập trình bằng ngôn ngữ Pascal, máy tính cần được cài đặt môi trường lập trình trên ngôn ngữ này.

Dưới đây là minh hoạ việc viết và chạy một chương trình cụ thể trong môi trường lập trình *Turbo Pascal.*

Khi khởi động phần mềm Turbo Pascal, cửa sổ soạn thảo chương trình như hình 8 dưới đây. Ta có thể sử dụng bàn phím để soạn thảo chương trình tương tự như soạn thảo văn bản với Word.

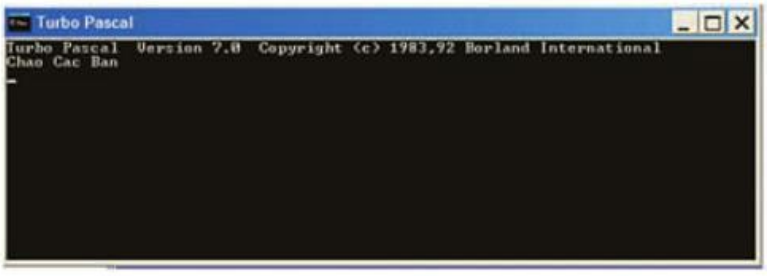


Hình 8

Sau khi đã soạn thảo xong, nhấn tổ hợp phím **Alt+F9** để dịch chương trình. Chương trình dịch sẽ kiểm tra các lỗi chính tả và cú pháp; nếu gặp câu lệnh sai, chương trình dịch sẽ thông báo để người viết chương trình dễ nhận biết và chỉnh sửa. Nếu đã hết lỗi, sau khi dịch, màn hình có dạng như hình 9 dưới đây:

Hình 9

Để chạy chương trình, ta nhấn tổ hợp phím **Ctrl+F9**. Trên màn hình sẽ hiện ra kết quả làm việc của chương trình, chẳng hạn dòng chữ "*Chao Cac Ban*" như hình 10 dưới đây.



Hình 10

GHI NHỚ

1. Ngôn ngữ lập trình là tập hợp các kí hiệu và quy tắc viết các lệnh tạo thành một chương trình hoàn chỉnh và thực hiện được trên máy tính.

2. Nhiều ngôn ngữ lập trình có tập hợp các từ khoá dành riêng cho những mục đích sử dụng nhất định.

3. Một chương trình thường có hai phần: Phần khai báo và phần thân chương trình.

4. Tên được dùng để phân biệt các đại lượng trong chương trình và do người lập trình đặt.

Câu hỏi và bài tập

1. Hãy cho biết các thành phần cơ bản của một ngôn ngữ lập trình.

2. Cho biết sự khác nhau giữa từ khoá và tên. Cho biết cách đặt tên trong chương trình.

3. Trong các tên sau đây, tên nào là hợp lệ trong ngôn ngữ Pascal?

A) a; B) Tamgiac; C) 8a; D) Tam giac;

E) beginprogram; F) end; G) b1; H) abc.

4. Hãy cho biết các thành phần chính trong cấu trúc của chương trình.

5. Các chương trình Pascal sau đây có hợp lệ không, tại sao?

a) Chương trình 1

```
begin
end.
```

b) Chương trình 2

```
begin
 program CT_thu;
 writeln('Chao cac ban');
end.
```

Bài đọc thêm 1. Một số ngôn ngữ lập trình thông dụng

C là ngôn ngữ lập trình dành cho các nhà lập trình chuyên nghiệp và hiện được dùng nhiều nhất trên thế giới.

Java là ngôn ngữ lập trình tương đối mới, phù hợp cho lập trình để tạo các chương trình ứng dụng trên mạng Internet.

Basic là ngôn ngữ lập trình tương đối dễ dùng, có thể nhanh chóng tạo ra các chương trình ứng dụng, cũng được rất nhiều nhà lập trình sử dụng.

Pascal do nhà bác học Niklaus Wirth sáng lập ra vào những năm 70 của thế kỉ XX. Đây là một ngôn ngữ có cú pháp sáng sủa, dễ hiểu và thường được dạy trong nhà trường và dành cho người mới học lập trình. Chính vì thế Pascal thường được gọi là "ngôn ngữ lập trình của học đường". Hai môi trường lập trình trên ngôn ngữ này đang được sử dụng phổ biến hiện nay ở Việt Nam là Turbo Pascal và Free Pascal.

Bài thực hành 1

LÀM QUEN VỚI TURBO PASCAL

1. Mục đích, yêu cầu

- Bước đầu làm quen với môi trường lập trình Turbo Pascal, nhận diện màn hình soạn thảo, cách mở các bảng chọn và chọn lệnh.
- Gõ được một chương trình Pascal đơn giản.
- Biết cách dịch, sửa lỗi trong chương trình, chạy chương trình và xem kết quả.

2. Nội dung

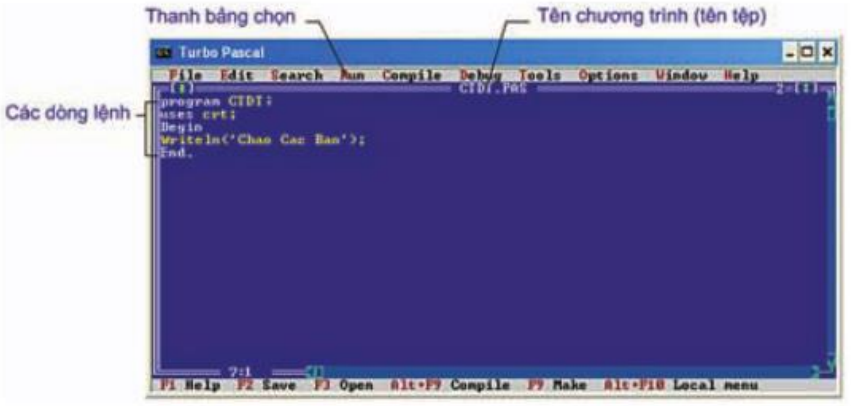
BÀI 1. Làm quen với việc khởi động và thoát khỏi Turbo Pascal. Nhận biết các thành phần trên màn hình của Turbo Pascal.

a) Khởi động Turbo Pascal bằng một trong hai cách:

Cách 1: Nháy đúp chuột vào biểu tượng  trên màn hình nền;

Cách 2: Nháy đúp chuột vào tên tệp Turbo.exe trong thư mục chứa tệp này (thường là thư mục con TP\BIN).

b) Quan sát màn hình của Turbo Pascal và so sánh với hình 11 dưới đây:



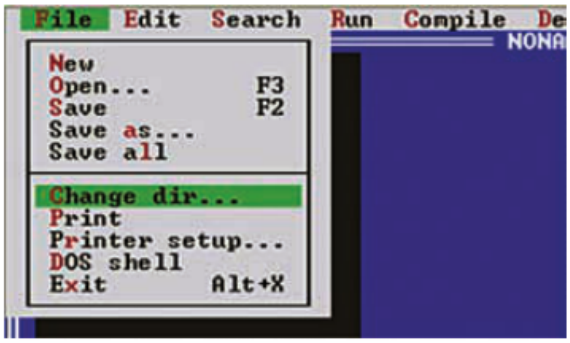
Hình 11

c) Nhận biết các thành phần: Thanh bảng chọn; tên tệp đang mở; con trỏ; dòng trợ giúp phía dưới màn hình.

d) Nhấn phím F10 để mở bảng chọn, sử dụng các phím mũi tên sang trái và sang phải (← và →) để di chuyển giữa các bảng chọn.

e) Nhấn phím Enter để mở một bảng chọn.

f) Quan sát các lệnh trong từng bảng chọn.



Hình 12

Mở các bảng chọn bằng cách khác: Nhấn tổ hợp phím Alt và phím tắt của bảng chọn (chữ màu đỏ ở tên bảng chọn, ví dụ phím tắt của bảng chọn File là F, bảng chọn Run là R,...).

g) Sử dụng các phím mũi tên lên và xuống (↑ và ↓) để di chuyển giữa các lệnh trong một bảng chọn.

h) Nhấn tổ hợp phím Alt+X để thoát khỏi Turbo Pascal.

BÀI 2. Soạn thảo, lưu, dịch và chạy một chương trình đơn giản.

a) Khởi động lại Turbo Pascal và gõ các dòng lệnh dưới đây:

```
program CT_Dau_tien;
uses crt;
begin
    clrscr;
    writeln('Chao cac ban');
    writeln('Toi la Turbo Pascal');
end.
```

Lưu ý

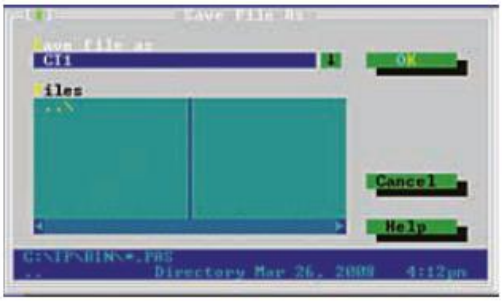
– Gõ đúng và không để sót các dấu nháy đơn ('), dấu chấm phẩy (;) và dấu chấm (.) trong các dòng lệnh.

– Tương tự như soạn thảo văn bản, khi soạn thảo cũng có thể sử dụng các phím mũi tên hoặc dùng chuột để di chuyển con trỏ, nhấn phím Enter để xuống dòng mới, nhấn các phím Delete hoặc BackSpace để xoá.

– Cặp từ khoá `begin` và `end` bao giờ cũng đi thành cặp. Do vậy, khi soạn thảo chương trình để khỏi bỏ sót, mỗi khi gõ vào từ khoá `begin`, em nên gõ luôn từ khoá `end` rồi sau đó hãy chèn các câu lệnh vào giữa cặp từ khoá đó.

– Câu lệnh `uses crt` được dùng để khai báo thư viện `crt`, còn lệnh `clrscr` có tác dụng xoá màn hình kết quả. Chỉ có thể sử dụng câu lệnh `clrscr` sau khi đã khai báo thư viện `crt`.

b) Nhấn phím F2 (hoặc lệnh File → Save) để lưu chương trình. Khi hộp thoại hiện ra, gõ tên tệp (ví dụ CT1) trong ô Save file as (phần mở rộng ngầm định là .pas) và nhấn Enter (hoặc nháy OK).



Hình 13

c) Nhấn tổ hợp phím Alt+F9 để dịch chương trình. Khi đó chương trình được dịch và kết quả hiện ra có thể như hình 14 sau đây:

Hình 14

Nhấn phím bất kì để đóng hộp thoại.

d) Nhấn tổ hợp phím Ctrl+F9 để chạy chương trình. Sau đó nhấn tổ hợp phím Alt+F5 để quan sát kết quả.

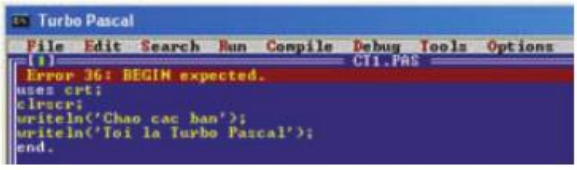
Hình 15

Nhấn phím bất kì để quay về màn hình soạn thảo.

Như vậy, chúng ta đã viết được một chương trình hoàn chỉnh và chạy được.

BÀI 3. Chỉnh sửa chương trình và nhận biết một số lỗi.

a) Xoá dòng lệnh **begin**. Dịch chương trình và quan sát thông báo lỗi như hình 16 dưới đây:



Hình 16. Lỗi 36: Thiếu BEGIN

b) Nhấn phím bất kì và gõ lại lệnh **begin** như cũ. Xoá dấu chấm sau chữ **end**. Dịch chương trình và quan sát thông báo lỗi (h. 17).

Hình 17. Lỗi 10: Không tìm thấy kết thúc tệp

Lưu ý

– Dấu chấm phẩy (;) được dùng để phân cách các lệnh trong Pascal. Sau câu lệnh ngay trước từ khoá **end** có thể không cần đặt dấu chấm phẩy.
– Từ khoá **end** kết thúc phần thân chương trình luôn có một dấu chấm (.) đi kèm.

c) Nhấn tổ hợp phím Alt+X để thoát khỏi Turbo Pascal, nhưng không lưu các chỉnh sửa.

BÀI 4. Hãy chỉnh sửa chương trình để in ra lời chào và tên của em, ví dụ:

```
Chao cac ban
Toi la Pham Nhu Anh
```

TỔNG KẾT

1. Các bước đã thực hiện:
   1. Khởi động Turbo Pascal;
   2. Soạn thảo chương trình;
   3. Biên dịch chương trình: **Alt+F9**;
   4. Chạy chương trình: **Ctrl+F9**;
2. Pascal không phân biệt chữ hoa, chữ thường: `begin`, `BeGin` hay `BEGIN` đều đúng.
3. Các từ khoá của Pascal trong bài là: `program`, `begin`, `end`, `uses`.
4. Lệnh kết thúc chương trình là `end.` (có dấu chấm), mọi thông tin đứng sau lệnh này bị bỏ qua trong quá trình dịch chương trình.
5. Dấu chấm phẩy (;) được dùng để phân cách các lệnh trong Pascal.
6. Lệnh `writeln` in thông tin ra màn hình và đưa con trỏ xuống đầu dòng tiếp theo. Có thể in thông tin dạng văn bản hoặc dạng số. Văn bản cần in ra phải được đặt trong cặp dấu nháy đơn.

   Lệnh `write` tương tự như `writeln`, nhưng *không đưa con trỏ xuống đầu dòng tiếp theo*.
7. Câu lệnh `clrscr` dùng để xoá màn hình kết quả và chỉ sử dụng được khi đã khai báo thư viện `crt`. Thư viện `crt` chứa các lệnh viết sẵn để thao tác với màn hình và bàn phím.

Bài đọc thêm 2. Một số bảng chọn thường dùng trong Pascal

1. Bảng chọn **File** chứa một số lệnh để làm việc với tệp:
   New: Mở cửa sổ mới để soạn thảo chương trình;
   Open: Mở tệp chương trình đã được lưu trên đĩa;
   Save: Lưu tệp đang soạn thảo;
   Save as: Lưu tệp đang soạn thảo với một tên khác;
   Save all: Lưu tất cả các tệp đang mở (kể cả những tệp bị che khuất);
   Exit: Thoát khỏi Turbo Pascal.
2. Bảng chọn **Compile** gồm một số lệnh biên dịch:
   Compile: Biên dịch chương trình đang làm việc;
   Destination: Thay đổi vị trí lưu kết quả biên dịch (trong bộ nhớ hay tạo tệp chạy trực tiếp).
3. Bảng chọn **Run**
   Run: Chạy chương trình đang làm việc và đã biên dịch.
4. Bảng chọn **Options** gồm một số lệnh thiết đặt các tuỳ chọn.

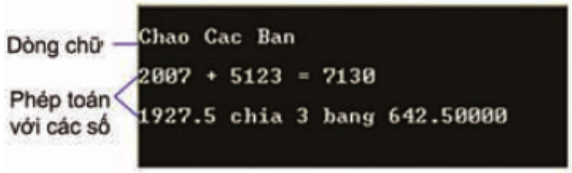
BÀI 3

CHƯƠNG TRÌNH MÁY TÍNH VÀ DỮ LIỆU

1. Dữ liệu và kiểu dữ liệu

Máy tính là công cụ xử lí thông tin, còn chương trình chỉ dẫn cho máy tính cách thức xử lí thông tin để có kết quả mong muốn. Thông tin rất đa dạng nên dữ liệu trong máy tính cũng rất khác nhau về bản chất. Để dễ dàng quản lí và tăng hiệu quả xử lí, các ngôn ngữ lập trình thường phân chia dữ liệu thành các *kiểu* khác nhau: chữ, số nguyên, số thập phân,...

Ví dụ 1. Hình 18 dưới đây minh hoạ kết quả thực hiện của một chương trình: in ra màn hình với các kiểu dữ liệu quen thuộc là chữ và số.



Hình 18

Các kiểu dữ liệu khác nhau thường được xử lí theo các cách khác nhau. Chẳng hạn, ta có thể thực hiện các phép toán số học với các số, nhưng với các kí tự hay xâu kí tự thì các phép toán đó không có nghĩa.

Các ngôn ngữ lập trình định nghĩa sẵn một số kiểu dữ liệu cơ bản. Kiểu dữ liệu xác định miền giá trị của dữ liệu và các phép toán có thể thực hiện trên các giá trị đó. Dưới đây là một số kiểu dữ liệu thường dùng nhất:

- *Số nguyên*, ví dụ số học sinh của một lớp, số sách trong thư viện,...
- *Số thực*, ví dụ chiều cao của bạn Bình, điểm trung bình môn Toán,...
- *Kí tự* là một chữ, chữ số hay kí hiệu đặc biệt khác, ví dụ: "a", "A", "+", "1" (chữ số 1, khác với số nguyên 1), " " (kí tự trống),... Trong đa số các trường hợp, kí tự thường là một "chữ cái" lấy từ bảng chữ cái của ngôn ngữ lập trình.
- *Xâu kí tự* (hay *xâu*) là dãy liên tiếp các kí tự (tối đa 255 kí tự), ví dụ: "*Chao cac ban*", "*Lop 8E*", "2/9/1945"...

Trong các ngôn ngữ lập trình, dữ liệu kiểu số nguyên còn có thể được phân chia tiếp thành các kiểu nhỏ hơn theo các phạm vi giá trị khác nhau, dữ liệu kiểu số thực còn có thể được phân chia thành các kiểu có độ chính xác (số chữ số thập phân) khác nhau.

Ngoài các kiểu nói trên, mỗi ngôn ngữ lập trình cụ thể còn định nghĩa nhiều kiểu dữ liệu khác. Số các kiểu dữ liệu và tên kiểu dữ liệu trong mỗi ngôn ngữ lập trình có thể khác nhau.

Ví dụ 2. Bảng 1 dưới đây liệt kê một số kiểu dữ liệu cơ bản của ngôn ngữ lập trình Pascal:

Tên kiểu	Phạm vi giá trị
`integer`	Số nguyên trong khoảng từ − 32768 đến 32767.
`real`	Số thực có giá trị tuyệt đối trong khoảng $2{,}9 \times 10^{-39}$ đến $1{,}7 \times 10^{38}$ và số 0.
`char`	Kí tự trong bảng chữ cái.
`string`	Xâu kí tự, tối đa gồm 255 kí tự.

Bảng 1

Trong Pascal, để chỉ rõ cho chương trình dịch hiểu dãy chữ số là kiểu xâu, ta phải đặt dãy số đó trong cặp dấu nháy đơn. Ví dụ '5324', '863'.

2. Các phép toán với dữ liệu kiểu số

Trong mọi ngôn ngữ lập trình ta đều có thể thực hiện các phép toán số học cộng, trừ, nhân và chia với các số nguyên và số thực.

Chẳng hạn, bảng dưới đây là kí hiệu của các phép toán số học đó trong ngôn ngữ Pascal:

Kí hiệu	Phép toán	Kiểu dữ liệu
`+`	cộng	số nguyên, số thực
`-`	trừ	số nguyên, số thực
`*`	nhân	số nguyên, số thực
`/`	chia	số nguyên, số thực
`div`	chia lấy phần nguyên	số nguyên
`mod`	chia lấy phần dư	số nguyên

Bảng 2

Chúng ta đã quen thuộc với các phép toán cộng, trừ, nhân và chia. Tuy nhiên, hãy lưu ý rằng hầu hết các ngôn ngữ lập trình đều xem kết quả chia hai số n và m (tức n/m) là số thực, cho dù n và m là các số nguyên và n có thể chia hết cho m.

Dưới đây là các ví dụ về phép chia, phép chia lấy phần nguyên và phép chia lấy phần dư:

5/2 = 2.5 −12/5 = −2.4.

5 div 2 = 2 −12 div 5 = −2

5 mod 2 = 1 −12 mod 5 = −2

Sử dụng dấu ngoặc, ta có thể kết hợp các phép tính số học nói trên để có các *biểu thức số học* phức tạp hơn. Sau đây là một số ví dụ về biểu thức số học và cách viết chúng trong ngôn ngữ lập trình Pascal:

Biểu thức số học	Cách viết trong Pascal
$a \times b - c + d$	`a*b-c+d`
$15 + 5 \times \frac{a}{2}$	`15+5*(a/2)`
$\frac{x+5}{a+3} - \frac{y}{b+5} \quad (x+2)^2$	`(x+5)/(a+3)-y/(b+5)*(x+2)*(x+2)`

Chú ý rằng khi viết các biểu thức toán, để dễ phân biệt, ta có thể dùng các cặp dấu ngoặc tròn (và), dấu ngoặc vuông [và], dấu ngoặc nhọn { và } để gộp các phép toán, nhưng trong các ngôn ngữ lập trình chỉ được sử dụng *dấu ngoặc tròn* cho mục đích này.

Ví dụ, biểu thức $\frac{(a + b)(c - d) + 6}{3} - a$ khi viết trong Pascal sẽ có dạng:

```
((a + b)*(c - d) + 6)/3 - a
```

3. Các phép so sánh

Ngoài các phép toán số học, ta còn thường *so sánh* các số.

Khi viết chương trình, để so sánh dữ liệu (số, biểu thức,...) chúng ta sử dụng các kí hiệu do ngôn ngữ lập trình quy định.

Kí hiệu các phép toán và phép so sánh có thể khác nhau, tuỳ theo từng ngôn ngữ lập trình.

Bảng 3 dưới đây cho biết kí hiệu của các phép so sánh trong ngôn ngữ Pascal:

Phép so sánh	Kí hiệu toán học	Kí hiệu trong Pascal	Ví dụ trong Pascal
Bằng	=	=	5 = 5
Khác	≠	<>	6 <> 5
Nhỏ hơn	<	<	3 < 5
Nhỏ hơn hoặc bằng	≤	<=	5 <= 6
Lớn hơn	>	>	9 > 6
Lớn hơn hoặc bằng	≥	>=	9 >= 6

Bảng 3

Kết quả của phép so sánh chỉ có thể là đúng hoặc sai. Ví dụ, phép so sánh 9 > 6 cho kết quả đúng, 10 = 9 cho kết quả sai hoặc 5 < 3 cũng cho kết quả sai,...

Để so sánh giá trị của hai biểu thức, chúng ta cũng sử dụng một trong các kí hiệu toán học trong bảng 3. Ví dụ:

$$5 \times 2 = 9;\ 15 + 7 > 20 - 3;\ 5 + x \leq 10$$

Kết quả so sánh thứ nhất là sai (10 = 9), còn kết quả so sánh thứ hai (22 > 17) là đúng. Kết quả so sánh thứ ba ($5 + x \leq 10$) đúng hoặc sai phụ thuộc vào giá trị cụ thể của x.

4. Giao tiếp người - máy tính

Trong khi thực hiện chương trình máy tính, con người thường có nhu cầu can thiệp vào quá trình tính toán, thực hiện việc kiểm tra, điều chỉnh, bổ sung. Ngược lại, máy tính cũng cho thông tin về kết quả tính toán, thông báo, gợi ý,... Quá trình trao đổi dữ liệu hai chiều như thế thường được gọi là giao tiếp hay tương tác giữa người và máy tính. Với các máy tính cá nhân, tương tác người-máy thường được thực hiện nhờ các thiết bị chuột, bàn phím và màn hình. Dưới đây là một số trường hợp tương tác người-máy.

a) Thông báo kết quả tính toán

Thông báo kết quả tính toán là yêu cầu đầu tiên đối với mọi chương trình. Ví dụ, câu lệnh

```
write('Dien tich hinh tron la ',X);
```

in kết quả tính diện tích hình tròn ra màn hình như hình 19 dưới đây:

Hình 19

b) Nhập dữ liệu

Một trong những tương tác thường gặp là chương trình yêu cầu nhập dữ liệu. Chương trình sẽ tạm ngừng để chờ người dùng "nhập dữ liệu" từ bàn phím hay bằng chuột. Hoạt động tiếp theo của chương trình sẽ tuỳ thuộc vào dữ liệu được nhập vào.

Ví dụ, chương trình yêu cầu nhập năm sinh từ bàn phím. Khi đó ta cần gõ một số tự nhiên ứng với năm sinh. Sau khi nhấn phím **Enter** để xác nhận, chương trình sẽ tiếp tục hoạt động.

Hình 20

Hai câu lệnh Pascal dưới đây sẽ cho kết quả như hình trên:

```
write('Ban hay nhap nam sinh:');
read(NS);
```

c) Tạm ngừng chương trình

Có hai chế độ tạm ngừng của chương trình: Tạm ngừng trong một khoảng thời gian nhất định và tạm ngừng cho đến khi người dùng nhấn phím.

Ví dụ 3. Giả sử trong chương trình Pascal có các câu lệnh sau:

```
Writeln('Cac ban cho 2 giay nhe...');
Delay(2000);
```

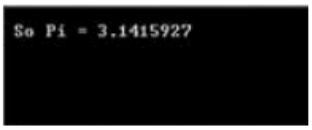
Sau khi in ra màn hình dòng chữ "*Cac ban cho 2 giay nhe...*", chương trình sẽ tạm ngừng trong 2 giây, sau đó mới thực hiện tiếp.

Hình 21

Ví dụ 4. Khi chạy đoạn chương trình Pascal có các câu lệnh

```
writeln('So Pi = ',Pi);
readln;
```

sau khi thông báo kết quả tính số π, chương trình sẽ tạm ngừng chờ người dùng nhấn phím **Enter**, rồi mới thực hiện tiếp.



Hình 22

d) Hộp thoại

Hộp thoại được sử dụng như một công cụ cho việc giao tiếp người-máy tính trong khi chạy chương trình. Ví dụ, khi người dùng thao tác để thoát khỏi chương trình đang chạy, hộp thoại dạng sau đây có thể xuất hiện:

Hình 23

Khi đó, nếu nháy chuột vào nút **Đồng ý**, chương trình sẽ kết thúc còn nháy nút **Huỷ lệnh**, chương trình vẫn tiếp tục như bình thường.

GHI NHỚ

1. Các ngôn ngữ lập trình thường phân chia dữ liệu cần xử lí theo các kiểu khác nhau, với các phép toán có thể thực hiện trên từng kiểu dữ liệu đó.

2. Quá trình trao đổi dữ liệu hai chiều giữa người và máy tính khi chương trình hoạt động thường được gọi là giao tiếp hoặc tương tác người-máy.

Câu hỏi và bài tập

1. Hãy nêu ít nhất hai kiểu dữ liệu và một phép toán có thể thực hiện được trên một kiểu dữ liệu, nhưng phép toán đó không có nghĩa trên kiểu dữ liệu kia.

2. Dãy chữ số 2010 có thể thuộc những kiểu dữ liệu nào?

3. Hãy phân biệt ý nghĩa của các câu lệnh Pascal sau đây:

```
Writeln('5+20=','20+5');
```
và
```
Writeln('5+20=',20+5);
```

Hai lệnh sau có tương đương với nhau không? Tại sao?

```
Writeln('100');
```
và
```
Writeln(100);
```

4. Viết các biểu thức toán dưới đây bằng các kí hiệu trong Pascal:

a) $\frac{a}{b} + \frac{c}{d}$;

b) $ax^2 + bx + c$;

c) $\frac{1}{x} - \frac{a}{5}(b + 2)$;

d) $(a^2 + b)(1 + c)^3$.

5. Chuyển các biểu thức được viết trong Pascal sau đây thành các biểu thức toán:

a) `(a + b)*(a + b)- x/y;`

b) `b/(a*a + c);`

c) `a*a/((2*b + c)*(2* b + c));`

d) `1 + 1/2 + 1/(2*3) + 1/(3*4)+ 1/(4*5).`

6. Hãy xác định kết quả của các phép so sánh sau đây:

a) $15 - 8 \geq 3$;

b) $(20 - 15)^2 \neq 25$;

c) $11^2 = 121$;

d) $x > 10 - 3x$.

7. Viết các biểu thức ở bài tập 6 bằng các kí hiệu trong Pascal.

Bài thực hành 2

VIẾT CHƯƠNG TRÌNH ĐỂ TÍNH TOÁN

1. Mục đích, yêu cầu

- Luyện tập soạn thảo, chỉnh sửa chương trình, biên dịch, chạy và xem kết quả hoạt động của chương trình trong môi trường Turbo Pascal.
- Thực hành với các biểu thức số học trong chương trình Pascal.

2. Nội dung

BÀI 1. Luyện tập gõ các biểu thức số học trong chương trình Pascal.

a) Viết các biểu thức toán học sau đây dưới dạng biểu thức trong Pascal:

a) $15 \times 4 - 30 + 12$;

b) $\frac{10 + 5}{3 + 1} - \frac{18}{5 + 1}$;

c) $\frac{(10 + 2)^2}{(3 + 1)}$;

d) $\frac{(10 + 2)^2 - 24}{(3 + 1)}$.

Lưu ý: Chỉ được dùng dấu ngoặc đơn để nhóm các phép toán.

b) Khởi động Turbo Pascal và gõ chương trình sau để tính các biểu thức trên:

```
begin
  writeln('15* 4 - 30 + 12 = ', 15* 4 - 30 + 12);
  writeln('(10 + 5)/(3 + 1)-18/(5 + 1)= ',(10 + 5)/(3 + 1)-18/(5 + 1));
  writeln('(10 + 2)*(10 + 2)/(3 + 1)= ', (10 + 2)*(10 + 2)/(3 + 1));
  write('((10 + 2)*(10 + 2) - 24)/(3 + 1) =', ((10 + 2)*(10 + 2)-24)/(3 + 1));
  readln
end.
```

Lưu ý: Các biểu thức Pascal được đặt trong câu lệnh `writeln` để in ra kết quả. Em sẽ có cách viết khác sau khi làm quen với khái niệm biến ở bài 4.

c) Lưu chương trình với tên CT2.pas. Dịch, chạy chương trình và kiểm tra kết quả nhận được trên màn hình.

BÀI 2. Tìm hiểu các phép chia lấy phần nguyên và phép chia lấy phần dư với số nguyên. Sử dụng các câu lệnh tạm ngừng chương trình.

a) Mở tệp mới và gõ chương trình sau đây:

```
uses crt;
begin
    clrscr;
    writeln('16/3 =', 16/3);
    writeln('16 div 3 =', 16 div 3);
    writeln('16 mod 3 =', 16 mod 3);
    writeln('16 mod 3 = ', 16-(16 div 3)*3);
    writeln('16 div 3 = ', (16-(16 mod 3))/3);
end.
```

b) Dịch và chạy chương trình. Quan sát các kết quả nhận được và cho nhận xét về các kết quả đó.

c) Thêm các câu lệnh `delay(5000)` vào sau mỗi câu lệnh `writeln` trong chương trình trên. Dịch và chạy chương trình. Quan sát chương trình tạm dừng 5 giây sau khi in từng kết quả ra màn hình.

d) Thêm câu lệnh `readln` vào chương trình (trước từ khoá **end**). Dịch và chạy lại chương trình. Quan sát kết quả hoạt động của chương trình. Nhấn phím **Enter** để tiếp tục.

BÀI 3. Tìm hiểu thêm về cách in dữ liệu ra màn hình.

Mở lại tệp chương trình **CT2.pas** và sửa ba lệnh cuối (trước từ khoá **end**) thành:

```
writeln((10 + 5)/(3 + 1) - 18/(5 + 1):4:2);
writeln((10 + 2)*(10 + 2)/(3 + 1):4:2);
writeln(((10 + 2)*(10 + 2) - 24)/(3 + 1):4:2);
```

Dịch và chạy lại chương trình. Quan sát kết quả trên màn hình và rút ra nhận xét của em.

TỔNG KẾT

1. Kí hiệu của các phép toán số học trong Pascal: **+**, **–**, *****, **/**, **mod** và **div**.
2. Các lệnh thường được dùng để tạm ngừng chương trình:
   - `delay(x)` tạm ngừng chương trình trong vòng x phần nghìn giây, sau đó tự động tiếp tục chạy.
   - `readln` (không có tham số) tạm ngừng chương trình cho đến khi người dùng nhấn phím **Enter**.
3. Câu lệnh Pascal `writeln(<giá trị thực>:n:m)` được dùng để điều khiển cách in các số thực trên màn hình; trong đó *giá trị thực* là số hay biểu thức số thực và *n, m* là các số tự nhiên. *n* là độ rộng (tính bằng số chữ số) được in ra với *m* là số chữ số thập phân. Lưu ý rằng các kết quả in ra màn hình được căn thẳng lề phải.

BÀI 4

SỬ DỤNG BIẾN TRONG CHƯƠNG TRÌNH

1. Biến là công cụ trong lập trình

Hoạt động cơ bản của chương trình máy tính là xử lí dữ liệu. Trước khi được máy tính xử lí, mọi dữ liệu nhập vào đều được lưu trong bộ nhớ của máy tính. Ví dụ, nếu muốn cộng hai số *a* và *b*, trước hết hai số đó sẽ được nhập và lưu trong bộ nhớ máy tính, sau đó máy tính sẽ thực hiện phép cộng a + b.

Để chương trình luôn biết chính xác dữ liệu cần xử lí được lưu ở vị trí nào trong bộ nhớ, các ngôn ngữ lập trình cung cấp một công cụ lập trình rất quan trọng. Đó là *biến nhớ*, hay được gọi ngắn gọn là *biến*.

Trong lập trình, *biến* được dùng để *lưu trữ dữ liệu*. Dữ liệu do biến lưu trữ được gọi là *giá trị của biến*, nó có thể thay đổi trong khi thực hiện chương trình.

Chúng ta hãy xét một số ví dụ để hiểu vai trò của biến nhớ trong lập trình.

Ví dụ 1. Giả sử cần in kết quả của phép cộng 15 + 5 ra màn hình. Trong bài thực hành 2, ta đã biết có thể sử dụng câu lệnh Pascal sau đây:

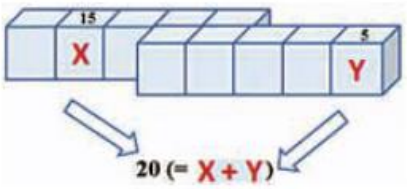
```
writeln(15+5);
```

Tình hình sẽ khác, nếu như hai số 15 và 5 được nhập trước từ bàn phím (hoặc là các kết quả tính toán trước đó). Sau khi nhận được các số 15 và 5, chương trình lưu trữ các số này ở những vị trí nào đó trong bộ nhớ. Chúng ta không biết giá trị của các số được nhập từ trước (hoặc các kết quả tính toán trung gian) nên không thể tính được tổng các số đã nhập vào để sử dụng lệnh in ra màn hình. Vì thế ta sử dụng hai biến *X* và *Y* để lưu giá trị của các số được nhập vào, tức 15 và 5 (hoặc kết quả tính toán trung gian), sau đó có thể sử dụng lệnh

```
writeln(X+Y);
```

để in kết quả ra màn hình.

Với việc sử dụng biến như trên, chương trình sẽ tự biết lấy các số 15 và 5 từ những vị trí nào trong bộ nhớ để thực hiện phép cộng (h. 24).



Hình 24

Ví dụ này cũng cho thấy, một cách hình ảnh, có thể xem hai biến *X* và *Y* như là "tên" của các vùng nhớ chứa các giá trị tương ứng.

Ví dụ 2. Giả sử cần tính giá trị của các biểu thức $\frac{100 + 50}{3}$ và $\frac{100 + 50}{5}$ sau đó in kết quả ra màn hình. Chúng ta có thể tính các biểu thức này một cách trực tiếp. Để ý rằng tử số trong các biểu thức là như nhau. Do đó có thể tính giá trị tử số và lưu tạm thời trong một biến trung gian *X*, sau đó thực hiện các phép chia. Về mặt toán học, điều này được thực hiện như sau (h. 25):

```
X = 100 + 50
Y = X/3
Z = X/5
```

Hình 25

2. Khai báo biến

Tất cả các biến dùng trong chương trình cần phải được khai báo ngay trong phần khai báo của chương trình. Việc khai báo biến gồm:

- Khai báo *tên biến*;
- Khai báo *kiểu dữ liệu* của biến.

Tên biến phải tuân theo quy tắc đặt tên của ngôn ngữ lập trình.

Ví dụ 3. Hình 26 là một ví dụ về cách khai báo biến trong Pascal:

```
var m,n : integer;
    S, dientich: real;
    thong_bao: string;
```

Hình 26

Trong ví dụ trên:

- **var** là từ khoá của ngôn ngữ lập trình dùng để khai báo biến,

- m, n là các biến có kiểu nguyên (integer),

- S, dientich là các biến có kiểu thực (real),

- thong_bao là biến kiểu xâu (string).

Tuỳ theo ngôn ngữ lập trình, cú pháp khai báo biến có thể khác nhau.

3. Sử dụng biến trong chương trình

Sau khi khai báo, ta có thể sử dụng các biến trong chương trình. Các thao tác có thể thực hiện với các biến là:

- *Gán* giá trị cho biến;

- *Tính toán* với giá trị của biến.

Kiểu dữ liệu của giá trị được gán cho biến thường phải trùng với kiểu của biến và *khi được gán một giá trị mới, giá trị cũ của biến bị xoá đi*. Ta có thể thực hiện việc gán giá trị cho biến tại bất kì thời điểm nào trong chương trình, do đó giá trị của biến có thể thay đổi.

Câu lệnh gán giá trị trong các ngôn ngữ lập trình thường có dạng:

Tên biến ← *Biểu thức cần gán giá trị cho biến*;

trong đó, dấu ← biểu thị phép gán. Ví dụ:

$x \leftarrow -c/b$ (biến x nhận giá trị bằng $-c/b$);

$x \leftarrow y$ (biến x được gán giá trị của biến y);

$i \leftarrow i + 5$ (biến i được gán giá trị hiện tại của i cộng thêm 5 đơn vị).

Việc gán giá trị cho biến còn có thể thực hiện bằng câu lệnh nhập dữ liệu.

Tuỳ theo ngôn ngữ lập trình, kí hiệu của câu lệnh gán cũng có thể khác nhau. Ví dụ, trong ngôn ngữ Pascal, người ta kí hiệu phép gán là dấu kép := để phân biệt với dấu bằng (=) của phép so sánh.

Ví dụ 4. Bảng dưới đây mô tả lệnh gán giá trị và tính toán với các biến trong Pascal:

Lệnh trong Pascal	Ý nghĩa
`X:=12;`	Gán giá trị số 12 vào biến nhớ *X*.
`X:=Y;`	Gán giá trị đã lưu trong biến nhớ *Y* vào biến nhớ *X*.
`X:=(a+b)/2;`	Thực hiện phép toán tính trung bình cộng hai giá trị nằm trong hai biến nhớ *a* và *b*. Kết quả gán vào biến nhớ *X*.
`X:=X+1;`	Tăng giá trị của biến nhớ *X* lên 1 đơn vị, kết quả gán trở lại biến *X*.

4. Hằng

Ngoài công cụ chính để lưu trữ dữ liệu là biến, các ngôn ngữ lập trình còn có công cụ khác là *hằng*. Khác với biến, hằng là đại lượng có *giá trị không đổi* trong suốt quá trình thực hiện chương trình.

Giống như biến, muốn sử dụng hằng, ta cũng cần phải khai báo tên của hằng. Tuy nhiên hằng phải được gán giá trị ngay khi khai báo.

Dưới đây là ví dụ khai báo hằng trong Pascal:

```
const pi = 3.14;
      bankinh = 2;
```

Hình 27

trong đó:

- `const` là từ khoá để khai báo hằng,
- Các hằng `pi`, `bankinh` được gán giá trị tương ứng là 3.14 và 2.

Với khai báo trên, để tính chu vi của hình tròn, ta có thể dùng câu lệnh sau:

```
chuvi:= 2*pi*bankinh;
```

Việc sử dụng hằng rất hiệu quả nếu giá trị của hằng (bán kính) được sử dụng trong nhiều câu lệnh của chương trình. Nếu sử dụng hằng, khi cần thay đổi giá trị, ta chỉ cần chỉnh sửa một lần, tại nơi khai báo mà không phải tìm và sửa trong cả chương trình.

Cần lưu ý rằng ta không thể dùng câu lệnh để thay đổi giá trị của hằng (như đối với biến) ở bất kì vị trí nào trong chương trình. Ví dụ, đối với các hằng `pi` và `bankinh` đã khai báo ở trên, các câu lệnh gán sau đây trong chương trình là không hợp lệ:

```
pi:= 3.1416;
bankinh:= bankinh + 2;
```

GHI NHỚ

1. Biến và hằng là các đại lượng được đặt tên dùng để *lưu trữ dữ liệu*. Giá trị của biến có thể thay đổi, còn giá trị của hằng được giữ nguyên trong suốt quá trình thực hiện chương trình.

2. Biến và hằng phải được khai báo trước khi sử dụng.

Câu hỏi và bài tập

1. Giả sử *A* được khai báo là biến với kiểu dữ liệu số thực, *X* là biến với kiểu dữ liệu xâu. Các phép gán sau đây có hợp lệ không?

a) A:= 4; b) X:= 3242;

c) X:= '3242'; d) A:= 'Ha Noi'.

2. Nêu sự khác nhau giữa biến và hằng. Cho một vài ví dụ về khai báo biến và hằng.

3. Giả sử ta đã khai báo một *hằng Pi* với giá trị 3.14. Có thể gán lại giá trị 3.1416 cho *Pi* trong phần thân chương trình được không? Tại sao?

4. Trong Pascal, khai báo nào sau đây là đúng?

a) `var tb: real;`

b) `var 4hs: integer;`

c) `const x : real;`

d) `var R = 30;`

5. Hãy liệt kê các lỗi nếu có trong chương trình dưới đây và sửa lại cho đúng:

```
var a,b := integer;
const  c:= 3;
begin
  a:= 200
  b:= a/c;
  write(b);
  readln
end.
```

6. Hãy cho biết kiểu dữ liệu của các biến cần khai báo dùng để viết chương trình để giải các bài toán dưới đây:

a) Tính diện tích *S* của hình tam giác với độ dài một cạnh *a* và chiều cao tương ứng *h* (*a* và *h* là các số tự nhiên được nhập vào từ bàn phím).

b) Tính kết quả *c* của phép chia lấy phần nguyên và kết quả *d* của phép chia lấy phần dư của hai số nguyên *a* và *b*.

Bài thực hành 3

KHAI BÁO VÀ SỬ DỤNG BIẾN

1. Mục đích, yêu cầu

Bước đầu làm quen cách khai báo và sử dụng biến trong chương trình.

2. Nội dung

Tìm hiểu các kiểu dữ liệu trong Pascal và cách khai báo biến với các kiểu dữ liệu:

Tên kiểu dữ liệu	Phạm vi giá trị
`Byte`	Số nguyên từ 0 đến 255.
`Integer`	Số nguyên từ − 32768 đến 32767.
`Real`	Số thực có giá trị tuyệt đối trong khoảng $2,9\times10^{-39}$ đến $1,7\times10^{38}$ và số 0.
`Char`	Kí tự trong bảng chữ cái.
`String`	Xâu kí tự gồm tối đa 255 kí tự.

Cú pháp khai báo biến:

`var` < *danh sách biến* > : < *kiểu dữ liệu* >;

trong đó:

– *danh sách biến* là danh sách một hoặc nhiều tên biến và được cách nhau bởi dấu phẩy (,).

– *kiểu dữ liệu* là một trong các kiểu dữ liệu của Pascal.

Ví dụ :

```
var X,Y: byte;
var So_nguyen: integer;
var Chieu_cao, Can_nang: real;
var Ho_va_Ten: string;
```

BÀI 1. Viết chương trình Pascal có khai báo và sử dụng biến.

Bài toán: Một cửa hàng cung cấp dịch vụ bán hàng thanh toán tại nhà. Khách hàng chỉ cần đăng kí số lượng mặt hàng cần mua, nhân viên cửa hàng sẽ giao hàng và nhận tiền thanh toán tại nhà khách hàng. Ngoài trị giá hàng hoá, khách hàng còn phải trả thêm phí dịch vụ. Hãy viết chương trình Pascal để tính tiền thanh toán trong trường hợp khách hàng chỉ mua một mặt hàng duy nhất.

Gợi ý: Công thức cần tính:

Tiền thanh toán = Đơn giá × Số lượng + Phí dịch vụ

a) Khởi động Pascal. Gõ chương trình sau và tìm hiểu ý nghĩa của từng câu lệnh trong chương trình:

```
program Tinh_tien;
uses crt;
var
   soluong: integer;
   dongia, thanhtien: real;
   thongbao: string;
const phi=10000;
begin
   clrscr;
   thongbao:= 'Tong so tien phai thanh toan :';
   {Nhap don gia va so luong hang}
   write('Don gia = '); readln(dongia);
   write('So luong = ');readln(soluong);
   thanhtien:= soluong*dongia+phi;
   (*In ra so tien phai tra*)
   writeln(thongbao,thanhtien:10:2);
   readln
end.
```

b) Lưu chương trình với tên **TINHTIEN.PAS**. Dịch và chỉnh sửa các lỗi gõ, nếu có.

c) Chạy chương trình với các bộ dữ liệu (đơn giá và số lượng) như sau (1000, 20), (3500, 200), (18500, 123). Kiểm tra tính đúng của các kết quả in ra.

d) Chạy chương trình với bộ dữ liệu (1, 35000). Quan sát kết quả nhận được. Hãy thử đoán lí do tại sao chương trình cho kết quả sai.

BÀI 2. Thử viết chương trình nhập các số nguyên *x* và *y*, in giá trị của *x* và *y* ra màn hình. Sau đó hoán đổi các giá trị của *x* và *y* rồi in lại ra màn hình giá trị của *x* và *y*.

Tham khảo chương trình sau:

```
program hoan_doi;
var x,y,z:integer;
begin
    read(x,y);
    writeln(x,' ',y);
    z:=x;
    x:=y;
    y:=z;
    writeln(x,' ',y);
    readln
end.
```

Lưu ý: Sau khi thực hiện câu lệnh `x:=y` thì biến `x` có giá trị bằng *y* và giá trị ban đầu của biến `x` (là *x*) không còn nữa. Do vậy, trong chương trình phải dùng biến `z` lưu trữ giá trị *x* rồi mới thực hiện việc gán giá trị *x* cho biến `y` nhờ câu lệnh `y:=z`.

TỔNG KẾT

1. Cú pháp khai báo biến trong Pascal:
   var *<danh sách biến>: <kiểu dữ liệu>;*
   trong đó *danh sách biến* gồm tên các biến và được cách nhau bởi dấu phẩy.
2. Cú pháp lệnh gán trong Pascal:
   <biến>:= <biểu thức>
3. Lệnh *read(<danh sách biến>)* hay *readln(<danh sách biến>)*, trong đó *danh sách biến* là tên các biến đã khai báo, được sử dụng để nhập dữ liệu từ bàn phím. Sau khi nhập dữ liệu cần nhấn phím **Enter** để xác nhận. Nếu giá trị nhập vào vượt quá phạm vi của biến thì kết quả tính toán thường sẽ sai.
4. Nội dung *chú thích* nằm trong cặp dấu { và } bị bỏ qua khi dịch chương trình. Các *chú thích* được dùng để làm cho chương trình dễ đọc, dễ hiểu. Ngoài ra có thể sử dụng cặp các dấu (* và *) để tạo chú thích.

BÀI 5

TỪ BÀI TOÁN ĐẾN CHƯƠNG TRÌNH

1. Bài toán và xác định bài toán

Bài toán là khái niệm quen thuộc trong các môn học như Toán, Vật lí,... Chẳng hạn tính tổng của các số tự nhiên từ 1 đến 100, tính quãng đường ô tô đi được trong 3 giờ với tốc độ 60km/giờ là những ví dụ về bài toán.

Tuy nhiên, hằng ngày ta thường gặp và giải quyết các công việc đa dạng hơn nhiều. Ví dụ, lập bảng cửu chương, lập bảng điểm của các bạn trong lớp hoặc so sánh chiều cao của hai bạn Long và Trang,... cũng là những ví dụ về bài toán. Như vậy, có thể hiểu:

Bài toán là một công việc hay một nhiệm vụ cần phải giải quyết.

Để giải quyết được một bài toán cụ thể, người ta cần *xác định bài toán*, tức là xác định rõ *các điều kiện cho trước* và *kết quả cần thu được*.

Ví dụ 1. Xét các bài toán tính diện tích hình tam giác, tìm đường đi tránh các điểm nghẽn giao thông trong giờ cao điểm và nấu một món ăn.

a) Để tính diện tích hình tam giác:

– Điều kiện cho trước: Một cạnh và đường cao tương ứng với cạnh đó;

– Kết quả cần thu được: Diện tích hình tam giác.

b) Đối với bài toán tìm đường đi tránh các điểm nghẽn giao thông:

– Điều kiện cho trước: Vị trí điểm nghẽn giao thông và các con đường có thể đi từ vị trí hiện tại tới vị trí cần tới;

– Kết quả cần thu được: Đường đi từ vị trí hiện tại tới vị trí cần tới mà không qua điểm nghẽn giao thông.

c) Đối với bài toán nấu một món ăn:

– Điều kiện cho trước: Các thực phẩm hiện có (trứng, mỡ, mắm, muối, rau,...);

– Kết quả cần thu được: Một món ăn.

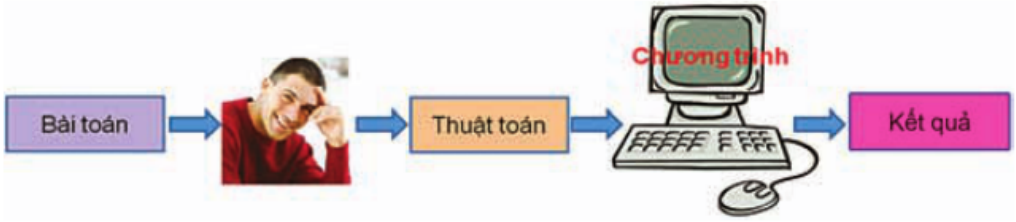
Xác định bài toán là bước đầu tiên và là bước rất quan trọng trong việc giải bài toán.

2. Quá trình giải bài toán trên máy tính

Việc dùng máy tính giải một bài toán chính là đưa cho máy tính dãy hữu hạn các thao tác đơn giản mà nó có thể thực hiện được để từ các điều kiện cho trước ta nhận được kết quả cần tìm.

Dãy hữu hạn các thao tác cần thực hiện để giải một bài toán được gọi là *thuật toán*.

Máy tính không thể tự mình tìm ra lời giải của các bài toán. Cách giải của một bài toán cụ thể, tức thuật toán, là tư duy sáng tạo của con người. Tuy nhiên, việc mô tả thuật toán chưa đủ đối với máy tính mà cần diễn đạt thuật toán dưới dạng mà máy tính có thể hiểu và thực hiện được. Kết quả diễn đạt thuật toán là chương trình được viết trong một ngôn ngữ lập trình nào đó. Máy tính sẽ chạy chương trình và cho ta lời giải của bài toán (h. 28).



Hình 28

Nói một cách khác, thuật toán là các bước để giải một bài toán, còn chương trình là thể hiện của thuật toán trong một ngôn ngữ lập trình cụ thể.

Quá trình giải bài toán trên máy tính gồm các bước sau:

- *Xác định bài toán*: Từ phát biểu của bài toán, ta xác định đâu là thông tin đã cho (INPUT) và đâu là thông tin cần tìm (OUTPUT).
- *Mô tả thuật toán*: Diễn tả cách giải bài toán bằng dãy các thao tác cần phải thực hiện.
- *Viết chương trình*: Dựa vào mô tả thuật toán ở trên, viết chương trình bằng một ngôn ngữ lập trình thích hợp.

Cần phải lưu ý rằng, để giải một bài toán có thể có nhiều thuật toán khác nhau, song mỗi thuật toán chỉ dùng để giải một bài toán cụ thể. Vì vậy, khi mô tả thuật toán, người ta thường chỉ ra cả điều kiện cho trước và kết quả cần nhận được để dễ nhận biết thuật toán đó dùng để giải bài toán nào.

3. Thuật toán và mô tả thuật toán

Trong phần này chúng ta sẽ tìm hiểu sâu hơn về khái niệm thuật toán.

Nhiều công việc chúng ta thường làm mà không phải suy nghĩ nhiều, tuy nhiên, nếu hệ thống lại, ta có thể thấy thực chất đó là những thuật toán. Đơn giản như việc pha trà mời khách có thể mô tả dưới dạng thuật toán như sau:

INPUT: Trà, nước sôi, ấm và chén.

OUTPUT: Chén trà đã pha để mời khách.

Bước 1. Tráng ấm, chén bằng nước sôi.

Bước 2. Cho trà vào ấm.

Bước 3. Rót nước sôi vào ấm và đợi khoảng 3 đến 4 phút.

Bước 4. Rót trà ra chén để mời khách.

Việc *liệt kê* các bước như trên là một cách thường dùng để *mô tả thuật toán*. Nếu không có mô tả gì khác trong thuật toán, các bước của thuật toán được thực hiện một cách tuần tự theo trình tự như đã được chỉ ra.

Mặc dù không được nêu rõ trong khái niệm thuật toán, song thuật toán phải được mô tả đủ cụ thể để bất kì đối tượng nào, với cùng khả năng và điều kiện như nhau, khi thực hiện thuật toán cũng đều đạt được kết quả như nhau. Để minh hoạ, chúng ta xét thêm một vài ví dụ:

Bài toán *"Giải phương trình bậc nhất dạng tổng quát* $bx + c = 0$*"*:

INPUT: Các số b và c.

OUTPUT: Nghiệm của phương trình bậc nhất.

Bước 1. Nếu $b = 0$ chuyển tới bước 3.

Bước 2. Tính nghiệm của phương trình $x = -\frac{c}{b}$ và chuyển tới bước 4.

Bước 3. *Nếu* $c \neq 0$, thông báo phương trình đã cho vô nghiệm. Ngược lại ($c = 0$), thông báo phương trình có vô số nghiệm.

Bước 4. Kết thúc.

Bài toán *"Làm món trứng tráng"*

INPUT: Trứng, dầu ăn, muối và hành.

OUTPUT: Trứng tráng.

Bước 1. Đập trứng, tách vỏ và cho trứng vào bát.

Bước 2. Cho một chút muối và hành tươi thái nhỏ vào bát trứng. Dùng đũa quấy mạnh để trộn đều trứng, muối, hành.

Bước 3. Cho một thìa dầu ăn vào chảo, đun nóng rồi đổ trứng đã trộn vào. Đun tiếp trong khoảng 1 phút.

Bước 4. Lật mặt trên của miếng trứng úp xuống dưới. Đun tiếp trong khoảng 1 phút.

Bước 5. Lấy trứng ra đĩa.

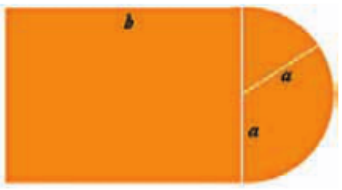
Rõ ràng, bất kì ai biết về các phép toán số học hay hiểu biết một chút về làm bếp, theo đúng trình tự và chỉ dẫn ở các bước trong các thuật toán nêu trên đều có thể tính ra nghiệm của phương trình đã cho hay tự làm cho mình món trứng tráng.

Tóm lại, có thể hiểu:

Thuật toán là dãy hữu hạn các thao tác cần thực hiện theo một trình tự xác định để thu được kết quả cần thiết từ những điều kiện cho trước.

4. Một số ví dụ về thuật toán

Ví dụ 2. Một hình A được ghép từ một hình chữ nhật với chiều rộng $2a$, chiều dài b và một hình bán nguyệt bán kính a như hình 29 dưới đây:



Hình 29

INPUT: Số a là $\frac{1}{2}$ chiều rộng của hình chữ nhật và là bán kính của hình bán nguyệt, b là chiều dài của hình chữ nhật.

OUTPUT: Diện tích của hình A.

Thuật toán đơn giản để tính diện tích hình A có thể gồm các bước sau:

Bước 1. $S_1 \leftarrow 2ab$ {Tính diện tích hình chữ nhật};

Bước 2. $S_2 \leftarrow \frac{\pi a^2}{2}$ {Tính diện tích hình bán nguyệt};

Bước 3. $S \leftarrow S_1 + S_2$ và kết thúc.

Lưu ý: Trong biểu diễn thuật toán, người ta cũng thường sử dụng kí hiệu ← để chỉ phép gán giá trị của một biểu thức cho một biến.

Ví dụ 3. Tính tổng của 100 số tự nhiên đầu tiên.

INPUT: Dãy 100 số tự nhiên đầu tiên: 1, 2, ..., 100.

OUTPUT: Giá trị của tổng 1 + 2 + ...+ 100.

Ý tưởng để giải bài toán trên là dùng một biến SUM để lưu giá trị của tổng. Việc tính SUM có thể được thực hiện như sau: Đầu tiên gán cho SUM giá trị bằng 0. Tiếp theo lần lượt thêm các giá trị 1, 2, 3, ..., 100 vào SUM. Vấn đề là ở chỗ tổ chức việc "lần lượt thêm vào" như thế nào? Cách dễ nhận thấy nhất là thực hiện liên tiếp 100 phép cộng:

Bước 1. SUM ← 0.

Bước 2. SUM ← SUM + 1.

...

Bước 101. SUM ← SUM + 100.

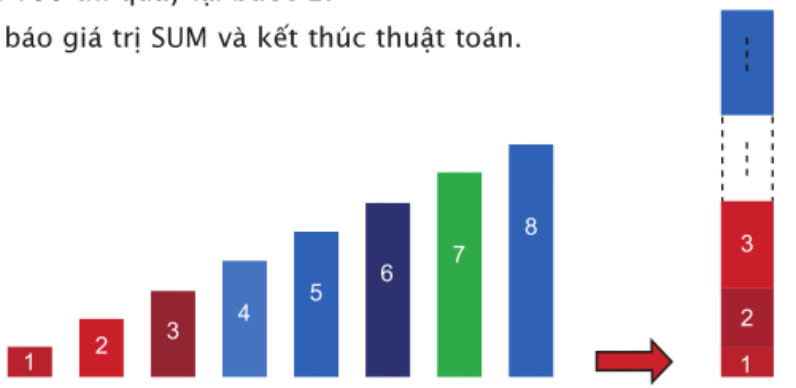
Tuy nhiên, việc mô tả thuật toán như trên là quá dài dòng (nhất là khi không chỉ tính tổng của 100 số mà số các số cần tính tổng lớn hơn nhiều). Để ý một chút ta có thể thấy trong tất cả các bước nêu trên đều chỉ có một phép toán được thực hiện: cộng thêm vào SUM lần lượt các giá trị 1, 2, 3,..., 100. Tức là chỉ có một thao tác "cộng" được lặp đi lặp lại 100 lần. Mặt khác, việc cộng thêm số *i* vào SUM chỉ được thực hiện khi *i* không vượt quá 100. Vì vậy, thuật toán tìm SUM có thể được mô tả ngắn gọn hơn như sau:

Bước 1. SUM ← 0; *i* ← 1.

Bước 2. SUM ← SUM + *i* ; *i* ← *i* + 1.

Bước 3. Nếu *i* ≤ 100 thì quay lại bước 2.

Bước 4. Thông báo giá trị SUM và kết thúc thuật toán.



Hình 30

Ví dụ 4. Đổi giá trị của hai biến x và y.

INPUT: Hai biến x, y có giá trị tương ứng là a và b.

OUTPUT: Hai biến x, y có giá trị tương ứng là b và a.

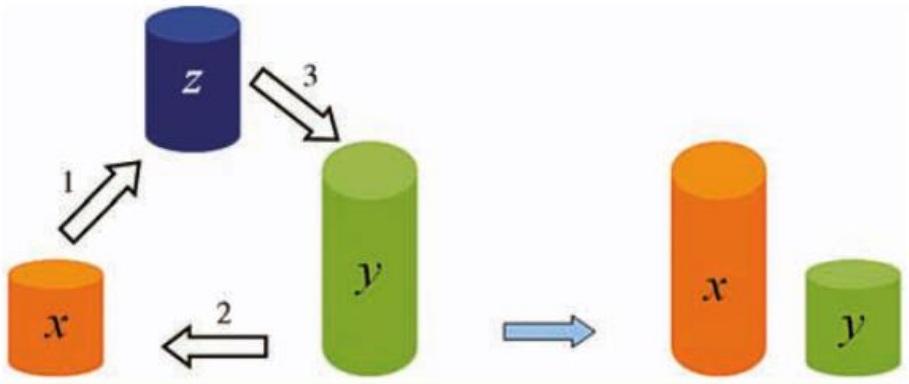
Trong bài 2 của bài thực hành 3, chúng ta đã tìm hiểu và viết chương trình hoán đổi các giá trị của hai biến x và y. Ví dụ này sẽ mô tả thuật toán để viết chương trình đó.

Ta không thể thực hiện trực tiếp hai phép gán: $x \leftarrow y$ và $y \leftarrow x$, bởi sau phép gán thứ nhất, giá trị của x đã bị thay bằng giá trị của y và kết quả của hai phép gán này là cả hai biến x và y cùng có giá trị ban đầu của biến y. Vì thế, cần dùng một biến trung gian, ví dụ biến z, để lưu tạm thời giá trị của biến x. Do vậy, ta có thuật toán sau:

Bước 1. $z \leftarrow x$ {Sau bước này giá trị của z sẽ bằng a} ;

Bước 2. $x \leftarrow y$ {Sau bước này giá trị của x sẽ bằng b} ;

Bước 3. $y \leftarrow z$ {Sau bước này giá trị của y sẽ bằng giá trị của z, chính là a, giá trị ban đầu của biến x}



Hình 31

Ví dụ 5. Cho hai số thực a và b. Hãy cho biết kết quả so sánh hai số đó dưới dạng "a lớn hơn b", "a nhỏ hơn b" hoặc "a bằng b".

INPUT: Hai số thực a và b.

OUTPUT: Kết quả so sánh.

Bài toán rất đơn giản. Nhìn qua, thuật toán sau đây dường như có thể giải quyết bài toán này:

Bước 1. Nếu $a > b$, kết quả là "a lớn hơn b".

Bước 2. Nếu $a < b$, kết quả là "a nhỏ hơn b"; Ngược lại, kết quả là "a bằng b" và kết thúc thuật toán.

Tuy nhiên, nếu thử lại các bước với $a = 6$ và $b = 5$, ta sẽ thấy sau bước 1 có kết quả "a lớn hơn b", nhưng đến bước 2, khi kiểm tra $a < b$ không thoả mãn ta lại có tiếp kết quả "a bằng b" và như thế ta nhận được hai kết quả.

Vì vậy, để có kết quả đúng, cần mô tả chính xác hơn điều kiện kết thúc thuật toán như sau:

Bước 1. Nếu $a > b$, kết quả là "a lớn hơn b" và *chuyển đến bước* 3.

Bước 2. Nếu $a < b$, kết quả là "a nhỏ hơn b"; Ngược lại, kết quả là "a bằng b".

Bước 3. Kết thúc thuật toán.

Ví dụ 6. Tìm số lớn nhất trong dãy A các số $a_1, a_2, ..., a_n$ cho trước.

Ta sẽ dùng biến *MAX* để lưu số lớn nhất của dãy A. Việc xác định *MAX* có thể được thực hiện như sau: Đầu tiên gán giá trị a_1 cho biến *MAX*. Tiếp theo, lần lượt so sánh các số $a_2, ..., a_n$ của dãy A với *MAX*. Nếu $a_i > MAX$, ta gán a_i cho *MAX*.

Từ đó, ta có thuật toán sau:

INPUT: Dãy A các số $a_1, a_2, ..., a_n$ ($n \geq 1$).

OUTPUT: Giá trị $MAX = \max\{a_1, a_2, ..., a_n\}$.

Thuật toán 1

Bước 1. $MAX \leftarrow a_1$; $i \leftarrow 1$.

Bước 2. Nếu $a_i > MAX$, $MAX \leftarrow a_i$.

Bước 3. $i \leftarrow i + 1$.

Bước 4. Nếu $i \leq n$ *thì quay lại bước 2.*

Bước 5. Thông báo giá trị *MAX* và kết thúc thuật toán.

Dưới đây minh hoạ thuật toán trên với trường hợp chọn thỏ nặng nhất trong bốn chú thỏ có trọng lượng tương ứng là 2, 1, 5, 3 ki-lô-gam.

a) Trước hết, ta gán *MAX* là trọng lượng của thỏ số 1, tức *MAX* = 2.

b) So sánh *MAX* với trọng lượng của thỏ số 2. Vì trọng lượng của thỏ số 2 nhẹ hơn trọng lượng của thỏ số 1, do đó *MAX* vẫn bằng 2.


MAX = 2

c) Tiếp theo, so sánh *MAX* với trọng lượng của thỏ số 3. Vì trọng lượng của thỏ số 3 lớn hơn *MAX*, do đó *MAX* được đặt lại bằng 5.


MAX = 5

d) Cuối cùng, so sánh *MAX* với trọng lượng của thỏ số 4. *MAX* lớn hơn trọng lượng của thỏ số 4, do đó *MAX* vẫn bằng 5. Kết quả, thỏ nặng nhất có trọng lượng là 5kg.


MAX = 5

GHI NHỚ

1. Xác định bài toán là việc xác định các điều kiện ban đầu (thông tin vào – INPUT) và các kết quả cần thu được (thông tin ra – OUTPUT).

2. Giải bài toán trên máy tính nghĩa là đưa ra dãy hữu hạn các thao tác đơn giản (thuật toán) để máy tính thực hiện và cho kết quả.

3. Quá trình giải một bài toán trên máy tính gồm các bước: xác định bài toán; xây dựng thuật toán; lập chương trình.

4. Thuật toán là dãy hữu hạn các thao tác cần thực hiện theo một trình tự xác định để nhận được kết quả cần tìm từ những điều kiện cho trước.

Câu hỏi và bài tập

1. Hãy chỉ ra INPUT và OUTPUT của các bài toán sau:

a) Xác định số học sinh trong lớp cùng mang họ Trần.

b) Tính tổng của các phần tử lớn hơn 0 trong dãy n số cho trước.

c) Tìm số các số có giá trị nhỏ nhất trong n số đã cho.

2. Giả sử x và y là các biến số. Hãy cho biết kết quả của việc thực hiện thuật toán sau:

Bước 1. $x \leftarrow x + y$

Bước 2. $y \leftarrow x - y$

Bước 3. $x \leftarrow x - y$

3. Cho trước ba số dương a, b và c. Hãy mô tả thuật toán cho biết ba số đó có thể là độ dài ba cạnh của một tam giác hay không.

4. Cho hai biến x và y. Hãy mô tả thuật toán đổi giá trị của hai biến nói trên (nếu cần) để x và y theo thứ tự có giá trị không giảm.

5. Hãy cho biết kết quả của thuật toán sau:

Bước 1. $SUM \leftarrow 0$; $i \leftarrow 0$.

Bước 2. *Nếu* $i > 100$ *thì chuyển tới bước 4.*

Bước 3. $i \leftarrow i+1$; $SUM \leftarrow SUM + i$. *Quay lại bước 2.*

Bước 4. *Thông báo giá trị SUM và kết thúc thuật toán.*

6. Hãy mô tả thuật toán giải bài toán tính tổng các phần tử của dãy số $A = (a_1, a_2, \ldots, a_n)$ cho trước.

BÀI 6

CÂU LỆNH ĐIỀU KIỆN

1. Hoạt động phụ thuộc vào điều kiện

Trong cuộc sống hằng ngày, chúng ta thực hiện phần lớn các hoạt động một cách *tuần tự* theo thói quen hoặc theo kế hoạch đã được xác định từ trước. Ví dụ:

- Mỗi sáng em thức dậy, tập thể dục buổi sáng, làm vệ sinh cá nhân, ăn sáng và đến trường,...
- Long thường đi đá bóng cùng các bạn vào sáng chủ nhật hằng tuần.

Tuy nhiên các hoạt động của con người thường bị tác động bởi sự thay đổi của các hoàn cảnh cụ thể. Nhiều hoạt động sẽ bị thay đổi, bị điều chỉnh cho phù hợp.

- "Nếu" em bị ốm, em sẽ không tập thể dục buổi sáng.
- "Nếu" trời không mưa vào ngày chủ nhật, Long đi đá bóng; ngược lại Long sẽ ở nhà giúp mẹ dọn dẹp nhà cửa.

Trong cuộc sống hằng ngày, từ "*nếu*" trong các câu trên được dùng để chỉ một "*điều kiện*". Các điều kiện đó là: "Em bị ốm" hoặc "Trời mưa". Hoạt động tiếp theo của em hoặc của bạn Long sẽ phụ thuộc vào các điều kiện đó có xảy ra hay không.

Có thể liệt kê được rất nhiều tình huống tương tự khác, chẳng hạn khi đi trên đường phố nếu gặp đèn đỏ, ta phải dừng lại; nếu khách đến nhà, em pha trà mời khách,...

Tóm lại, có những hoạt động chỉ được thực hiện khi một điều kiện cụ thể được xảy ra. Điều kiện thường là một sự kiện được mô tả sau từ "*nếu*".

2. Tính đúng hoặc sai của các điều kiện

Mỗi điều kiện nói trên được mô tả dưới dạng một phát biểu. Hoạt động tiếp theo phụ thuộc vào kết quả kiểm tra phát biểu đó đúng hay sai. Vậy kết quả kiểm tra có thể là gì?

Điều kiện	Kiểm tra	Kết quả	Hoạt động tiếp theo
Trời mưa?	Long nhìn ra ngoài trời và thấy trời mưa.	Đúng	Long ở nhà (không đi đá bóng).
Em bị ốm?	Buổi sáng thức dậy, em thấy mình hoàn toàn khoẻ mạnh.	Sai	Em tập thể dục buổi sáng như thường lệ.

Khi kết quả kiểm tra là *đúng*, ta nói điều kiện được *thoả mãn*, còn khi kết quả kiểm tra là *sai*, ta nói điều kiện *không thoả mãn*.

Ngoài những điều kiện gắn với các sự kiện đời thường như trên, trong Tin học chúng ta có thể gặp nhiều dạng điều kiện khác, ví dụ:

- *Nếu* nháy nút  *ở* góc trên, bên phải cửa sổ trên màn hình máy tính, (*thì*) cửa sổ sẽ được đóng lại.
- *Nếu* $X > 5$, (*thì* hãy) in giá trị của X ra màn hình.
- *Nếu* (ta) nhấn phím **Pause/Break**, (*thì*) chương trình (sẽ bị) ngừng.

3. Điều kiện và phép so sánh

Để so sánh hai giá trị số hoặc hai biểu thức có giá trị số, chúng ta sử dụng các kí hiệu toán học như: $=$, $\neq$, $<$, $\leq$, $>$ và $\geq$. Chúng ta cũng biết rằng các phép so sánh có kết quả *đúng* hoặc *sai*.

Các phép so sánh có vai trò rất quan trọng trong việc mô tả thuật toán và lập trình. Chúng thường được sử dụng để biểu diễn các điều kiện. Phép so sánh cho kết quả đúng có nghĩa điều kiện được thoả mãn; ngược lại, điều kiện không được thoả mãn.

Ví dụ 1. Ta muốn chương trình in ra màn hình giá trị lớn hơn trong số hai giá trị của các biến a và b. Khi đó giá trị của biến a hoặc b được in ra phụ thuộc vào phép so sánh $a > b$ là đúng hay sai:

"*Nếu* $a > b$, in giá trị của biến a ra màn hình;

ngược lại, in giá trị của biến b ra màn hình."

Trong trường hợp này điều kiện được biểu diễn bằng phép so sánh $a > b$.

Tương tự, khi giải phương trình bậc nhất dạng tổng quát $bx + c = 0$, để tính nghiệm của phương trình chúng ta cần kiểm tra các điều kiện được cho bằng các phép so sánh $b = 0$ và $c \neq 0$.

4. Cấu trúc rẽ nhánh

Về cơ bản, khi thực hiện chương trình, máy tính sẽ *thực hiện một cách tuần tự* các câu lệnh từ trên xuống dưới. Để thay đổi trình tự đó, ngôn ngữ lập trình có các câu lệnh cho phép máy tính thực hiện một câu lệnh nào đó, nếu một điều kiện cụ thể được thoả mãn; ngược lại, nếu điều kiện không được thoả mãn thì bỏ qua câu lệnh hoặc thực hiện một câu lệnh khác.

Ví dụ 2. Một hiệu sách thực hiện đợt khuyến mãi lớn với nội dung sau: Nếu mua sách với tổng số tiền ít nhất là 100 nghìn đồng, khách hàng sẽ được giảm 30% tổng số tiền phải thanh toán. Hãy mô tả hoạt động tính tiền cho khách.

Ta có thể mô tả hoạt động tính tiền cho khách hàng bằng các bước dưới đây:

Với mỗi khách hàng, thực hiện :

Bước 1. Tính tổng số tiền T khách hàng đã mua sách.

Bước 2. Nếu $T \geq 100000$, số tiền phải thanh toán là $70\% \times T$.

Bước 3. In hoá đơn.

Ví dụ 3. Cũng như ví dụ 2, nhưng chính sách khuyến mãi được thực hiện như sau : Nếu tổng số tiền không nhỏ hơn 100 nghìn đồng, khách hàng sẽ được giảm 30% tổng số tiền phải thanh toán. Trong trường hợp ngược lại, những khách hàng mua với tổng số tiền không đến 100 nghìn đồng sẽ chỉ được giảm 10%.

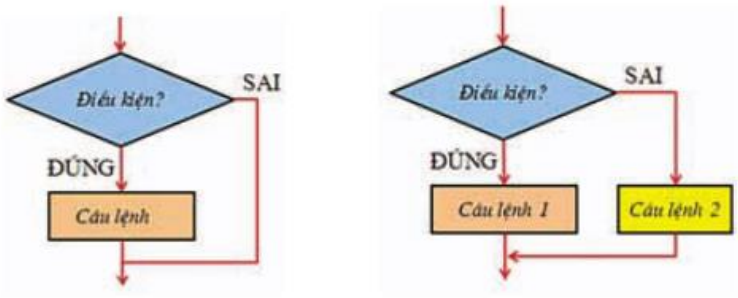
Khi đó, cần phải tính lại tiền cho khách trong cả hai trường hợp, tổng số tiền không nhỏ hơn 100 nghìn đồng và tổng số tiền nhỏ hơn 100 nghìn đồng. Thuật toán có thể được sửa lại như sau:

Bước 1. Tính tổng số tiền T khách hàng đã mua sách.

Bước 2. Nếu $T \geq 100000$, số tiền phải thanh toán là $70\% \times T$; Ngược lại, số tiền phải thanh toán là $90\% \times T$.

Bước 3. In hoá đơn.

Cách thể hiện hoạt động phụ thuộc vào điều kiện như trong ví dụ 2 được gọi là *cấu trúc rẽ nhánh dạng thiếu* (h. 32a), còn trong ví dụ 3 được gọi là *cấu trúc rẽ nhánh dạng đủ* (h. 32b). Cấu trúc rẽ nhánh giúp cho việc lập trình được linh hoạt hơn.



a) Cấu trúc rẽ nhánh dạng thiếu *b) Cấu trúc rẽ nhánh dạng đủ*

Hình 32

5. Câu lệnh điều kiện

Trong các ngôn ngữ lập trình, các cấu trúc rẽ nhánh được thể hiện bằng *câu lệnh điều kiện*.

Trong Pascal, câu lệnh điều kiện dạng thiếu được viết với các từ khoá **if** và **then** như sau:

if *<điều kiện>* **then** *<câu lệnh>*;

Khi gặp *câu lệnh điều kiện* này, chương trình sẽ kiểm tra *điều kiện*. Nếu *điều kiện* được thoả mãn, chương trình sẽ thực hiện *câu lệnh* sau từ khoá **then**. Ngược lại, *câu lệnh* đó bị bỏ qua.

Ví dụ 4. Nhiều chương trình yêu cầu người dùng nhập một số hợp lệ, chẳng hạn không lớn hơn 5, từ bàn phím. Chương trình đọc số, kiểm tra tính hợp lệ và thông báo nếu không hợp lệ. Khi đó các hoạt động của chương trình có thể biểu diễn bằng thuật toán sau đây:

Bước 1. Nhập số *a*;

Bước 2. *Nếu* *a* > 5 *thì* thông báo lỗi;

Các câu lệnh điều kiện dạng thiếu của Pascal dưới đây sẽ thể hiện thuật toán trên:

```
readln(a);
if a>5 then write('So da nhap khong hop le.');
```

Ví dụ 5. Cần viết chương trình tính kết quả của *a* chia cho *b*, với *a* và *b* là hai số bất kì. Phép tính chỉ thực hiện được khi $b \neq 0$. Chương trình cần kiểm tra giá trị của *b*, nếu $b \neq 0$ thì thực hiện phép chia; nếu $b = 0$ sẽ thông báo lỗi.

Nếu b ≠ 0 *thì* tính kết quả

ngược lại thì thông báo lỗi

Dưới đây là câu lệnh Pascal thể hiện cấu trúc rẽ nhánh dạng đủ nói trên:

```
if b<>0 then x:= a/b
        else write('Mau so bang 0, khong chia duoc');
```

Câu lệnh điều kiện **if…then…else…** mô tả trong ví dụ này là câu lệnh điều kiện dạng đầy đủ. Câu lệnh điều kiện dạng đầy đủ của Pascal có cú pháp:

if <*điều kiện*> **then** <*câu lệnh 1*> **else** <*câu lệnh 2*>;

Với câu lệnh điều kiện này, chương trình sẽ kiểm tra *điều kiện*. Nếu *điều kiện* được thoả mãn, chương trình sẽ thực hiện *câu lệnh 1* sau từ khoá **then**. Trong trường hợp ngược lại, *câu lệnh 2* sẽ được thực hiện.

GHI NHỚ

1. Cấu trúc rẽ nhánh được sử dụng để chỉ thị cho máy tính thực hiện các hoạt động khác nhau tuỳ theo một điều kiện cụ thể có được thoả mãn hay không. Cấu trúc rẽ nhánh có hai dạng: Dạng thiếu và dạng đầy đủ.

2. Trong lập trình, điều kiện trong cấu trúc rẽ nhánh thường được biểu diễn bằng các phép so sánh.

3. Mọi ngôn ngữ lập trình đều có câu lệnh điều kiện để thể hiện các cấu trúc rẽ nhánh.

Câu hỏi và bài tập

1. Em hãy nêu một vài ví dụ về các hoạt động hằng ngày phụ thuộc vào điều kiện.

2. Hãy cho biết các điều kiện hoặc biểu thức sau đây cho kết quả đúng hay sai:

a) 123 là số chia hết cho 3.

b) Nếu ba cạnh a, b và c của một tam giác thoả mãn $c^2 = a^2 + b^2$ thì tam giác đó có một góc vuông.

c) $15^2 > 200$.

d) $x^2 < 1$.

3. Hai người bạn cùng chơi trò đoán số. Một người nghĩ trong đầu một số tự nhiên nhỏ hơn 10. Người kia đoán xem bạn đã nghĩ số gì. Nếu đoán đúng, người đoán sẽ được cộng thêm 1 điểm, nếu sai sẽ không được cộng điểm. Luân phiên nhau nghĩ và đoán. Sau 10 lần, ai được nhiều điểm hơn, người đó sẽ thắng.

Hãy phát biểu quy tắc thực hiện một nước đi ở trò chơi. Hoạt động nào sẽ được thực hiện, nếu điều kiện của quy tắc đó thoả mãn? Hoạt động nào sẽ được thực hiện, nếu điều kiện của quy tắc đó không thoả mãn?

4. Một trò chơi máy tính rất hứng thú đối với các em nhỏ là hứng trứng. Một quả trứng rơi từ một vị trí ngẫu nhiên trên cao. Người chơi dùng các phím mũi tên → hoặc ← để điều khiển một chiếc khay di chuyển theo chiều ngang để hứng quả trứng. Mỗi lần người chơi nhấn phím mũi tên (→ hoặc ←) thì chiếc khay sẽ dịch chuyển (sang phải hoặc sang trái) một đơn vị khoảng cách. Nếu người chơi không nhấn phím nào hoặc nhấn phím khác hai phím nói trên thì chiếc khay sẽ đứng yên.

Điều kiện để điều khiển chiếc khay trong trò chơi là gì ? Hoạt động nào sẽ được thực hiện, nếu điều kiện đó thoả mãn? Hoạt động nào sẽ được thực hiện, nếu điều kiện đó không thoả mãn ?

5. Các câu lệnh Pascal sau đây được viết đúng hay sai ?

a) `if x:= 7 then a = b;`

b) `if x > 5; then a:= b;`

c) `if x > 5 then a:= b; m:= n;`

d) `if x > 5 then a:= b; else m:= n;`

6. Với mỗi câu lệnh sau đây giá trị của biến *X* sẽ là bao nhiêu, nếu trước đó giá trị của X bằng 5 ?

a) `if (45 mod 3) = 0 then X:= X + 1;`

b) `if X > 10 then X:= X + 1;`

Bài thực hành 4

SỬ DỤNG LỆNH ĐIỀU KIỆN IF...THEN

1. Mục đích, yêu cầu

- Luyện tập sử dụng câu lệnh điều kiện **if…then**.
- Rèn luyện kĩ năng ban đầu về đọc các chương trình đơn giản và hiểu được ý nghĩa của thuật toán sử dụng trong chương trình.

2. Nội dung

Bảng dưới đây cho biết các câu lệnh Pascal để thực hiện cấu trúc rẽ nhánh:

Dạng thiếu

nếu *<điều kiện>* **thì** *<câu lệnh>*;

if *<điều kiện>* **then** *<câu lệnh>*;

Dạng đầy đủ

nếu *<điều kiện>* **thì** *<câu lệnh 1>* **nếu không thì** *<câu lệnh 2>*;

if *<điều kiện>* **then** *<câu lệnh 1>* **else** *<câu lệnh 2>*;

BÀI 1. Viết chương trình nhập hai số nguyên *a* và *b* khác nhau từ bàn phím và in hai số đó ra màn hình theo thứ tự không giảm.

a) Mô tả thuật toán để giải bài toán đã cho (tham khảo thêm bài tập 4, bài 5).

b) Gõ chương trình sau đây :

```
program Sap_xep;
uses crt;
var A, B: integer;
begin
    clrscr;
    write('Nhap so A: '); readln(A);
    write('Nhap so B: '); readln(B);
```

```
    if A<B then writeln(A,'   ', B) else writeln(B,'   ',A);
    readln
end.
```

c) Tìm hiểu ý nghĩa của các câu lệnh trong chương trình. Nhấn Alt+F9 để dịch và sửa lỗi gõ, nếu có. Nhấn Ctrl+F9 để chạy chương trình với các bộ dữ liệu (12, 53), (65, 20) để thử chương trình. Cuối cùng lưu chương trình với tên Sap_xep.pas.

BÀI 2. Viết chương trình nhập chiều cao của hai bạn Long và Trang, in ra màn hình kết quả so sánh chiều cao của hai bạn, chẳng hạn "Bạn Long cao hơn". Tham khảo thuật toán trong ví dụ 5, bài 5.

a) Gõ chương trình sau đây:

```
program Ai_cao_hon;
uses crt;
var Long, Trang: Real;
begin
    clrscr;
    write('Nhap chieu cao cua Long:'); readln(Long);
    write('Nhap chieu cao cua Trang:'); readln(Trang);
    if Long>Trang then writeln('Ban Long cao hon');
    if Long<Trang then writeln('Ban Trang cao hon')
    else writeln('Hai ban cao bang nhau');
    readln
end.
```

b) Lưu chương trình với tên Aicaohon.pas. Dịch và sửa lỗi gõ, nếu có.

c) Chạy chương trình với các bộ dữ liệu (1.5, 1.6) và (1.6, 1.5) và (1.6, 1.6). Quan sát các kết quả nhận được và nhận xét. Hãy tìm chỗ chưa đúng trong chương trình.

d) Sửa lại chương trình để có kết quả đúng: chỉ in ra màn hình *một* thông báo kết quả.

Tham khảo và tìm hiểu ý nghĩa của đoạn chương trình sau đây:

```
if Long > Trang then writeln('Ban Long cao hon')
else if Long < Trang then writeln('Ban Trang cao hon')
else writeln('Hai ban cao bang nhau');
```

Lưu ý: Trong đoạn chương trình tham khảo trên chúng ta đã sử dụng hai câu lệnh **if…then** *lồng nhau*:

if *<điều kiện 1>* **then** *<câu lệnh 1>* **else**

if *<điều kiện 2 >* **then** *<câu lệnh 2>* **else** *<câu lệnh 3>*;

BÀI 3. Dưới đây là chương trình nhập ba số dương *a*, *b* và *c* từ bàn phím, kiểm tra và in ra màn hình kết quả kiểm tra ba số đó có thể là độ dài các cạnh của một tam giác hay không.

Ý tưởng : Ba số dương *a*, *b* và *c* là độ dài các cạnh của một tam giác khi và chỉ khi $a + b > c$, $b + c > a$ và $c + a > b$.

```
program Ba_canh_tam_giac;
uses crt;
var a, b, c: real;
begin
   Clrscr;
   write('Nhap ba so a, b va c:'); readln(a,b,c);
   if (a+b>c) and (b+c>a) and (c+a>b) then
   writeln('a, b va c la 3 canh cua mot tam giac!')
   else writeln('a, b, c khong la 3 canh cua 1 tam giac!');
   Readln
end.
```

Tìm hiểu ý nghĩa của các câu lệnh trong chương trình, soạn, dịch và chạy chương trình với các số tuỳ ý.

Lưu ý: Trong chương trình trên chúng ta sử dụng từ khoá **and** để kết hợp nhiều phép so sánh đơn giản thành một phép so sánh phức hợp. Giá trị của phép so sánh này là *đúng* khi và chỉ khi *tất cả* các phép so sánh đơn giản đều có giá trị *đúng*. Ngược lại, chỉ cần một phép so sánh thành phần có giá trị *sai* thì nó có giá trị *sai*.

TỔNG KẾT

1. Câu lệnh điều kiện dạng thiếu : **if** *<điều kiện>* **then** *<câu lệnh 1>* ;
2. Câu lệnh điều kiện dạng đầy đủ:

   if *<điều kiện>* **then** *<câu lệnh 1>* **else** *<câu lệnh 2>*;
3. Có thể sử dụng các câu lệnh **if…then** lồng nhau.
4. Sử dụng từ khoá **and** có thể kết hợp nhiều phép so sánh đơn giản thành một phép so sánh phức hợp. Giá trị của phép so sánh này là *đúng* khi và chỉ khi *tất cả* các phép so sánh đơn giản đều *đúng*. Ngược lại, nó có giá trị *sai*.

   Ví dụ: (a > 0) **and** (a <= 5)

   Từ khoá **or** cũng được sử dụng để kết hợp nhiều phép so sánh đơn giản. Giá trị của phép so sánh này chỉ *sai* khi *tất cả* các phép so sánh thành phần đều *sai*. Ngược lại, nó có giá trị *đúng*.

   Ví dụ: `(a>0) or (a<=5)`

BÀI 7

CÂU LỆNH LẶP

1. Các công việc phải thực hiện nhiều lần

Trong cuộc sống hằng ngày, nhiều hoạt động được thực hiện lặp đi lặp lại nhiều lần.

Có những hoạt động mà chúng ta thường thực hiện lặp đi lặp lại với một số lần nhất định và biết trước, chẳng hạn đánh răng mỗi ngày hai lần, mỗi ngày tắm một lần,... Chúng ta còn lặp lại những công việc với số lần không thể xác định trước: học cho đến khi thuộc bài, nhặt từng cọng rau cho đến khi xong,...

Khi viết chương trình máy tính cũng vậy. Để chỉ dẫn cho máy tính thực hiện đúng công việc, trong nhiều trường hợp ta cũng cần phải yêu cầu máy tính thực hiện một số câu lệnh nhiều lần.

2. Câu lệnh lặp - một lệnh thay cho nhiều lệnh

Ví dụ 1. Giả sử ta cần vẽ ba hình vuông có cạnh 1 đơn vị như hình 33. Mỗi hình vuông là ảnh dịch chuyển của hình bên trái nó một khoảng cách 2 đơn vị. Do đó, ta chỉ cần lặp lại thao tác vẽ hình vuông ba lần. Việc vẽ hình có thể thực hiện được bằng thuật toán sau đây:

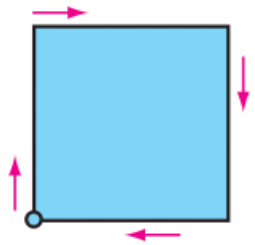
Bước 1. Vẽ hình vuông (vẽ liên tiếp bốn cạnh và trở về đỉnh ban đầu).

Bước 2. Nếu số hình vuông đã vẽ được ít hơn 3, di chuyển bút vẽ về bên phải 2 đơn vị và trở lại bước 1; ngược lại, kết thúc thuật toán.

Hình 33

Riêng với bài toán vẽ một hình vuông (h. 34), thao tác chính là vẽ bốn cạnh bằng nhau, hay lặp lại bốn lần thao tác vẽ một đoạn thẳng. Sau mỗi lần

vẽ đoạn thẳng, thước kẻ được quay một góc 90° sang phải tại vị trí của bút vẽ. Thuật toán sau đây sẽ mô tả các bước để vẽ hình vuông:



Hình 34

Bước 1. $k \leftarrow 1$ (k là số đoạn thẳng cần vẽ).

Bước 2. Vẽ đoạn thẳng 1 đơn vị độ dài và quay thước 90° sang phải. $k \leftarrow k+1$.

Bước 3. Nếu $k \leq 4$ thì trở lại bước 2; Ngược lại, kết thúc thuật toán.

Lưu ý rằng, biến k được sử dụng như là biến đếm để ghi lại số cạnh đã vẽ được.

Ví dụ 2. Giả sử cần tính tổng của 100 số tự nhiên đầu tiên, tức là tính:

$$S = 1 + 2 + 3 + ... + 100.$$

Hoạt động chính khi giải bài toán này là thực hiện phép cộng. Thuật toán trong ví dụ 3, bài 5 đã mô tả việc thực hiện lặp lại phép cộng 100 lần.

Cách mô tả các hoạt động lặp trong thuật toán như trong ví dụ trên được gọi là *cấu trúc lặp*.

Mọi ngôn ngữ lập trình đều có "cách" để chỉ thị cho máy tính thực hiện cấu trúc lặp với một câu lệnh. Đó là các *câu lệnh lặp*.

3. Ví dụ về câu lệnh lặp

Các ngôn ngữ lập trình thường có nhiều dạng câu lệnh lặp. Câu lệnh lặp thường gặp trong Pascal có dạng:

`for` <*biến đếm*> `:=` <*giá trị đầu*> `to` <*giá trị cuối*> `do` <*câu lệnh*>;

Trong đó, `for`, `to`, `do` là các từ khoá, *biến đếm* là biến kiểu nguyên, giá trị đầu và giá trị cuối là các giá trị nguyên.

Câu lệnh lặp sẽ thực hiện *câu lệnh* nhiều lần, mỗi lần là một vòng lặp. Số vòng lặp là biết trước và bằng

giá trị cuối − giá trị đầu + 1.

Khi thực hiện, ban đầu biến đếm sẽ nhận giá trị bằng *giá trị đầu*, sau mỗi vòng lặp, biến đếm được tự động tăng thêm một đơn vị cho đến khi bằng *giá trị cuối*.

Ví dụ 3. Chương trình sau sẽ in ra màn hình thứ tự lần lặp:

```
program Lap;
var i: Integer;
begin
    for i:= 1 to 10 do
         writeln('Day la lan lap thu ',i);
    Readln
end.
```

Ví dụ 4. Để in một chữ "O" trên màn hình, ta có thể sử dụng lệnh:

```
writeln('O');
```

Nếu muốn viết chương trình ghi nhận các vị trí của một quả trứng rơi từ trên cao xuống, ta có thể lặp lại lệnh trên nhiều lần (ví dụ, 10 lần) như trong chương trình sau:

```
uses crt;
var i: integer;
begin
    clrscr;
    for i:= 1 to 10 do
         begin writeln('O'); delay(100) end;
    Readln
end.
```

Dịch và chạy chương trình này, ta sẽ thấy kết quả như ở hình 35 dưới đây:

Hình 35

Lưu ý. Trong ví dụ 4, các *câu lệnh đơn giản* `writeln('O')` và `delay(100)` được đặt trong hai từ khoá **`begin`** và **`end`** để tạo thành một *câu lệnh ghép* trong Pascal. Từ đây về sau, khi nói *câu lệnh*, ta có thể hiểu đó là câu lệnh đơn hoặc câu lệnh ghép.

Trong thực tế, để có mười kết quả, chúng ta phải thực hiện mười lần một hoạt động (có thể với những điều kiện khác nhau). Máy tính thực hiện công việc xử lí thông tin thay cho con người và cũng phải thực hiện ngần ấy hoạt động. Câu lệnh lặp góp phần giúp giảm nhẹ công sức viết chương trình máy tính.

4. Tính tổng và tích bằng câu lệnh lặp

Ví dụ 5. Chương trình sau đây sẽ tính tổng của *N* số tự nhiên đầu tiên, với *N* là số tự nhiên được nhập vào từ bàn phím (xem ví dụ 2).

```
program Tinh_tong;
var N, i: integer;
    S: longint;
begin
    write('Nhap so N = '); readln(N);
    S:=0;
    for i:= 1 to N do S:= S + i;
    writeln('Tong cua ',N,' so tu nhien dau tien S = ',S);
    Readln
end.
```

Lưu ý. Vì với *N* lớn, tổng của *N* số tự nhiên đầu tiên có thể rất lớn nên trong chương trình trên ta sử dụng một kiểu dữ liệu mới của Pascal, kiểu `longint` (được khai báo cho biến *S*). Đây cũng là kiểu số nguyên, nhưng có thể lưu các số nguyên trong phạm vi từ −2147483648 đến 2147483647, lớn hơn nhiều so với kiểu `integer` (chỉ từ −32768 đến 32767).

Ví dụ 6. Ta kí hiệu N! là tích N số tự nhiên đầu tiên, đọc là N giai thừa:

$$N! = 1.2.3...N$$

Dưới đây là chương trình tính N! với N là số tự nhiên được nhập vào từ bàn phím. Chương trình sử dụng một câu lệnh lặp **`for…do`**:

```
program Tinh_Giai_thua;
var N, i: Integer;
    P: longint;
begin
    write('N = '); readln(N);
    P:= 1;
    for i:= 1 to N do P:=P*i;
    writeln(N,'! = ',P);
    Readln
end.
```

Lưu ý. Vì N! là số rất lớn so với N, do vậy cần lưu ý khai báo biến chứa giá trị của nó đủ lớn.

GHI NHỚ

1. Cấu trúc lặp được sử dụng để chỉ thị cho máy tính thực hiện lặp lại một vài hoạt động nào đó cho đến khi một điều kiện nào đó được thoả mãn.

2. Mọi ngôn ngữ lập trình đều có các câu lệnh lặp để thể hiện cấu trúc lặp.

3. Ngôn ngữ Pascal thể hiện cấu trúc lặp với số lần lặp cho trước bằng câu lệnh `for...do`.

Câu hỏi và bài tập

1. Cho một vài ví dụ về hoạt động được thực hiện lặp lại trong cuộc sống hằng ngày.

2. Hãy cho biết tác dụng của câu lệnh lặp với số lần biết trước.

3. Khi thực hiện câu lệnh lặp, máy tính kiểm tra một điều kiện. Với lệnh lặp

`for` *<biến đếm>*:= *<giá trị đầu>* **`to`** *<giá trị cuối>* **`do`** *<câu lệnh>*;

của Pascal, điều kiện cần phải kiểm tra là gì?

4. Sau khi thực hiện đoạn chương trình sau, giá trị của biến j bằng bao nhiêu?

```
j: = 0;
for i:= 0 to 5 do j:= j + 2;
```

5. Các câu lệnh Pascal sau có hợp lệ không, vì sao?

a) `for i:= 100 to 1 do writeln('A');`

b) `for i:= 1.5 to 10.5 do writeln('A');`

c) `for i= 1 to 10 do writeln('A');`

d) `for i:=1 to 10 do; writeln('A');`

e) `var x :real; begin for x:= 1 to 10 do writeln('A'); end.`

6. Hãy mô tả thuật toán để tính tổng sau đây :

$$A = \frac{1}{1.3} + \frac{1}{2.4} + \frac{1}{3.5} + ... + \frac{1}{n(n+2)}.$$

7. Hãy sửa lại chương trình ở ví dụ 5 để tính tổng của các số tự nhiên chia hết cho 3 không vượt quá N cho trước.

Bài thực hành 5

SỬ DỤNG LỆNH LẶP FOR...DO

1. Mục đích, yêu cầu

- Viết chương trình Pascal có câu lệnh lặp **for...do**.
- Tiếp tục nâng cao kĩ năng đọc hiểu chương trình.

2. Nội dung

BÀI 1.

a) Hãy gõ chương trình ở ví dụ 5 bài 7 và thực hiện với các giá trị N = 3, 4, 5,... để kiểm tra kết quả tính tổng của N số tự nhiên đầu tiên.

b) Hãy thay đoạn chương trình

```
for i := 1 to N do S := S + i;
   writeln('Tong ',N,' so tu nhien dau tien S = ', S);
```

bằng đoạn chương trình

```
for i := 1 to N do
if i mod 2 = 0 then S := S + i;
writeln('Tong cac so chan nho hon hoac bang ',N,' la = ', S);
```

Cho biết kết quả thực hiện chương trình với N = 8, 9, 10 là gì?

BÀI 2. Viết chương trình in ra màn hình bảng nhân của một số từ 1 đến 9, số được nhập từ bàn phím và dừng màn hình để có thể quan sát kết quả.

Hình 36

a) Gõ chương trình sau đây:

```
uses crt;
var N, i: integer;
begin
    clrscr;
    write('Nhap so N = '); readln(N);
    writeln;
    writeln('Bang nhan ',N);
    writeln;
    for i:= 1 to 10 do writeln(N,' x ', i:2,' = ', N*i:3);
    readln
end.
```

b) Tìm hiểu ý nghĩa của các câu lệnh trong chương trình, dịch chương trình và sửa lỗi, nếu có.

c) Chạy chương trình với các giá trị nhập vào lần lượt bằng 1, 2,..., 10. Quan sát kết quả nhận được trên màn hình.

BÀI 3. Chỉnh sửa chương trình để làm đẹp kết quả trên màn hình.

Kết quả của chương trình nhận được trong bài 2 có hai nhược điểm sau đây:

- Các hàng kết quả quá sát nhau nên khó đọc;
- Các hàng kết quả không được cân đối với hàng tiêu đề.

Hình 37

Nên sửa chương trình bằng cách chèn thêm một hàng trống giữa các hàng kết quả và đẩy các hàng này sang phải một khoảng cách nào đó.

a) Chỉnh sửa câu lệnh lặp của chương trình như sau:

```
for i:= 1 to 10 do
begin
    GotoXY(5,WhereY);
    writeln(N,' x ', i:2,' = ', N*i:3);
    writeln
end;
```

Lưu ý:

- Chỉ sử dụng được các lệnh `GotoXY`, `WhereX` và `WhereY` sau khi khai báo thư viện `crt` của Pascal.

- Màn hình máy tính được chia thành các cột và các hàng, được tính bắt đầu từ góc trên bên trái. Câu lệnh `GotoXY(a,b)` có tác dụng đưa con trỏ về *cột a*, *hàng b*.

- `WhereX` cho biết số thứ tự của *cột* và `WhereY` cho biết số thứ tự của *hàng* đang có con trỏ. Ví dụ `GotoXY(5,WhereY)` đưa con trỏ về vị trí cột 5 của hàng hiện tại.

b) Dịch và chạy chương trình với các giá trị gõ vào từ bàn phím. Quan sát kết quả nhận được trên màn hình.

BÀI 4. Cũng như câu lệnh **`if`**, có thể dùng câu lệnh **`for`** lồng trong một câu lệnh **`for`** khác khi thực hiện lặp. Sử dụng các câu lệnh **`for...do`** lồng nhau để in ra màn hình các số từ 0 đến 99 theo dạng bảng như hình sau:

Hình 38

a) Tìm hiểu chương trình sau đây :

```
program Tao_bang;
uses crt;
var
i: byte; {chi so cua hang}
j: byte; {chi so cua cot}
begin
clrscr; {xoa man hinh}
```

```
for i:= 0 to 9 do  {viet theo tung hang}
   begin
        for j:= 0 to 9 do {viet theo tung cot tren moi hang}
        write(10*i+j:4); {viet cac so ij ra man hinh}
        writeln;   {xuong hang moi}
   end;  {xong hang thu i}
readln {dung chuong trinh de xem ket qua}
end.
```

b) Gõ và chạy chương trình, quan sát kết quả trên màn hình. Sử dụng thêm các câu lệnh `GotoXY(a,b)` để điều chỉnh (một cách tương đối) bảng kết quả ra giữa màn hình.

TỔNG KẾT

1. Cấu trúc lặp với số lần lặp biết trước được thể hiện bằng câu lệnh Pascal **`for…do`**.
2. Giống như các câu lệnh rẽ nhánh **`if…then`**, các câu lệnh **`for…do`** cũng có thể lồng trong nhau. Khi đó các *biến đếm* trong các câu lệnh lặp phải khác nhau.
3. Câu lệnh `GotoXY(a,b)` có tác dụng đưa con trỏ về *cột a, hàng b*. `WhereX` cho biết số thứ tự của *cột* và `WhereY` cho biết số thứ tự của *hàng* đang có con trỏ.
4. Có thể kết hợp câu lệnh `GotoXY(a,b)` với các hàm chuẩn `WhereX` và `WhereY` để điều khiển vị trí của con trỏ trên màn hình.

BÀI 8

LẶP VỚI SỐ LẦN CHƯA BIẾT TRƯỚC

1. Các hoạt động lặp với số lần chưa biết trước

Trong bài trước chúng ta đã làm quen với các hoạt động lặp và cách chỉ thị cho máy tính thực hiện các hoạt động lặp với số lần đã được xác định trước. Chẳng hạn, để tính tổng các số nguyên từ 1 đến 100, ta có thể viết câu lệnh lặp để máy tính thực hiện phép cộng 100 lần.

Trong thực tế có nhiều hoạt động được thực hiện lặp đi lặp lại với số lần chưa được biết trước.

Ví dụ 1. Một ngày chủ nhật, bạn Long gọi điện cho Tuấn. Không có ai nhấc máy. Long quyết định gọi thêm hai lần nữa. Nếu vẫn không có ai nhấc máy thì chắc là không có ai ở nhà. Như vậy Long đã biết trước là mình sẽ lặp lại hoạt động gọi điện thêm hai lần. Một ngày khác, Long quyết định cứ 10 phút gọi điện một lần cho Tuấn cho đến khi nào có người nhấc máy. Lần này Long sẽ lặp lại hoạt động gọi điện mấy lần? Chưa thể biết trước được, có thể một lần, có thể hai hoặc nhiều hơn nữa. Điều kiện để kết thúc hoạt động lặp đó là *có người nhấc máy*.

Ví dụ 2. Nếu cộng lần lượt n số tự nhiên đầu tiên ($n = 1, 2, 3,...$), ta sẽ được các kết quả $T_1 = 1$, $T_2 = 1 + 2$, $T_3 = 1 + 2 + 3$,... tăng dần. Cần cộng bao nhiêu số tự nhiên đầu tiên để ta nhận được tổng T_n nhỏ nhất lớn hơn 1000 ? Trong trường hợp này, để quyết định thực hiện phép cộng với số tiếp theo hay dừng, trong từng bước cần phải kiểm tra tổng đã lớn hơn 1000 hay chưa.

Chúng ta hãy tìm hiểu các bước của thuật toán trong ví dụ này một cách cụ thể hơn. Kí hiệu S là tổng cần tìm và ta có thuật toán như sau:

Bước 1. $S \leftarrow 0$, $n \leftarrow 1$.

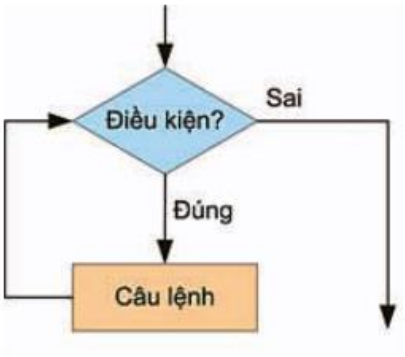
Bước 2. Nếu $S \leq 1000$, $S \leftarrow S + n$; *Ngược lại,* chuyển tới bước 4.

Bước 3. $n \leftarrow n + 1$ và quay lại bước 2.

Bước 4. In kết quả: S và n là số tự nhiên nhỏ nhất sao cho $S > 1000$. Kết thúc thuật toán.

Việc thực hiện phép cộng ở thuật toán trên được lặp lại với số lần chưa biết trước, phụ thuộc vào một *điều kiện* ($S \leq 1000$) và chỉ dừng khi điều kiện đó sai.

Nói chung, việc lặp lại một nhóm hoạt động với số lần chưa xác định trước phụ thuộc vào một *điều kiện* cụ thể có được thoả mãn hay không và có thể được mô tả như hình 39.



Hình 39

Để viết chương trình chỉ dẫn máy tính thực hiện các hoạt động lặp như trong các ví dụ trên, ta có thể sử dụng câu lệnh có dạng *lặp với số lần chưa biết trước*. Nói chung các ngôn ngữ lập trình đều có câu lệnh lặp dạng này.

2. Ví dụ về lệnh lặp với số lần chưa biết trước

Trong Pascal câu lệnh lặp với số lần chưa biết trước có dạng:

`while` *<điều kiện>* `do` *<câu lệnh>*;

trong đó:

- *điều kiện* thường là một phép so sánh;
- *câu lệnh* có thể là câu lệnh đơn giản hay câu lệnh ghép.

Câu lệnh lặp này được thực hiện như sau:

Bước 1. Kiểm tra *điều kiện*.

Bước 2. Nếu *điều kiện* SAI, *câu lệnh* sẽ bị bỏ qua và việc thực hiện lệnh lặp kết thúc. Nếu *điều kiện* đúng, thực hiện *câu lệnh* và quay lại bước 1.

Ví dụ 3. Chúng ta biết rằng, nếu n ($n > 0$) càng lớn thì $\frac{1}{n}$ càng nhỏ, nhưng luôn luôn lớn hơn 0.

Với giá trị nào của n thì $\frac{1}{n} < 0.005$ hoặc $\frac{1}{n} < 0.003$? Chương trình dưới đây tính số n nhỏ nhất để $\frac{1}{n}$ nhỏ hơn một sai số cho trước:

```
uses crt;
var x: real;
    n: integer;
const sai_so=0.003;
begin
    clrscr;
    x:= 1; n:= 1;
    while x >= sai_so do begin x:= 1/n; n:= n + 1 end;
    writeln('So n nho nhat de 1/n < ', sai_so:6:5, 'la ',n-1);
    readln
end.
```

Nếu chạy chương trình này, ta sẽ nhận được kết quả $n = 334$ (và sai số là 0.00299). Thay điều kiện *sai_so = 0.003* lần lượt bằng các điều kiện *sai_so = 0.002* và *sai_so = 0.001*, ta nhận được các kết quả $n = 501$ và $n = 1001$. Có thể kiểm tra các kết quả này bằng một phép chia đơn giản.

Ví dụ 4. Chương trình Pascal dưới đây thể hiện thuật toán tính tổng n số trong ví dụ 2:

```
var S, n: integer;
begin
    S:= 0; n:= 1;
    while S <= 1000 do
         begin  S:= S + n ; n:= n + 1 end;
    writeln('So n nho nhat de tong > 1000 la ', n-1);
    writeln('Tong dau tien > 1000 la ', S);
    readln
end.
```

Nếu chạy chương trình này ta sẽ nhận được $n = 45$ và tổng đầu tiên lớn hơn 1000 là 1035.

Ví dụ 5. Để viết chương trình tính tổng $T = 1 + \frac{1}{2} + \frac{1}{3} + ... + \frac{1}{100}$ ta có thể sử dụng lệnh lặp với số lần lặp biết trước **for…do**:

```
T:= 0;
for i:= 1 to 100 do T:= T + 1/i;
writeln(T);
```

Nếu sử dụng lệnh lặp **while…do**, đoạn chương trình dưới đây cũng cho cùng một kết quả:

```
T:= 0;
i:= 1;
while i <= 100 do begin T:= T + 1/i; i:= i + 1 end;
writeln(T);
```

Ví dụ này cho thấy rằng chúng ta có thể sử dụng câu lệnh **while…do** thay cho câu lệnh **for…do**.

3. Lặp vô hạn lần – Lỗi lập trình cần tránh

Khi viết chương trình sử dụng cấu trúc lặp cần chú ý tránh tạo nên vòng lặp không bao giờ kết thúc. Chẳng hạn, chương trình dưới đây sẽ lặp lại vô tận:

```
var a:integer;
begin
    a:=5;
    while a<6 do writeln('A');
end.
```

Trong chương trình trên, giá trị của biến *a* luôn luôn bằng 5, điều kiện *a<6* luôn luôn đúng nên lệnh `writeln('A')` luôn được thực hiện.

Do vậy, khi thực hiện vòng lặp, giá trị các biến trong *điều kiện* của câu lệnh phải được thay đổi để sớm hay muộn giá trị của *điều kiện* được chuyển từ *đúng* sang *sai*. Chỉ như thế chương trình mới không "rơi" vào những "vòng lặp vô tận". Trong các ví dụ 3, 4, 5 sau mỗi vòng lặp giá trị các biến `x`, `S`, `i` được thay đổi (`x` giảm đi còn `S`, `i` tăng lên) làm cho giá trị của điều kiện (`x >= sai_so` ở ví dụ 3; `S <= 1000` ở ví dụ 4; `i <= 100` ở ví dụ 5) sẽ buộc phải thay đổi khi `x` đủ nhỏ còn `S`, `i` đủ lớn.

GHI NHỚ

1. Ngoài cấu trúc lặp với số lần lặp biết trước, các ngôn ngữ lập trình còn có các câu lệnh lặp với số lần chưa biết trước.

2. while…do là câu lệnh lặp với số lần chưa biết trước trong Pascal.

Câu hỏi và bài tập

1. Nêu một vài ví dụ về hoạt động lặp với số lần chưa biết trước.

2. Hãy phát biểu sự khác biệt giữa câu lệnh lặp với số lần biết trước và câu lệnh lặp với số lần chưa biết trước.

3. Hãy tìm hiểu các thuật toán sau đây và cho biết khi thực hiện thuật toán, máy tính sẽ thực hiện bao nhiêu vòng lặp? Khi kết thúc, giá trị của S bằng bao nhiêu? Viết chương trình Pascal thể hiện các thuật toán đó.

a) Thuật toán 1

Bước 1. $S \leftarrow 10$, $x \leftarrow 0.5$.

Bước 2. Nếu $S \leq 5.2$, chuyển tới bước 4.

Bước 3. $S \leftarrow S - x$ và quay lại bước 2.

Bước 4. Thông báo S và kết thúc thuật toán.

b) Thuật toán 2

Bước 1. $S \leftarrow 10$, $n \leftarrow 0$.

Bước 2. Nếu $S \geq 10$, chuyển tới bước 4.

Bước 3. $n \leftarrow n + 3$, $S \leftarrow S - n$ quay lại bước 2.

Bước 4. Thông báo S và kết thúc thuật toán.

4. Hãy tìm hiểu mỗi đoạn lệnh sau đây và cho biết với đoạn lệnh đó chương trình thực hiện bao nhiêu vòng lặp ? Hãy rút ra nhận xét của em.

a)
```
S:= 0; n:= 0;
  while S <= 10 do
      begin n:= n + 1; S:= S + n end;
```

b)
```
S:= 0; n:= 0;
  while S <= 10 do
      n:= n + 1; S:= S + n;
```

5. Hãy chỉ ra lỗi trong các câu lệnh sau đây:

a) `X:= 10; while X:= 10 do X:= X + 5;`

b) `X:= 10; while X = 10 do X = X + 5;`

c) `S:= 0; n:= 0; while S <= 10 do n:= n + 1; S:= S + n;`

Bài thực hành 6

SỬ DỤNG LỆNH LẶP WHILE...DO

1. Mục đích, yêu cầu

- Viết chương trình Pascal sử dụng câu lệnh lặp với số lần chưa biết trước.
- Rèn luyện khả năng đọc chương trình, tìm hiểu tác dụng của các câu lệnh.

2. Nội dung

BÀI 1. Viết chương trình sử dụng lệnh lặp **while...do** để tính trung bình n số thực $x_1, x_2, x_3,..., x_n$. Các số n và $x_1, x_2, x_3,..., x_n$ được nhập vào từ bàn phím.

Ý tưởng: Sử dụng một biến đếm và lệnh lặp **while...do** để nhập và cộng dần các số vào một biến kiểu số thực cho đến khi nhập đủ n số.

a) Mô tả thuật toán của chương trình, các biến dự định sẽ sử dụng và kiểu của chúng.

b) Gõ chương trình sau đây và lưu chương trình với tên Tinh_TB.pas:

```
program Tinh_Trung_binh;
uses crt;
var n, dem : Integer;
    x, TB : real;
begin
    clrscr;
    dem:= 0; TB:= 0;
    write('Nhap so cac so can tinh n = '); readln(n);
    while dem < n do
          begin
                dem:= dem + 1;
                write('Nhap so thu ', dem,' = '); readln(x);
                TB:= TB + x;
          end;
    TB:= TB/n;
    writeln('Trung binh cua ', n,' so la = ', TB:10:3);
    writeln('Nhan Enter de thoat ...');
    readln
end.
```

c) Đọc và tìm hiểu ý nghĩa của từng câu lệnh. Dịch chương trình và sửa lỗi, nếu có. Chạy chương trình với các bộ dữ liệu được gõ từ bàn phím và kiểm tra kết quả nhận được.

d) Viết lại chương trình bằng cách sử dụng câu lệnh **for…do** thay cho câu lệnh **while…do**.

BÀI 2. Tìm hiểu chương trình nhận biết một số tự nhiên *N* được nhập vào từ bàn phím có phải là số nguyên tố hay không.

Ý tưởng: Kiểm tra lần lượt *N* có chia hết cho các số tự nhiên $2 \le i \le N - 1$ hay không. Kiểm tra tính chia hết bằng phép chia lấy phần dư (*mod*).

a) Đọc và tìm hiểu ý nghĩa của từng câu lệnh trong chương trình sau đây:

```
uses Crt;
var n, i: integer;
begin
   clrscr;
   write('Nhap vao mot so nguyen: '); readln(n);
   If n <= 1 then writeln(n,' khong la so nguyen to')
   else
        begin
         i:= 2;
         while (n mod i<>0) do i:= i + 1;
         if i = n then writeln(n,' la so nguyen to !')
         else writeln(n,' khong phai la so nguyen to!');
        end;
   readln
end.
```

b) Gõ, dịch và chạy thử chương trình với một vài độ chính xác khác nhau.

TỔNG KẾT

Câu lệnh lặp **while…do** có dạng

while <*điều kiện*> **do** <*câu lệnh*>;

BÀI 9

LÀM VIỆC VỚI DÃY SỐ

1. Dãy số và biến mảng

Ví dụ 1. Để khảo sát mức độ phân hoá giàu nghèo của một địa phương, người ta đã tiến hành thu thập thông tin về thu nhập của từng hộ gia đình trong địa phương đó. Cần viết chương trình tính mức thu nhập trung bình của các hộ gia đình trong địa phương và độ lệch giữa mức thu nhập của từng hộ gia đình so với mức thu nhập trung bình.

Rõ ràng việc giải bài toán trên gồm hai bước cơ bản:

1. Tính thu nhập trung bình bằng cách lấy tổng thu nhập của tất cả các hộ gia đình chia cho tổng số hộ.

2. Lần lượt lấy thu nhập của từng hộ trừ đi giá trị trung bình ở bước 1 để tính độ lệch giữa mức thu nhập của hộ đó so với mức thu nhập trung bình.

Giả sử số hộ gia đình được khảo sát là 50. Đoạn chương trình sau có thể giúp giải quyết bài toán trên:

```
Thunhap_TB := 0;
for i := 1 to 50 do
  begin
    write('Thu nhap cua gia dinh thu ',i);
    readln(a);
    Thunhap_TB := Thunhap_TB + a
  end;
Thunhap_TB := Thunhap_TB/50;
for i := 1 to 50 do
  begin
    write('Thu nhap cua gia dinh thu ',i);
    readln(a);
    writeln('Do lech so voi thu nhap TB cua ho ',i,' la: ', a - Thunhap_TB)
  end;
```

Do tại mỗi thời điểm một biến chỉ lưu được một giá trị duy nhất nên trong đoạn chương trình trên, mỗi khi cần tới thu nhập của hộ gia đình nào ta lại

phải thực hiện câu lệnh `readln(a)` để nhập mức thu nhập của hộ đó vào biến `a`. Cần lưu ý thêm, thao tác nhập mức thu nhập của các hộ gia đình từ bàn phím chiếm phần lớn thời gian trong quá trình thực hiện đoạn chương trình trên, mà ta lại phải thực hiện công việc đó hai lần.

Để chỉ phải nhập một lần, ta có thể khai báo nhiều biến, mỗi biến dùng để lưu trữ thu nhập của một hộ gia đình. Ví dụ, trong Pascal ta cần nhiều câu lệnh khai báo và nhập dữ liệu như sau:

```
Var Thunhap_1, Thunhap_2, Thunhap_3,...: real;
...
Read(Thunhap_1); Read(Thunhap_2); Read(Thunhap_3);...
```

Chú ý rằng địa phương cần khảo sát có bao nhiêu hộ gia đình thì cần viết đủ chừng ấy khai báo và câu lệnh nhập mức thu nhập - một công việc hoàn toàn không hề thú vị!

Nếu có cách nào để lưu nhiều dữ liệu liên quan với nhau (như *Thunhap_1*, *Thunhap_2, Thunhap_3,...* ở trên) bằng *một biến duy nhất* và đánh "số thứ tự" cho các giá trị đó, ta có thể sử dụng quy luật tăng hay giảm của "số thứ tự" và một vài câu lệnh lặp để xử lí dữ liệu một cách đơn giản hơn, chẳng hạn:

– Với *i chạy từ 1 đến 50*: hãy nhập *Thunhap_i*;

– Với *i chạy từ 1 đến 50*: hãy so sánh *Thunhap_TB* với *Thunhap_i*.

Để giúp giải quyết vấn đề trên, hầu hết các ngôn ngữ lập trình đều có một kiểu dữ liệu được gọi là *kiểu mảng*.

Dữ liệu kiểu mảng là *một tập hợp hữu hạn các phần tử có thứ tự, mọi phần tử đều có cùng một kiểu dữ liệu*, gọi là kiểu của phần tử. Việc sắp thứ tự được thực hiện bằng cách gán cho mỗi phần tử một *chỉ số*:

Hình 40

Trong bài này, chúng ta chỉ xét các mảng có các phần tử kiểu số nguyên hoặc số thực.

Khi khai báo một biến có kiểu dữ liệu là kiểu mảng, biến đó được gọi là *biến mảng*. Có thể nói rằng, khi sử dụng biến mảng, về thực chất chúng ta sắp thứ tự theo chỉ số các biến có *cùng kiểu* dưới một tên duy nhất.

Giá trị của biến mảng là một *mảng*, tức một dãy số (số nguyên, hoặc số thực) có thứ tự, mỗi số là giá trị của biến thành phần tương ứng.

2. Ví dụ về biến mảng

Để làm việc với các dãy số nguyên hay số thực, chúng ta phải khai báo biến mảng có kiểu tương ứng trong phần khai báo của chương trình.

Cách khai báo biến mảng trong các ngôn ngữ lập trình có thể khác nhau, nhưng luôn cần chỉ rõ: tên biến mảng, số lượng phần tử, kiểu dữ liệu chung của các phần tử.

Ví dụ, cách khai báo đơn giản một biến mảng trong ngôn ngữ Pascal như sau:

```
var Chieucao: array[1..50] of real;
```

Với câu lệnh trên, ta đã khai báo một biến có tên *Chieucao* gồm 50 phần tử, mỗi phần tử là biến có kiểu số thực.

Từ ví dụ trên, có thể thấy cách khai báo mảng trong Pascal như sau:

Tên mảng : **array** [*<chỉ số đầu>*.. *<chỉ số cuối>*] **of** *<kiểu dữ liệu>*

trong đó *chỉ số đầu* và *chỉ số cuối* là hai số nguyên thoả mãn

chỉ số đầu ≤ *chỉ số cuối* và *kiểu dữ liệu* có thể là `integer` hoặc `real`.

Ví dụ 2. Tiếp tục với ví dụ 1, thay vì khai báo các biến *Thunhap_1*, *Thunhap_2*, *Thunhap_3*,... để lưu mức thu nhập của các hộ gia đình, ta khai báo biến mảng *Thunhap* như sau:

```
var Thunhap: array[1..50] of real;
```

Cách khai báo và sử dụng biến mảng như trên có lợi gì?

Trước hết, có thể thay rất nhiều câu lệnh nhập và in dữ liệu ra màn hình bằng một câu lệnh lặp. Chẳng hạn, ta có thể viết

```
for i:= 1 to 50 do readln(Thunhap[i]);
```

để nhập mức thu nhập của các hộ gia đình. Thay vì phải viết 50 câu lệnh khai báo và 50 câu lệnh nhập, ta chỉ cần viết hai câu lệnh là đủ và kết quả đạt được là như nhau.

Ta còn có thể sử dụng biến mảng một cách rất hiệu quả trong xử lí dữ liệu. Để so sánh mức thu nhập của các hộ gia đình với một giá trị nào đó, ta cũng chỉ cần một câu lệnh lặp, chẳng hạn

```
for i:=1 to 50 do
if Thunhap[i] > Thunhap_TB then
   writeln('Ho dan ',i,' thu nhap tren trung binh');
```

Điều này giúp tiết kiệm rất nhiều thời gian và công sức viết chương trình.

Ví dụ 3. Giả sử chúng ta cần viết chương trình nhập điểm từng môn học cho các học sinh trong một lớp và tính toán trên các điểm đó. Vì mỗi học sinh có thể có nhiều điểm theo từng môn học: điểm Toán, điểm Văn, điểm Lí,... nên để xử lí đồng thời các loại điểm này, ta có thể khai báo nhiều biến mảng:

```
var DiemToan: array[1..50] of real;
var DiemVan: array[1..50] of real;
var DiemLi: array[1..50] of real;
```

hay

```
var DiemToan, DiemVan, DiemLi: array[1..50] of real;
```

Ngoài ra, ta cũng có thể xử lí điểm thi của *một* học sinh cụ thể (ví dụ như tính điểm trung bình của Lan, tính điểm cao nhất của Châu,...) hoặc tính điểm trung bình của cả lớp,...

DiemLi	8	6	7	6	...	...	...	...
DiemVan	7	8	6	9	...	...	...	...
DiemToan	9	7	8	7	...	...	...	...
Chỉ số	1	2	3	4	...	*i*	...	50

Hình 41

Sau khi một mảng đã được khai báo, chúng ta có thể làm việc với các phần tử của nó như làm việc với một biến thông thường như gán giá trị, đọc giá trị và thực hiện các tính toán với các giá trị đó thông qua tên của biến mảng và chỉ số tương ứng của phần tử. Chẳng hạn, trong các câu lệnh sau *A[i]* là phần tử thứ *i* của biến mảng *A*.

Ta có thể gán giá trị cho các phần tử của mảng A bằng câu lệnh gán:

```
A[1]:= 5;
A[2]:= 8+m;
```

hoặc nhập dữ liệu từ bàn phím bằng câu lệnh lặp:

```
for i:= 1 to 5 do readln(A[i]);
```

3. Tìm giá trị lớn nhất và nhỏ nhất của dãy số

Ví dụ 4. Viết chương trình nhập *N* số nguyên từ bàn phím và in ra màn hình số nhỏ nhất và số lớn nhất. *N* cũng được nhập từ bàn phím (xem lại thuật toán trong ví dụ 6, bài 5).

Trước hết ta khai báo biến *N* để nhập số các số nguyên sẽ được nhập vào. Sau đó khai báo *N* biến lưu các số được nhập vào như là các phần tử của một biến mảng *A*. Ngoài ra, cần khai báo một biến *i* làm biến đếm cho các lệnh lặp và biến *Max* để lưu số lớn nhất, *Min* để lưu số nhỏ nhất.

Phần khai báo của chương trình có thể như sau:

```
program MaxMin;
uses crt;
var i, n, Max, Min: integer;
    A: array[1..100] of integer;
```

Phần thân chương trình sẽ tương tự dưới đây:

```
begin
    clrscr;
    write('Hay nhap so phan tu cua day so, N = '); readln(n);
    writeln('Nhap cac phan tu cua day so:');
    for i:= 1 to n do
          begin
          write('a[',i,'] ='); readln(a[i]);
          end;
    Max:= a[1]; Min:= a[1];
    for i:= 2 to n do
          begin
                if Max < a[i] then Max:= a[i];
                if Min > a[i] then Min:= a[i]
          end;
    writeln('Phan tu Max la: ',Max);
    writeln('Phan tu Min la: ',Min);
    readln
end.
```

Trong chương trình này, chúng ta hãy lưu ý điểm sau: Số tối đa các phần tử của mảng (còn gọi là *kích thước* của mảng) phải được khai báo bằng một số cụ thể (ở đây là 100, mặc dù số các phần tử nhập vào sau này có thể nhỏ hơn nhiều so với 100).

GHI NHỚ

1. Dữ liệu kiểu mảng là một tập hợp hữu hạn các phần tử có thứ tự và mọi phần tử của mảng đều có cùng một kiểu dữ liệu.

2. Việc gán giá trị, nhập giá trị và tính toán với các giá trị của một phần tử trong biến mảng được thực hiện thông qua chỉ số tương ứng của phần tử đó.

3. Sử dụng các biến mảng và câu lệnh lặp giúp cho việc viết chương trình được ngắn gọn và dễ dàng hơn.

Câu hỏi và bài tập

1. Hãy nêu các lợi ích của việc sử dụng biến mảng trong chương trình.

2. Các khai báo biến mảng sau đây trong Pascal đúng hay sai?

a)
```
var X: Array[10,13] Of Integer;
```

b)
```
var X: Array[5..10.5] Of Real;
```

c)
```
var X: Array[3.4..4.8] Of Integer;
```

d)
```
var X: Array[10..1] Of Integer;
```

e)
```
var X: Array[4..10] Of Real;
```

3. Phát biểu "Có thể xem biến mảng là một biến được tạo từ nhiều biến có cùng kiểu, nhưng chỉ dưới một tên duy nhất" đúng hay sai?

4. Câu lệnh khai báo biến mảng sau đây máy tính có thực hiện được không?

```
var N: integer;
 A: array[1..N] of real;
```

5. Viết chương trình sử dụng biến mảng để nhập từ bàn phím các phần tử của một dãy số. Độ dài của dãy cũng được nhập từ bàn phím.

6. Viết chương trình sử dụng biến mảng để giải quyết bài toán nêu trong ví dụ 1.

7. *Độ lệch chuẩn* là một khái niệm rất quan trọng trong đánh giá dữ liệu thống kê. Giả sử ta có bộ dữ liệu thống kê gồm n phần tử có giá trị tương ứng là $x_1, x_2, ..., x_n$. Kí hiệu x_{TB} là giá trị trung bình của $x_1, x_2, ..., x_n$. Khi đó, độ lệch chuẩn của bộ dữ liệu trên được tính theo công thức sau:

$$\sqrt{\frac{\sum_{i=1}^{n}(x_i - x_{TB})^2}{n}}$$

(căn bậc hai của tổng các bình phương độ lệch từng phần tử so với giá trị trung bình chia cho số phần tử).

Hãy viết chương trình tính độ lệch chuẩn của dữ liệu thống kê về mức thu nhập của các hộ gia đình nêu trong ví dụ 1.

Lưu ý: Giá trị của độ lệch chuẩn phản ánh mức độ phân tán của dữ liệu thống kê. Áp dụng cho dữ liệu thống kê về mức thu nhập của các hộ gia đình trong một địa phương nêu trong ví dụ 1, nếu giá trị độ lệch chuẩn lớn thì mức độ phân hoá giàu nghèo của địa phương càng rõ rệt (độ chênh lệch về mức thu nhập của người giàu và người nghèo lớn).

Bài thực hành 7

XỬ LÍ DÃY SỐ TRONG CHƯƠNG TRÌNH

1. Mục đích, yêu cầu

- Làm quen với việc khai báo và sử dụng các biến mảng.
- Ôn luyện cách sử dụng câu lệnh lặp **for…do**.
- Củng cố các kĩ năng đọc, hiểu và chỉnh sửa chương trình.

2. Nội dung

BÀI 1. Viết chương trình nhập điểm của các bạn trong lớp. Sau đó in ra màn hình số bạn đạt kết quả học tập loại giỏi, khá, trung bình và kém (theo tiêu chuẩn từ 8.0 trở lên đạt loại giỏi, từ 6.5 đến 7.9 đạt loại khá, từ 5.0 đến 6.4 đạt loại trung bình và dưới 5.0 xếp loại kém).

a) Xem lại các ví dụ 2 và ví dụ 3, bài 8 về cách sử dụng và khai báo biến mảng trong Pascal.

b) Liệt kê các biến dự định sẽ sử dụng trong chương trình. Tìm hiểu phần khai báo dưới đây và tìm hiểu tác dụng của từng biến:

```
program Phanloai;
uses crt;
var i, n, Gioi, Kha, Trungbinh, Kem: integer;
    A: array[1..100] of real;
```

c) Gõ phần khai báo trên vào máy tính và lưu tệp với tên **Phanloai.pas**. Tìm hiểu các câu lệnh trong phần thân chương trình dưới đây:

```
begin
    clrscr;
    write('Nhap so cac ban trong lop, n = '); readln(n);
    writeln('Nhap diem :');
    for i:= 1 to n do
          begin write(i,'. '); readln(a[i])  end;
```

```
    Gioi:= 0; Kha:= 0; Trungbinh:= 0; Kem:= 0;
  for i:= 1 to n do
    begin
     if a[i] >= 8.0 then Gioi:= Gioi + 1;
     if a[i] < 5 then Kem:= Kem + 1;
     if (a[i] < 8.0) and (a[i] >= 6.5) then Kha:= Kha + 1;
     if (a[i] >= 5) and (a[i] < 6.5) then Trungbinh:= Trungbinh + 1
    end;
    writeln('Ket qua hoc tap :');
    writeln(Gioi,' ban hoc gioi');
    writeln(Kha,' ban hoc kha');
    writeln(Trungbinh,' ban hoc trung binh');
    writeln(Kem,' ban hoc kem');
    readln
  End.
```

d) Gõ tiếp phần chương trình này vào máy tính sau phần khai báo. Dịch, chạy chương trình.

BÀI 2. Bổ sung và chỉnh sửa chương trình trong bài 1 để nhập hai loại điểm Toán và Ngữ văn của các bạn, sau đó in ra màn hình điểm trung bình của mỗi bạn trong lớp

(theo công thức điểm trung bình = (điểm Toán + điểm Ngữ văn)/2), điểm trung bình của cả lớp theo từng môn Toán và Ngữ văn.

a) Tìm hiểu ý nghĩa của các câu lệnh sau đây:

Phần khai báo:

```
var i, n: integer;
    TbToan, TbVan: real;
    DiemToan, DiemVan: array[1..100] of real;
```

Phần thân chương trình:

```
begin
    writeln('Diem trung binh :');
    for i:= 1 to n do
        writeln(i,'.', (DiemToan[i] + DiemVan[i])/2:3:1);
    TbToan:= 0; TbVan:= 0;
    for i:= 1 to n do
        begin
            TbToan:= TbToan + DiemToan[i];
            TbVan:= TbVan + DiemVan[i]
        end;
    TbToan:= TbToan/n; TbVan:= TbVan/n;
    writeln('Diem trung binh mon Toan : ',TbToan :3:2);
    writeln('Diem trung binh mon Van : ', TbVan :3:2);
end.
```

b) Bổ sung các câu lệnh trên vào vị trí thích hợp trong chương trình. Thêm các lệnh cần thiết, dịch và chạy chương trình với các số liệu thử.

TỔNG KẾT

1. Cú pháp khai báo biến mảng kiểu số nguyên và số thực trong Pascal có dạng:

   var *<tên biến mảng>*: **array** [*<chỉ số đầu>..<chỉ số cuối>*] **of** integer;

   var *<tên biến mảng>*: **array**[*<chỉ số đầu>..<chỉ số cuối>*] **of** real;

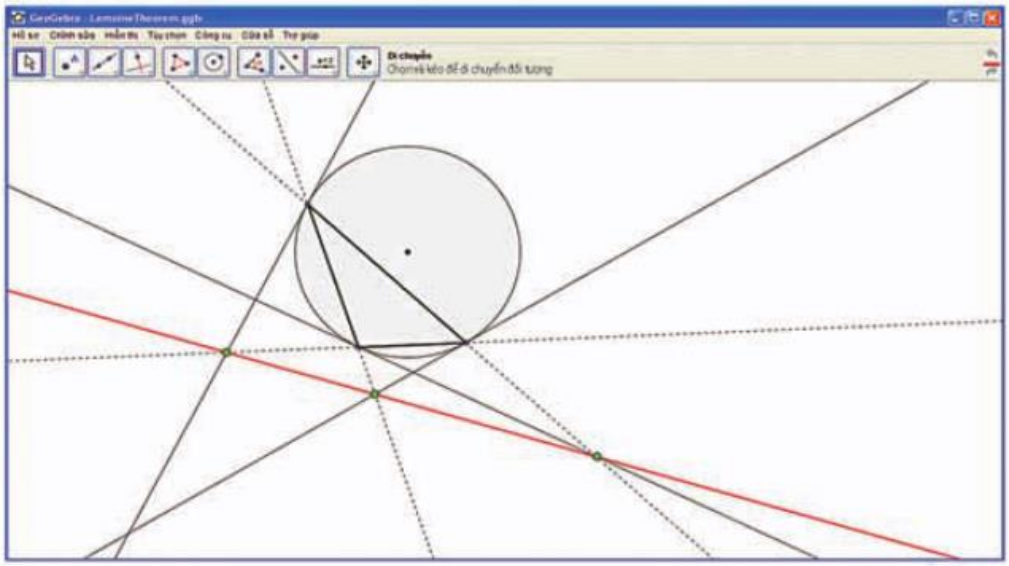
   trong đó *chỉ số đầu* không lớn hơn *chỉ số cuối*.

2. Tham chiếu tới phần tử của mảng được xác định bằng cách :

   <tên biến mảng>[*chỉ số*]

Phần 2

PHẦN MỀM HỌC TẬP



BÀI 10

LUYỆN GÕ PHÍM NHANH VỚI FINGER BREAKOUT

1. Giới thiệu phần mềm

Mục đích của phần mềm là luyện gõ bàn phím nhanh, chính xác.

2. Màn hình chính của phần mềm

a) Khởi động phần mềm

Để khởi động phần mềm em hãy nháy đúp chuột lên biểu tượng  .

b) Giới thiệu màn hình chính

Trong màn hình giới thiệu, nhấn phím Enter hoặc nháy nút OK để chuyển sang màn hình chính của phần mềm.

Màn hình chính của phần mềm ***Finger BreakOut***

Các thành phần trong màn hình chính của phần mềm gồm:

- Hình bàn phím ở vị trí trung tâm. Các phím được tô màu ứng với ngón tay gõ phím.

Màu của nhóm phím	Ngón tay gõ
 (xanh da trời nhạt)	Ngón út
 (vàng nhạt)	Ngón áp út (ngón đeo nhẫn)
 (cam nhạt)	Ngón giữa
 (xanh lá cây nhạt)	Ngón trỏ
 (tím nhạt)	Ngón cái

- Khung trống phía trên hình bàn phím là khu vực chơi.
- Khung bên phải chứa các lệnh và thông tin của lượt chơi. Ví dụ, tại ô Level có thể chọn các mức khó khác nhau của trò chơi: Bắt đầu (Beginner), Trung bình (Intermediate) và Nâng cao (Advanced).

c) Thoát khỏi phần mềm

Nếu muốn dừng chơi, hãy nháy chuột lên nút Stop ở khung bên phải.

Muốn thoát khỏi phần mềm, nháy nút  hoặc nhấn tổ hợp phím Alt+F4.

3. Hướng dẫn sử dụng

Để bắt đầu chơi, em hãy nháy chuột vào nút Start tại khung bên phải.

Trước mỗi lần chơi, hộp thoại giống như sau xuất hiện cho biết các phím (vùng bàn phím) sẽ được luyện gõ trong lần chơi đó.

Nhấn phím **Space** để bắt đầu chơi khi đã sẵn sàng.

Màn hình chính

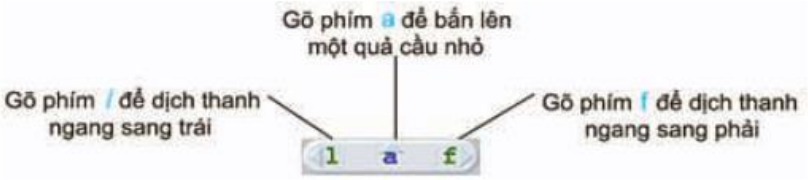
Khu vực chơi sẽ có các ô có dạng  làm thành khối. Nhiệm vụ của người chơi là làm các ô này biến mất khỏi màn hình bằng cách để các quả cầu va vào chúng.

Nếu tất cả các ô biến mất hết thì em đã thắng trong lượt chơi này.

Để di chuyển các quả cầu, em cần điều khiển một thanh ngang có ba kí tự:

Gõ các phím ứng với kí tự bên trái hoặc bên phải để di chuyển thanh ngang sang trái hoặc phải. Gõ kí tự ở giữa để bắn lên một quả cầu nhỏ  .

Lưu ý: Các chữ cái trong thanh ngang này sẽ thay đổi sau mỗi lần gõ phím.



Trên màn hình còn có thể có các quả cầu lớn  . Em cần chú ý đến các quả cầu lớn này. Không được để các quả cầu lớn này chạm "đất" bằng cách dịch chuyển thanh ngang sao cho chúng va vào thanh ngang rồi quay lên.

Nếu quả cầu lớn chạm đất, em sẽ mất một lượt chơi. Trong khi chơi, nếu được điểm cao người chơi sẽ được thưởng thêm các quả cầu lớn.

Như vậy, người chơi cần gõ phím thật nhanh, chính xác để điều khiển khéo léo các quả cầu.

Nếu có nhiều quả cầu lớn người chơi có thể thắng nhanh hơn.

Ở các mức khó hơn, sẽ còn có các con vật lạ  . Tuyệt đối không để các con vật này chạm vào thanh ngang. Nếu bị con vật chạm vào thanh ngang em sẽ mất một lượt chơi.

BÀI 11

HỌC VẼ HÌNH VỚI PHẦN MỀM GEOGEBRA

1. Em đã biết gì về GeoGebra?

Em đã được làm quen với phần mềm GeoGebra dùng để vẽ các hình hình học đơn giản như điểm, đoạn thẳng, đường thẳng. Đặc điểm quan trọng nhất của phần mềm GeoGebra là khả năng tạo ra sự gắn kết giữa các đối tượng hình học, được gọi là quan hệ như thuộc, vuông góc, song song. Đặc điểm này giúp cho phần mềm có thể vẽ được các hình rất chính xác và có khả năng tương tác như chuyển động nhưng vẫn giữ được mối quan hệ giữa các đối tượng.

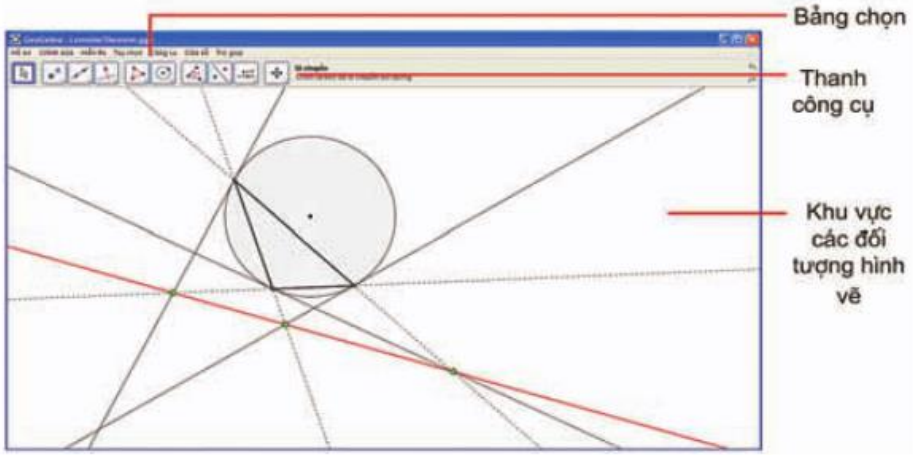
2. Làm quen với phần mềm GeoGebra

a) Khởi động

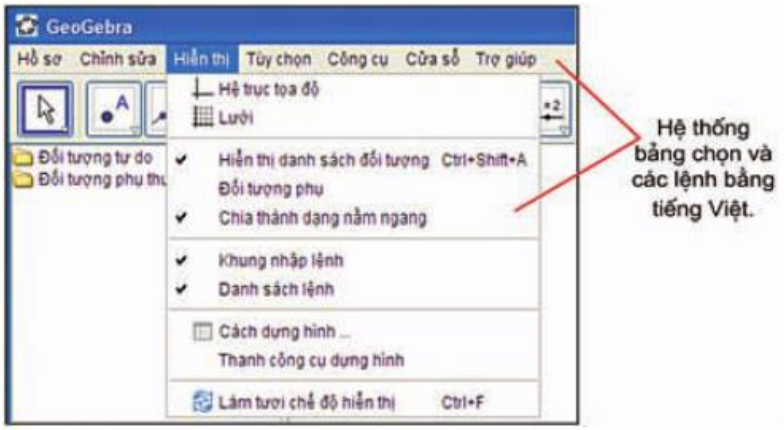
Nháy chuột tại biểu tượng  để khởi động chương trình.

b) Giới thiệu màn hình GeoGebra

Màn hình làm việc chính của phần mềm bao gồm bảng chọn, thanh công cụ và khu vực thể hiện các đối tượng.



• *Bảng chọn* là hệ thống các lệnh chính của phần mềm GeoGebra.



• *Thanh công cụ* của phần mềm chứa các công cụ làm việc chính. Đây chính là các công cụ dùng để vẽ, điều chỉnh và làm việc với các đối tượng.

Khi nháy chuột lên nút nhỏ hình thành tam giác phía dưới nút lệnh ta sẽ thấy xuất hiện các công cụ khác cùng nhóm.

Mỗi công cụ đều có một biểu tượng riêng tương ứng. Biểu tượng cho biết công dụng của công cụ đó.

c) Giới thiệu các công cụ làm việc chính

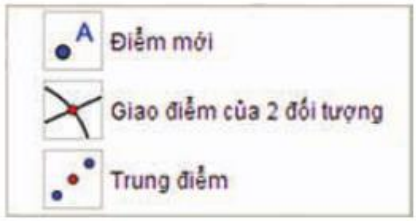
Để chọn một công cụ hãy nháy chuột lên biểu tượng của công cụ này.

• *Công cụ di chuyển*  có ý nghĩa đặc biệt là dùng để di chuyển hình. Với công cụ này, kéo thả đối tượng (điểm, đoạn, đường,...) để di chuyển nó. Công cụ này cũng dùng để chọn các đối tượng khi thực hiện các lệnh điều khiển thuộc tính của chúng.

Có thể chọn nhiều đối tượng bằng cách nhấn giữ phím Ctrl trong khi chọn hoặc kéo thả chuột để tạo một khung hình chữ nhật trên màn hình bao quanh các đối tượng muốn chọn.

Lưu ý: Khi đang sử dụng một công cụ khác, nhấn phím ESC để chuyển về công cụ di chuyển.

- *Các công cụ liên quan đến đối tượng điểm*



– Công cụ  dùng để tạo một điểm mới. Điểm được tạo có thể là điểm tự do trên mặt phẳng hoặc là điểm thuộc một đối tượng khác (ví dụ đường thẳng, đoạn thẳng).

Thao tác: Chọn công cụ và nháy chuột lên một điểm trống trên màn hình hoặc nháy chuột lên một đối tượng để tạo điểm thuộc đối tượng này.

– Công cụ  dùng để tạo ra điểm là giao của hai đối tượng đã có trên mặt phẳng.

Cách tạo: Chọn công cụ và lần lượt nháy chuột chọn hai đối tượng đã có trên mặt phẳng.

– Công cụ  dùng để tạo trung điểm của (đoạn thẳng nối) hai điểm cho trước: chọn công cụ rồi nháy chuột tại hai điểm này để tạo trung điểm.

- *Các công cụ liên quan đến đoạn, đường thẳng*

– Các công cụ  ,  ,  theo thứ tự dùng để tạo đường, đoạn, tia đi qua hai điểm cho trước.

Thao tác: Chọn công cụ, sau đó nháy chuột chọn lần lượt hai điểm trên màn hình.

– Công cụ  sẽ tạo ra một đoạn thẳng đi qua một điểm cho trước với độ dài có thể nhập trực tiếp từ bàn phím.

Thao tác: Chọn công cụ, chọn một điểm cho trước, sau đó nhập một giá trị số vào cửa sổ có dạng:

Nháy nút **Áp dụng** sau khi đã nhập xong độ dài đoạn thẳng.

Lưu ý: Trong cửa sổ trên có thể nhập một chuỗi kí tự là tên của một giá trị số được định nghĩa từ trước.

- *Các công cụ tạo mối quan hệ hình học*

– Công cụ  dùng để tạo đường thẳng đi qua một điểm và vuông góc với một đường hoặc đoạn thẳng cho trước.

Thao tác: Chọn công cụ, sau đó lần lượt chọn điểm, đường (đoạn, tia) hoặc ngược lại chọn đường (đoạn, tia) rồi chọn điểm.

– Công cụ  sẽ tạo ra một đường thẳng song song với một đường (đoạn) cho trước và đi qua một điểm cho trước.

Thao tác: Chọn công cụ, sau đó lần lượt chọn điểm, đường (đoạn, tia) hoặc ngược lại chọn đường (đoạn, tia) rồi chọn điểm.

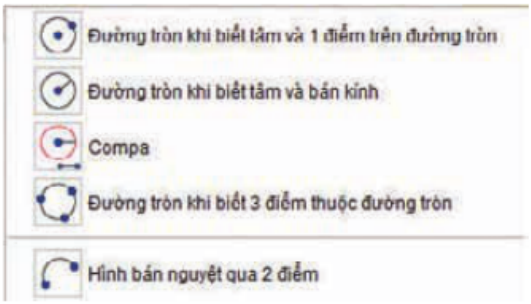
– Công cụ  dùng để vẽ đường trung trực của một đoạn thẳng hoặc hai điểm cho trước.

Thao tác: Chọn công cụ, sau đó chọn một đoạn thẳng hoặc chọn hai điểm cho trước trên mặt phẳng.

– Công cụ  dùng để tạo đường phân giác của một góc cho trước. Góc này xác định bởi ba điểm trên mặt phẳng.

Thao tác: Chọn công cụ, sau đó lần lượt chọn ba điểm trên mặt phẳng. Điểm chọn thứ hai chính là đỉnh của góc này.

- *Các công cụ liên quan đến đường tròn*



– Công cụ  tạo ra đường tròn bằng cách xác định tâm và một điểm trên đường tròn.

Thao tác: Chọn công cụ, chọn tâm đường tròn và điểm thứ hai nằm trên đường tròn.

– Công cụ  dùng để tạo ra đường tròn bằng cách xác định tâm và bán kính.

Thao tác: Chọn công cụ, chọn tâm đường tròn, sau đó nhập giá trị bán kính trong hộp thoại sau:

– Công cụ  dùng để tạo đường tròn khi biết tâm và độ dài bán kính theo một đoạn thẳng cho trước.

Thao tác: Chọn công cụ, chọn một đoạn thẳng làm bán kính, sau đó chọn một điểm làm tâm đường tròn.

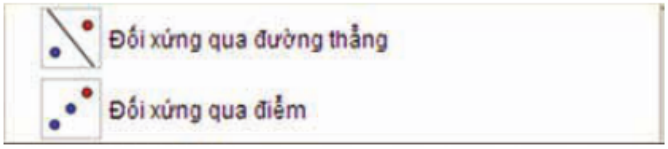
– Công cụ  dùng để vẽ đường tròn đi qua ba điểm cho trước.

Thao tác: Chọn công cụ, sau đó lần lượt chọn ba điểm.

– Công cụ  dùng để tạo hình bán nguyệt đi qua hai điểm đối xứng tâm.

Thao tác: Chọn công cụ, chọn lần lượt hai điểm. Nửa đường tròn được tạo sẽ là phần đường tròn theo chiều ngược kim đồng hồ từ điểm thứ nhất đến điểm thứ hai.

- *Các công cụ biến đổi hình học*



– Công cụ  dùng để tạo ra một đối tượng đối xứng với một đối tượng cho trước qua một trục là đường hoặc đoạn thẳng.

Thao tác: Chọn công cụ, chọn đối tượng cần biến đổi (có thể chọn nhiều đối tượng bằng cách kéo thả chuột tạo thành một khung chữ nhật chứa các đối tượng muốn chọn), sau đó nháy chuột lên đường hoặc đoạn thẳng làm trục đối xứng.

– Công cụ  dùng để tạo ra một đối tượng đối xứng với một đối tượng cho trước qua một điểm cho trước (điểm này gọi là tâm đối xứng).

Thao tác: Chọn công cụ, chọn đối tượng cần biến đổi (có thể chọn nhiều đối tượng bằng cách kéo thả chuột tạo thành một khung chữ nhật chứa các đối tượng muốn chọn), sau đó nháy chuột lên điểm là tâm đối xứng.

d) Các thao tác với tệp

Mỗi trang hình vẽ sẽ được lưu lại trong một tệp có phần mở rộng là ggb. Để lưu hình hãy nhấn tổ hợp phím Ctrl+S hoặc thực hiện lệnh Hồ sơ → Lưu lại từ bảng chọn. Nếu là lần đầu tiên lưu tệp, phần mềm sẽ yêu cầu nhập tên tệp. Gõ tên tệp tại vị trí File name và nháy chuột vào nút Save.

Để mở một tệp đã có, nhấn tổ hợp phím Ctrl+O hoặc thực hiện lệnh Hồ sơ → Mở. Chọn tệp cần mở hoặc gõ tên tệp tại ô File name, sau đó nháy chuột vào nút Open.

e) Thoát khỏi phần mềm

Nháy chuột chọn Hồ sơ → Đóng hoặc nhấn tổ hợp phím Alt+F4.

3. Đối tượng hình học

a) Khái niệm đối tượng hình học

Một hình hình học sẽ bao gồm nhiều đối tượng cơ bản. Các đối tượng hình học cơ bản bao gồm: điểm, đoạn thẳng, đường thẳng, tia, hình tròn, cung tròn.

b) Đối tượng tự do và đối tượng phụ thuộc

Em đã được làm quen với khái niệm *quan hệ* giữa các đối tượng.

Sau đây là một vài ví dụ:

- *Điểm thuộc đường thẳng*

Cho trước một đường thẳng, sau đó xác định một điểm "thuộc" đường thẳng này. Chúng ta có quan hệ "thuộc". Trong trường hợp này đối tượng điểm có quan hệ thuộc đối tượng đường thẳng.

- *Đường thẳng đi qua hai điểm*

Cho trước hai điểm. Vẽ một đường thẳng đi qua hai điểm này. Chúng ta có quan hệ "đi qua". Trong trường hợp này đường thẳng có quan hệ phụ thuộc vào hai điểm cho trước.

- *Giao của hai đối tượng hình học*

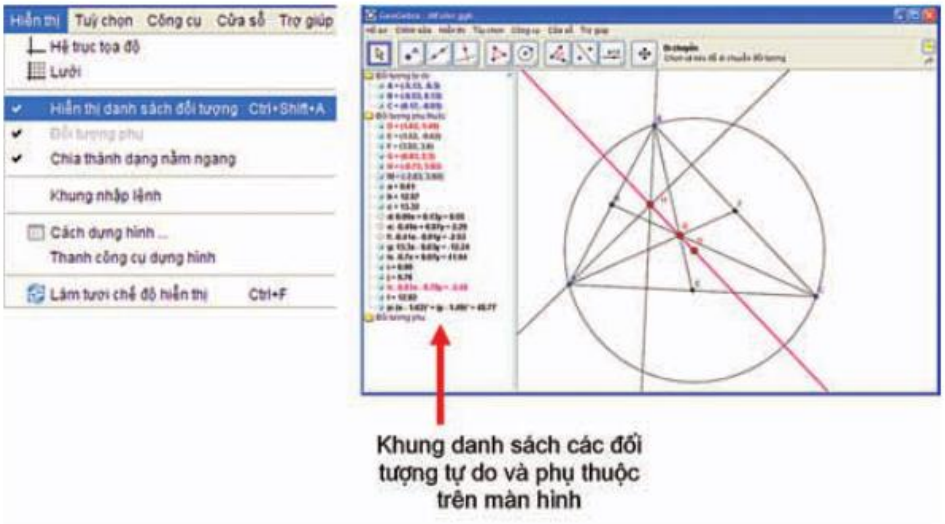
Cho trước một hình tròn và một đường thẳng. Dùng công cụ  để xác định giao của đường thẳng và đường tròn. Chúng ta sẽ có quan hệ "giao nhau". Giao điểm, nếu có, thuộc hai đối tượng ban đầu là đường tròn và đường thẳng.

Một đối tượng không phụ thuộc vào bất kì một đối tượng nào khác được gọi là đối tượng tự do. Các đối tượng còn lại gọi là đối tượng phụ thuộc. Như vậy mọi đối tượng hình học trong phần mềm GeoGebra đều có thể chia thành hai loại là tự do hay phụ thuộc.

c) Danh sách các đối tượng trên màn hình

Phần mềm GeoGebra cho phép hiển thị danh sách tất cả các đối tượng hình học hiện đang có trên trang hình.

Dùng lệnh **Hiển thị → Hiển thị danh sách đối tượng** để hiện/ẩn khung thông tin này trên màn hình.



d) Thay đổi thuộc tính của đối tượng

Các đối tượng hình học đều có các tính chất như tên (nhãn) đối tượng, cách thể hiện kiểu đường, màu sắc, ...

Sau đây là một vài thao tác thường dùng để thay đổi tính chất của đối tượng.

- *Ẩn đối tượng:* Để ẩn một đối tượng, thực hiện các thao tác sau:

1. Nháy nút phải chuột lên đối tượng;
2. Huỷ chọn **Hiển thị đối tượng** trong bảng chọn hiện ra:

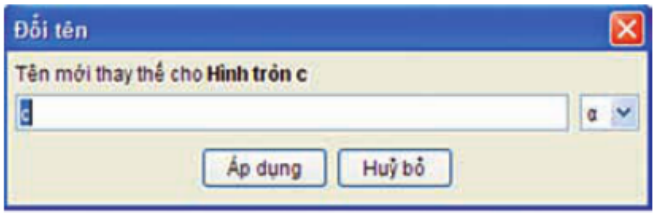
- *Ẩn/hiện tên (nhãn) của đối tượng:* Để làm ẩn hay hiện tên của đối tượng, thực hiện các thao tác sau:

1. Nháy nút phải chuột lên đối tượng trên màn hình;
2. Huỷ chọn **Hiển thị tên** trong bảng chọn hiện ra.

- *Thay đổi tên của đối tượng:* Muốn thay đổi tên của một đối tượng, thực hiện các thao tác sau:

1. Nháy nút phải chuột lên đối tượng trên màn hình;
2. Chọn lệnh **Đổi tên** trong bảng chọn hiện ra.

Sau đó nhập tên mới trong hộp thoại:



3. Nháy nút **Áp dụng** để thay đổi, nháy nút **Huỷ bỏ** nếu không muốn đổi tên.

• *Đặt/huỷ vết chuyển động của đối tượng:* Chức năng đặt vết khi đối tượng chuyển động có ý nghĩa đặc biệt trong các phần mềm "Toán học động". Chức năng này được sử dụng trong các bài toán dự đoán quỹ tích và khảo sát một tính chất nào đó của hình khi các đối tượng khác chuyển động.

Để đặt/huỷ vết chuyển động cho một đối tượng trên màn hình thực hiện thao tác sau:

1. Nháy nút phải chuột lên đối tượng;
2. Chọn **Mở dấu vết khi di chuyển**.

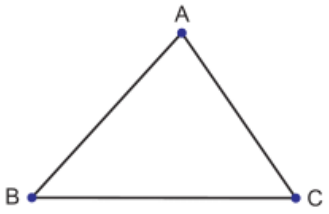
Để xoá các vết được vẽ, nhấn tổ hợp phím **Ctrl+F**.

• *Xoá đối tượng:* Muốn xoá hẳn đối tượng, ta có thể thực hiện một trong các thao tác sau:

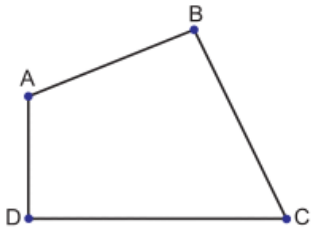
1. Dùng công cụ  chọn đối tượng rồi nhấn phím **Delete**.
2. Nháy nút phải chuột lên đối tượng và thực hiện lệnh.
3. Chọn công cụ  trên thanh công cụ và nháy chuột lên đối tượng muốn xoá.

4. Bài tập thực hành

1. Vẽ tam giác, tứ giác.

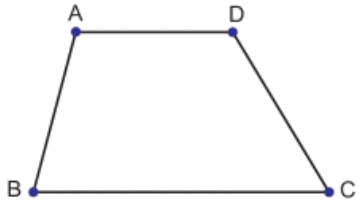


Dùng công cụ đoạn thẳng vẽ các cạnh của tam giác.



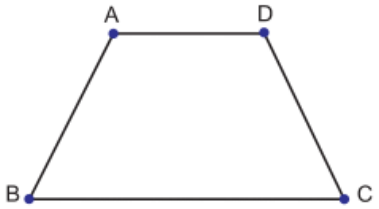
Dùng công cụ đoạn thẳng vẽ các cạnh của tứ giác.

2. Vẽ hình thang.



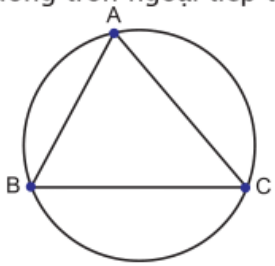
Cho trước ba đỉnh A, B, C. Dựng đỉnh D của hình thang ABCD dựa trên các công cụ đoạn thẳng và đường song song.

3. Vẽ hình thang cân.



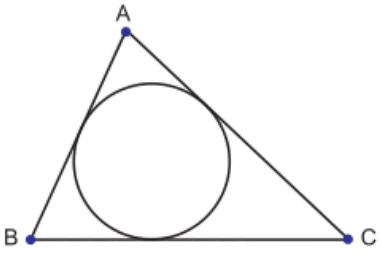
Cho trước ba đỉnh A, B, C. Dựng đỉnh D của hình thang cân ABCD dựa trên các công cụ đoạn thẳng, đường trung trực và phép biến đổi đối xứng qua trục.

4. Vẽ đường tròn ngoại tiếp tam giác



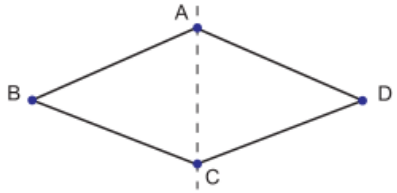
Cho trước tam giác ABC. Dùng công cụ đường tròn vẽ đường tròn đi qua ba điểm A, B, C.

5. Vẽ đường tròn nội tiếp tam giác.



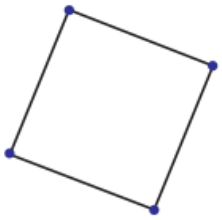
Cho trước tam giác ABC. Dùng các công cụ đường phân giác, đường vuông góc và đường tròn vẽ đường tròn nội tiếp tam giác ABC.

6. Vẽ hình thoi.



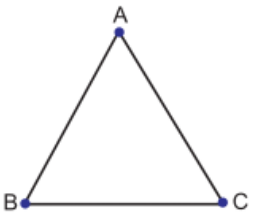
Cho trước cạnh AB và một đường thẳng đi qua A. Hãy vẽ hình thoi ABCD lấy đường thẳng đã cho là đường chéo. Sử dụng các công cụ thích hợp đã học để dựng các đỉnh C, D của hình thoi.

7. Vẽ hình vuông.



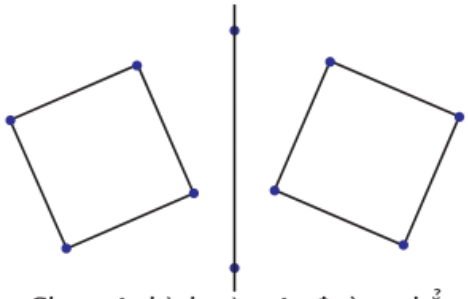
Sử dụng các công cụ thích hợp để vẽ một hình vuông nếu biết trước một cạnh.

8. Vẽ tam giác đều.



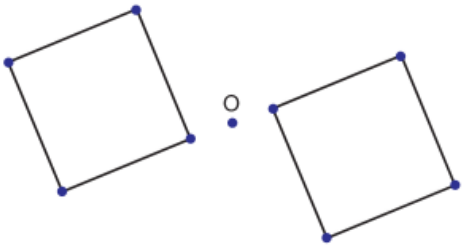
Cho trước cạnh BC, hãy vẽ tam giác đều ABC.

9. Vẽ một hình là đối xứng trục của một đối tượng cho trước trên màn hình.



Cho một hình và một đường thẳng trên mặt phẳng. Hãy dựng hình mới là đối xứng của hình đã cho qua trục là đường thẳng trên. Sử dụng công cụ đối xứng trục để vẽ hình.

10. Vẽ một hình là đối xứng qua tâm của một đối tượng cho trước trên màn hình.



Cho trước một hình và một điểm O. Hãy dựng hình mới là đối xứng qua tâm O của hình đã cho. Sử dụng công cụ đối xứng tâm để vẽ hình.

BÀI 12

QUAN SÁT HÌNH KHÔNG GIAN VỚI PHẦN MỀM YENKA

1. Giới thiệu phần mềm Yenka

Yenka là một phần mềm nhỏ, đơn giản nhưng rất hữu ích khi mới làm quen với các hình không gian như hình chóp, hình nón, hình trụ. Ngoài việc tạo ra các hình này, em còn có thể thay đổi kích thước, màu, di chuyển và sắp xếp chúng. Từ những hình không gian cơ bản em còn có thể sáng tạo ra các mô hình hoàn chỉnh như công trình xây dựng, kiến trúc theo ý mình.

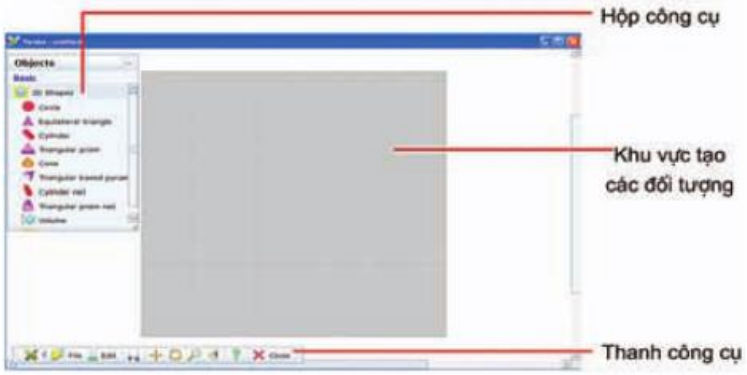
2. Giới thiệu màn hình làm việc chính của phần mềm

a) Khởi động phần mềm

Nháy đúp vào biểu tượng  để khởi động phần mềm, khi đó sẽ xuất hiện hộp thoại sau đây:

Nháy nút Try Basic Version để vào màn hình chính của phần mềm.

b) Màn hình chính



• *Hộp công cụ* dùng để tạo ra các hình không gian. Các hình sẽ được tạo ra tại khung chính giữa màn hình.

• *Thanh công cụ* chứa các nút lệnh dùng để điều khiển và làm việc với các đối tượng.

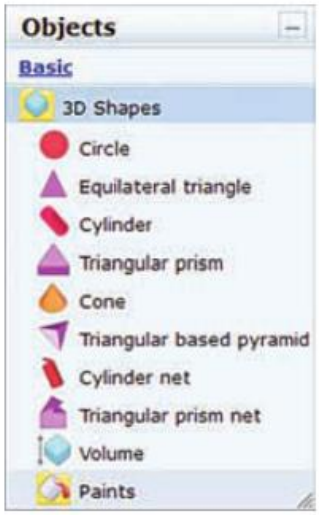
c) Thoát khỏi phần mềm

Muốn thoát khỏi phần mềm, nháy nút Close trên thanh công cụ.

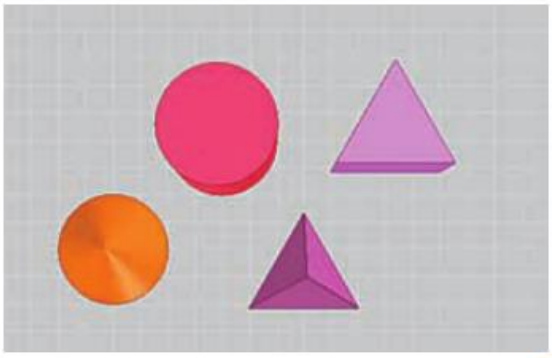
3. Tạo hình không gian

a) Tạo mô hình

Để thiết lập đối tượng hình, em phải làm việc với hộp công cụ:



Các công cụ dùng để tạo hình không gian thường gặp gồm: hình trụ ( ), hình nón ( ), hình chóp ( ) và hình lăng trụ ( ). Khi kéo thả các đối tượng này vào giữa màn hình, em sẽ nhận được mô hình có dạng sau:

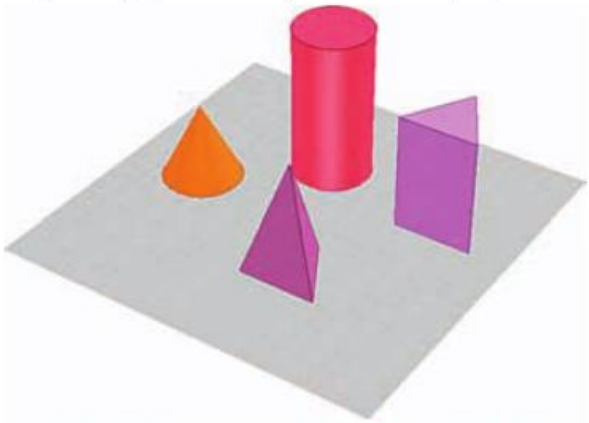


Mô hình gồm các hình không gian nhìn từ trên xuống.

• *Xoay mô hình trong không gian 3D*

1. Nháy vào biểu tượng  trên thanh công cụ. Khi đó con trỏ sẽ trở thành dạng  .

2. Đưa con trỏ chuột lên mô hình, nhấn giữ và di chuyển chuột, em sẽ thấy mô hình quay trong không gian 3D. Lệnh hết tác dụng khi em thả chuột.



Một cách nhìn khác vào mô hình của các hình không gian

• *Phóng to, thu nhỏ*

1. Nháy chuột vào biểu tượng  trên thanh công cụ. Khi đó con trỏ sẽ trở thành dạng 

2. Nhấn giữ và di chuyển chuột em sẽ thấy mô hình được phóng to, thu nhỏ tuỳ thuộc vào sự di chuyển của chuột. Lệnh hết tác dụng khi em thả chuột.

• *Dịch chuyển khung mô hình*

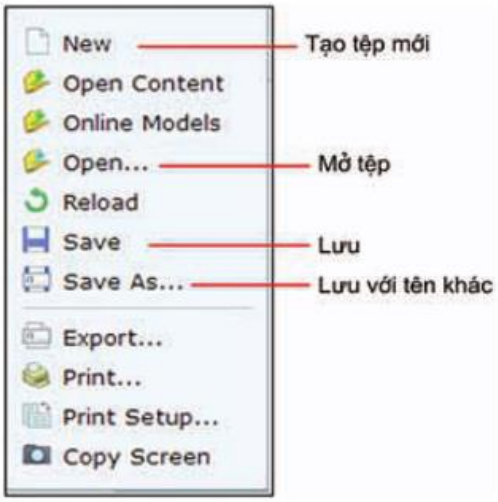
1. Nháy chuột vào biểu tượng  trên thanh công cụ. Khi đó con trỏ sẽ trở thành dạng  .

2. Nhấn giữ và di chuyển chuột em sẽ thấy mô hình chuyển động theo hướng di chuyển của chuột. Lệnh hết tác dụng khi em thả chuột.

b) Các lệnh tạo mới, lưu, mở tệp mô hình

Các tệp lưu mô hình có phần mở rộng ngầm định là yka. Các thao tác với tệp đều thông qua biểu tượng . Khi nháy chuột vào biểu tượng này một

bảng chọn xuất hiện có dạng như hình bên:



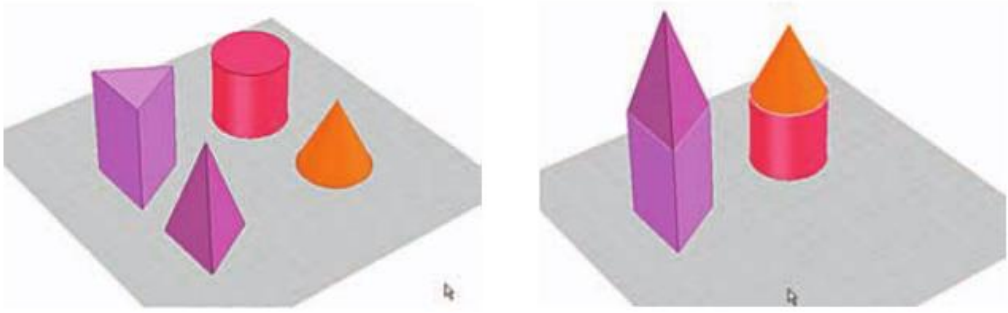
c) Xoá các đối tượng

Để xoá một hình, em hãy nháy chuột lên hình đó để chọn (đánh dấu) rồi nhấn phím Delete. Có thể chọn đồng thời nhiều đối tượng bằng cách nhấn giữ phím Ctrl trong khi chọn hoặc nhấn tổ hợp phím Ctrl+A để chọn tất cả các đối tượng hiện có trên màn hình.

4. Khám phá, điều khiển các hình không gian

a) Thay đổi, di chuyển

Muốn di chuyển một hình không gian, hãy kéo thả đối tượng đó. Chú ý khi di chuyển một hình lên đúng đỉnh của một hình khác ta sẽ được hai hình không gian chồng nhau. Với cách này, ta có thể tạo ra những hình với nhiều kiểu kiến trúc khác nhau.

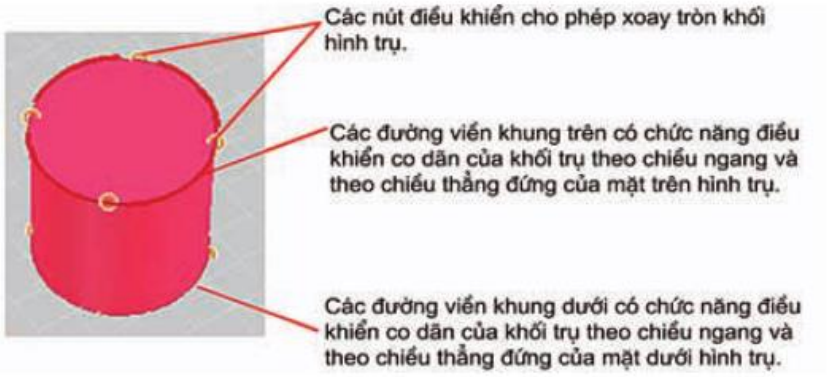


Có thể di chuyển hay xếp chồng các hình lên nhau.
Hình bên phải là kết quả của các khối đã xếp chồng lên nhau.

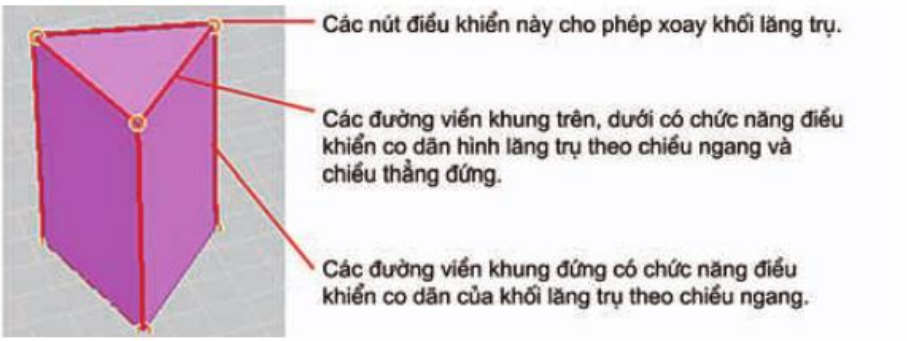
b) Thay đổi kích thước

Để thay đổi kích thước của một đối tượng trước tiên cần chọn hình. Khi đó sẽ xuất hiện các đường viền và các nút nhỏ trên đối tượng, cho phép tương tác để thay đổi kích thước. Tuỳ vào từng đối tượng mà các nút, đường viền có dạng khác nhau.

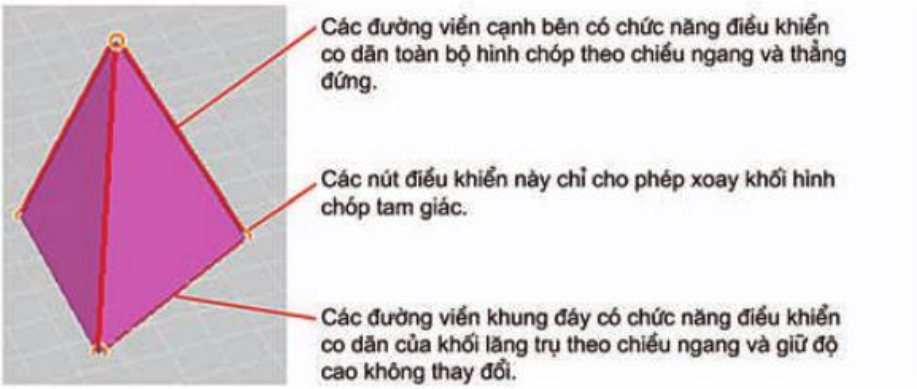
- *Hình trụ*



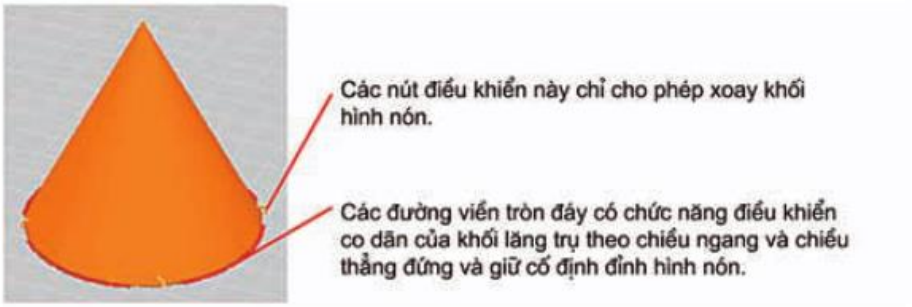
- *Hình lăng trụ tam giác*



- *Hình chóp tam giác*



- *Hình nón*

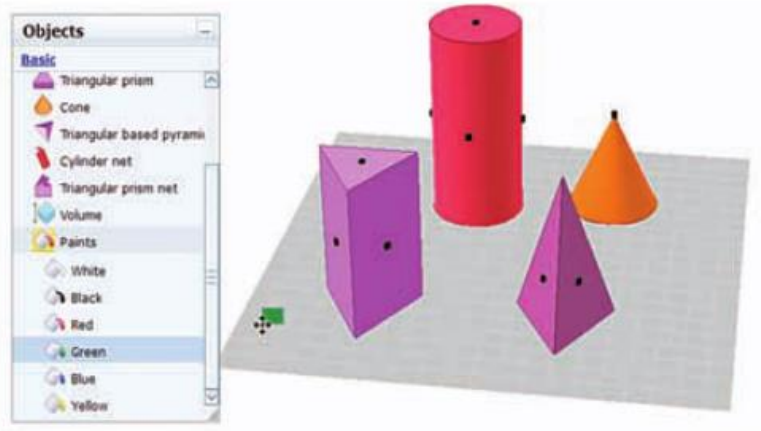


c) Thay đổi màu cho các hình

Muốn tô màu, thay đổi màu cho các hình, em dùng công cụ Paints. Khi nháy chuột vào công cụ này em sẽ thấy một danh sách các màu như hình bên.

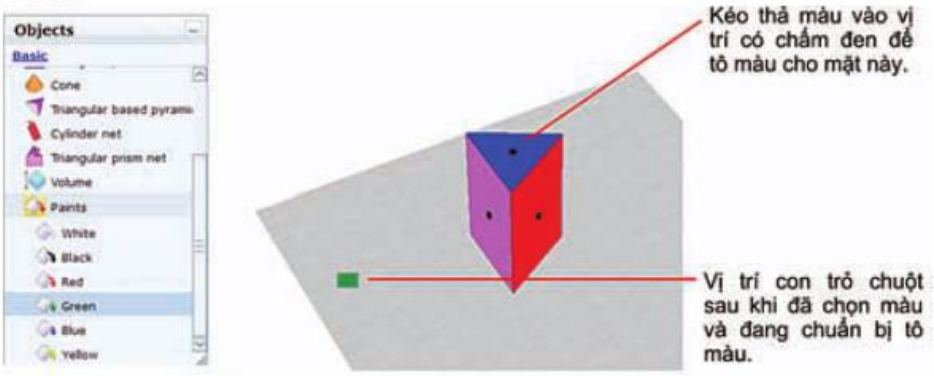
Các bước thực hiện tô màu:

Kéo thả một màu ra mô hình. Khi đó trên các hình xuất hiện các chấm đen cho biết hình đó có thể thay đổi màu. Kéo thả màu vào các chấm đen để tô màu.



Kéo thả màu cần tô vào các chấm đen để thay đổi màu.

Ví dụ, ta có thể tô màu các mặt của hình lăng trụ tam giác với các màu khác nhau.



d) Thay đổi tính chất của hình

Các tính chất của hình có thể được thay đổi thông qua hộp thoại tính chất đối tượng. Nháy đúp lên đối tượng, hộp thoại mô tả các thông tin, tính chất của đối tượng được mở ra.

Hình dưới đây là hộp thoại tính chất của hình lăng trụ đều. Chúng ta có thể thay đổi hai tham số quan trọng của hình là chiều cao (height) và độ dài cạnh đáy (base edge) bằng cách gõ trực tiếp vào ô hoặc nháy chuột vào các nút  để tăng, giảm từng đơn vị.

Hộp thoại tính chất của hình lăng trụ đều
(Có một vạch nối nhỏ từ đối tượng hình đến hộp thoại này).

e) Gấp giấy thành hình không gian

Một chức năng rất hay của phần mềm là cho phép ta quan sát cách tạo hình không gian từ một hình phẳng.

Phần mềm sẽ cho phép quan sát và thực hiện hai quá trình ngược nhau:

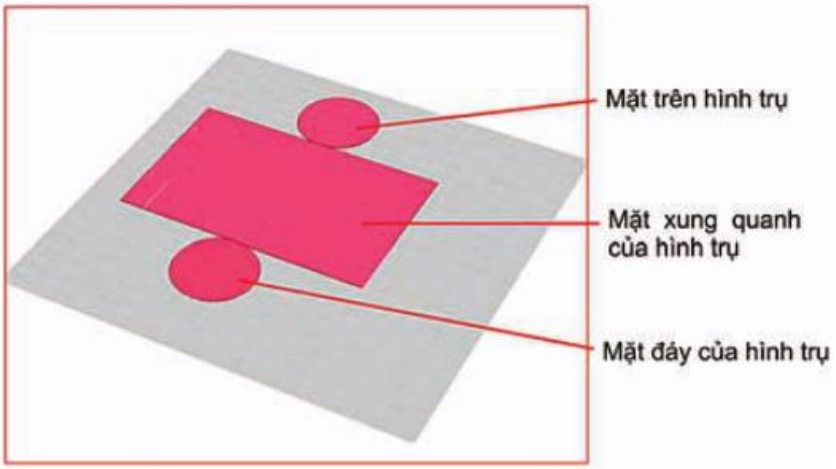
- Cho hình phẳng, cần "gấp" lại để tạo thành một hình không gian.
- Cho trước hình không gian, cần "mở" ra thành một hình phẳng.

• *Gấp hình phẳng để tạo hình không gian*

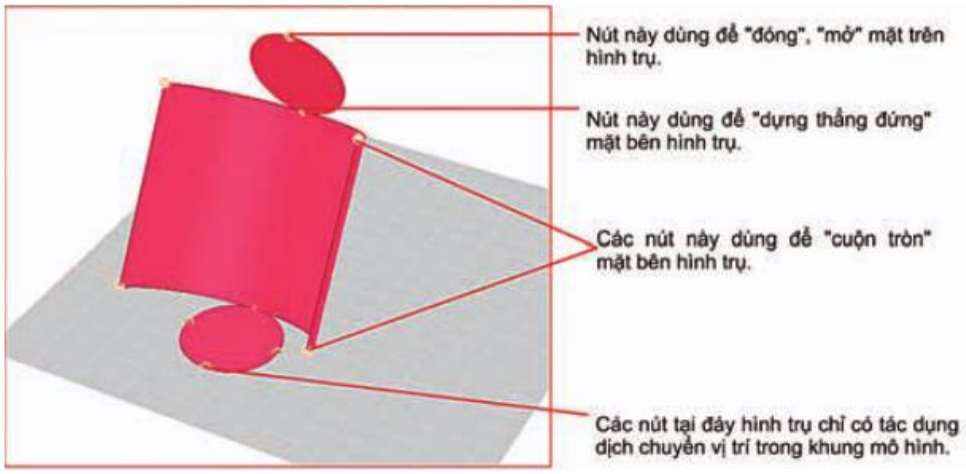
Sử dụng các công cụ đối tượng để tạo các hình phẳng trong khung mô hình bằng cách kéo thả các đối tượng này. Hiện tại phần mềm hỗ trợ cho hai hình, hình trụ Cylinder net và hình lăng trụ Triangular prism net .

1. Chọn Cylinder net hoặc Triangular prism net trong hộp công cụ. Kéo thả đối tượng này vào giữa màn hình.

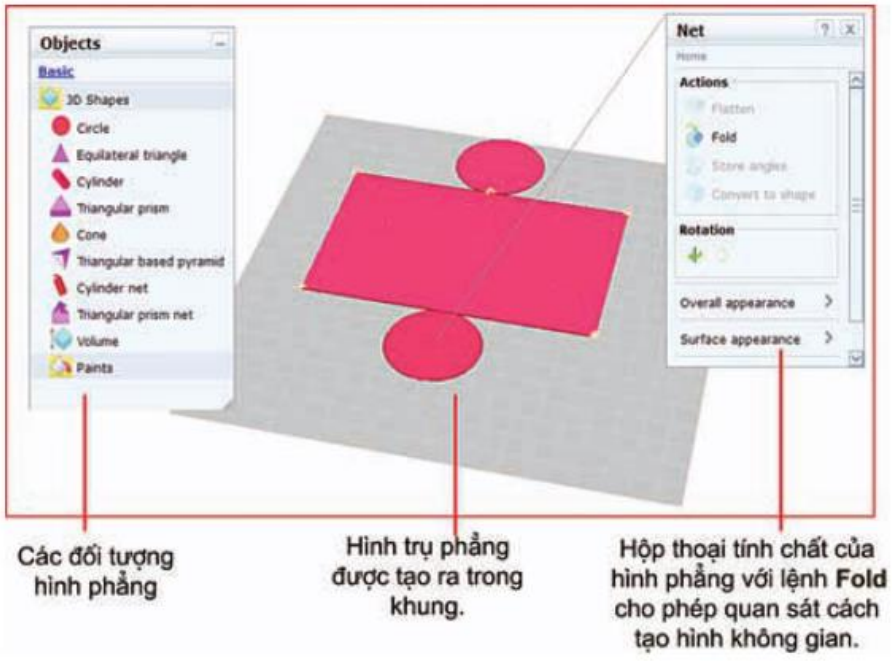
Ví dụ, hình trụ phẳng dưới đây:



2. Kéo thả chuột để thực hiện thao tác "gấp" hình phẳng này thành hình không gian tương ứng.

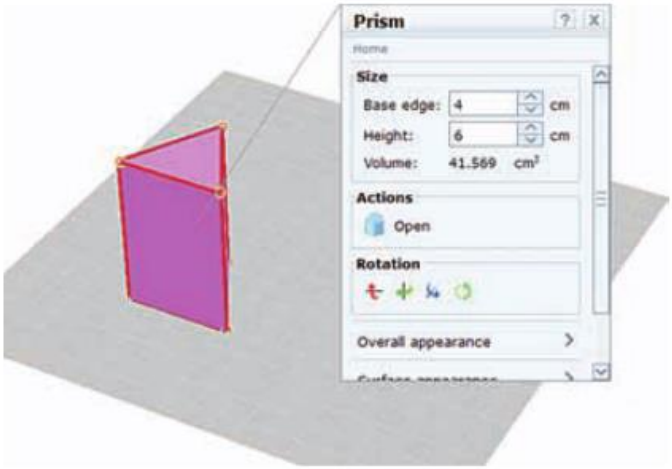


Có thể xem quá trình "gấp" một cách tự động như sau: nháy đúp chuột lên đối tượng để mở hộp thoại tính chất, sau đó chọn lệnh Fold trong hộp thoại này.



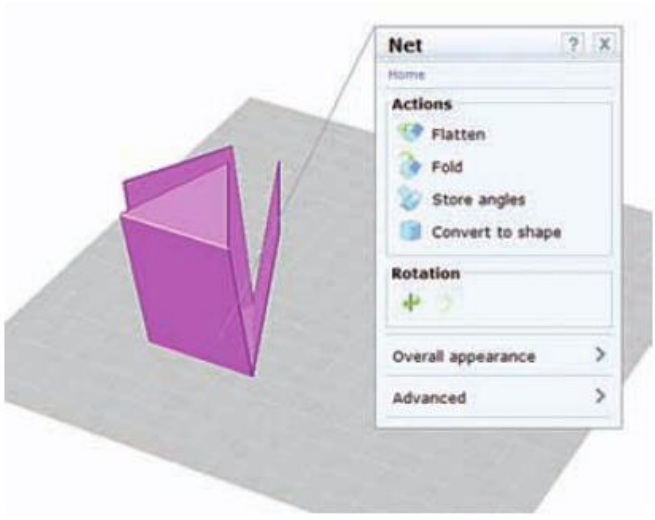
- *Mở hình không gian thành hình phẳng*

Ngược lại, đối với các hình không gian (hình trụ, lăng trụ, chóp), dùng lệnh **Open** trong hộp thoại tính chất để biến đổi hình không gian 3D thành "hình phẳng".



Hộp thoại tính chất hình lăng trụ tam giác đều.

Nháy nút **Open** để chuyển hình này sang dạng phẳng (Net):



Hộp thoại tính chất hình lăng trụ đã chuyển sang dạng phẳng.

Đối với hình phẳng, các lệnh sau đây có thể thực hiện:

- *Flatten:* Tự động làm phẳng hình này trong mô hình.
- *Fold:* Tự động gấp lại về trạng thái ban đầu hoặc một trạng thái thích hợp bởi lệnh Store angles.
- *Store angles:* Cố định vị trí của lệnh gấp lại. Lệnh này chỉ có tác dụng khi đang thực hiện lệnh Fold.
- *Convert to Shape:* Chuyển trạng thái hình phẳng thành hình 3D. Lệnh này chỉ có tác dụng khi đã thực hiện xong việc gấp hoàn toàn hình phẳng bởi lệnh Fold.

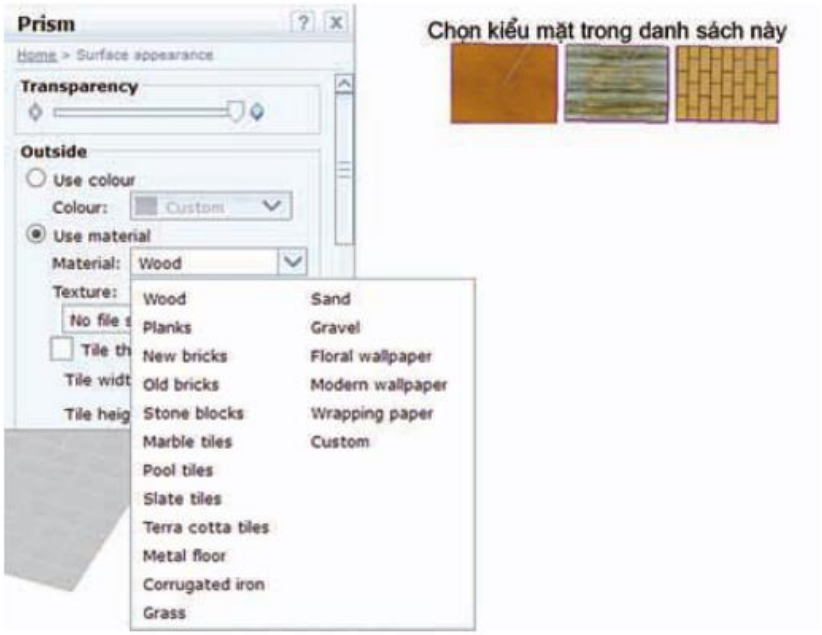
5. Một số chức năng nâng cao

a) Thay đổi mẫu thể hiện hình

Đối với các mặt của các hình không gian, ta không những có thể thay đổi màu, mà còn thay đổi được kiểu và mẫu thể hiện. Ví dụ, ta có thể "lát" mặt xung quanh của hình trụ bằng mẫu hình viên gạch, ...

Thao tác thực hiện:

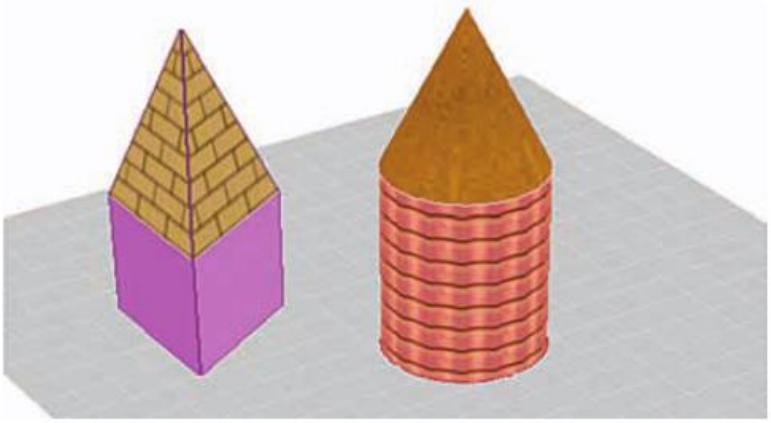
1. Nháy đúp chuột để mở hộp thoại tính chất của hình.



2. Chọn lệnh thay đổi kiểu bề mặt Surface appearance >.

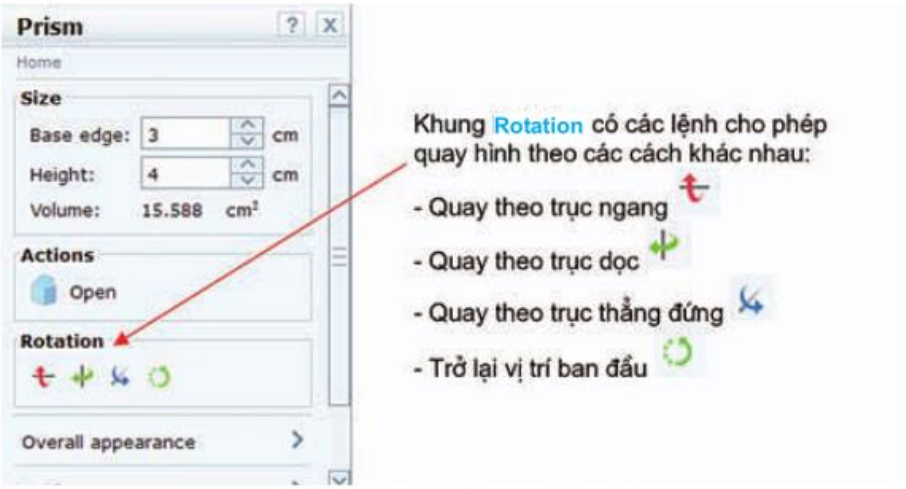
3. Trong hộp thoại tiếp theo, chọn **Use material** và chọn mẫu trong danh sách **Material** phía dưới.

Sau khi chọn mẫu như trên, kết quả có thể như sau:



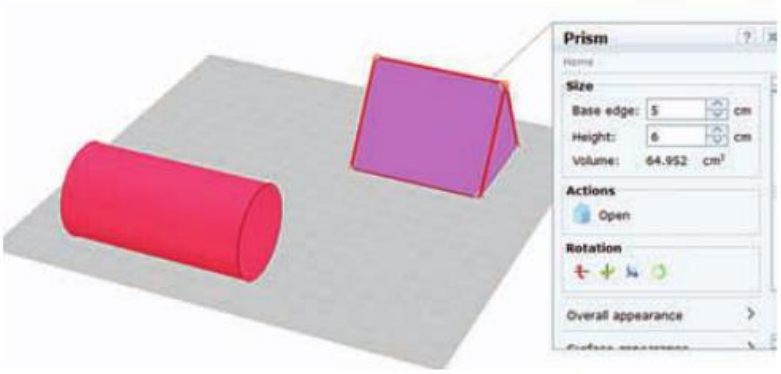
b) Quay hình trong không gian

Trong hộp thoại tính chất của hình, em có thể quay hình theo các cách khác nhau trong không gian.



*Các nút lệnh ở khung **Rotation***

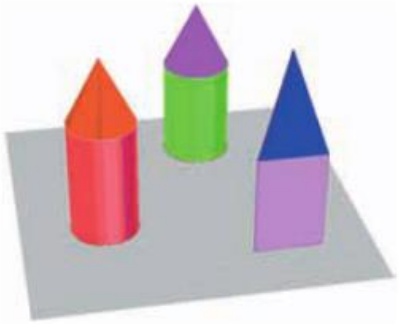
Trong hình dưới đây, hình trụ và hình lăng trụ đã được xoay quanh trục dọc để thành các hình nằm ngang trên mô hình.



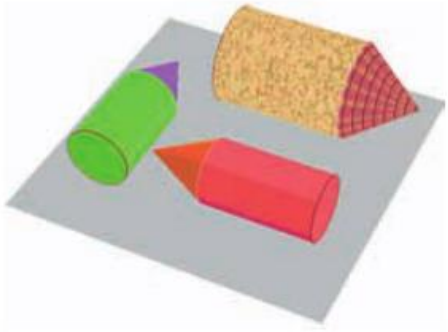
Kết hợp các chức năng và công cụ nâng cao, chúng ta có thể tạo ra được các hình không gian đa dạng, với màu và kiểu thể hiện phong phú.

6. Bài tập thực hành

1. Tạo các hình sau:

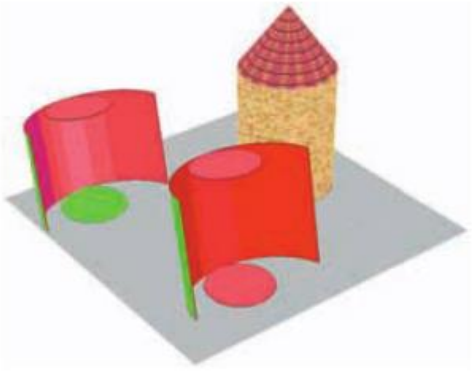


a)



b)

2. Từ hình của bài tập trên, hãy biến đổi để trở thành hình sau:



MỤC LỤC

Phần 1. Lập trình đơn giản

Phần 2. Phần mềm học tập

Chịu trách nhiệm xuất bản:
Chủ tịch Hội đồng Thành viên kiêm Tổng Giám đốc NGƯT NGÔ TRẦN ÁI
Phó Tổng Giám đốc kiêm Tổng biên tập GS.TS VŨ VĂN HÙNG

Biên tập lần đầu:
NGUYỄN THỊ THANH XUÂN - NGUYỄN THỊ NGUYÊN THUÝ

Biên tập tái bản:
DƯƠNG VŨ KHÁNH THUẬN - NGUYỄN THỊ NGUYÊN THUÝ

Thiết kế sách và trình bày bìa:
TẠ THANH TÙNG - LƯƠNG QUỐC HIỆP

Chế bản:
CÔNG TY CỔ PHẦN MĨ THUẬT & TRUYỀN THÔNG

Sửa bản in:
DƯƠNG VŨ KHÁNH THUẬN

TIN HỌC DÀNH CHO TRUNG HỌC CƠ SỞ - QUYỂN 3

Mã số: 2B827T4
Số đăng kí KHXB : 01-2014/CXB/111-1062/GD

In ..... bản (....), khổ 17 x 24 cm.
In tại nhà in .....
Số in : ..... Số XB : /CXB/ /GD
In xong và nộp lưu chiểu ngày ... tháng .... năm 2014.