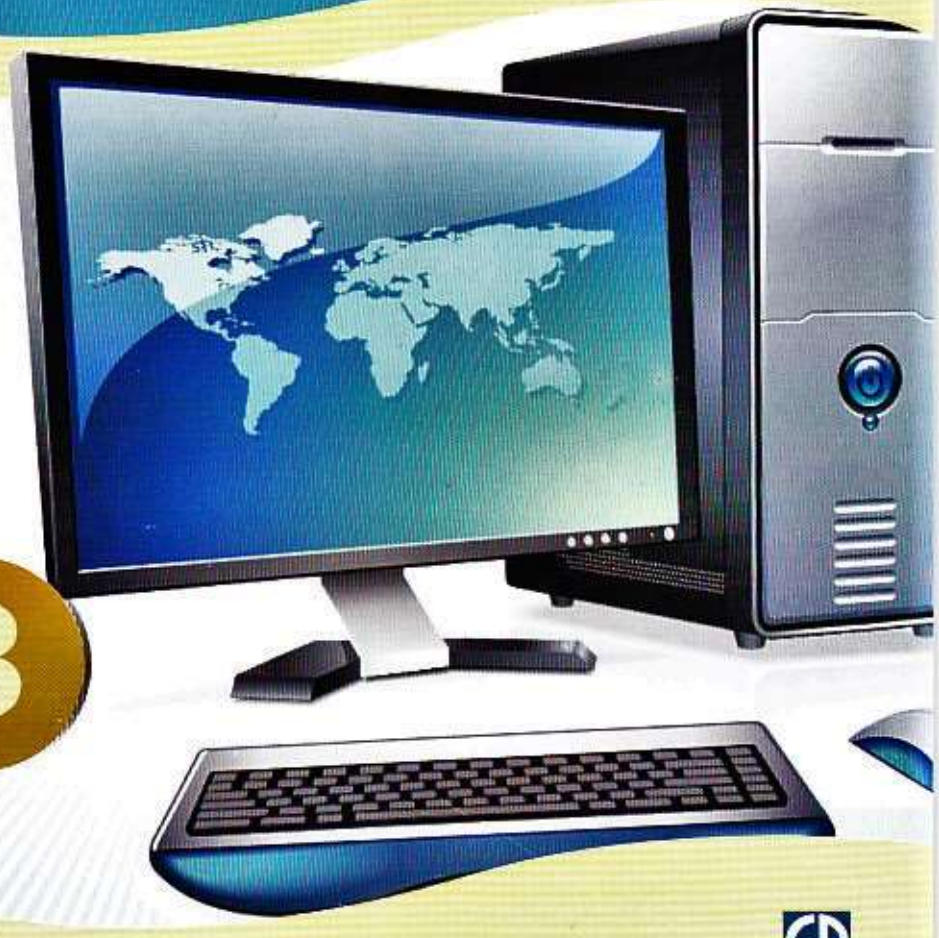
BỘ GIÁO DỤC VÀ ĐÀO TẠO

TIN HỌC

DÀNH CHO TRUNG HỌC CƠ SỞ



QUYỂN 3

NHÀ XUẤT BẢN GIÁO DỤC VIỆT NAM

MỤC LỤC

CHƯƠNG I. LẬP TRÌNH ĐƠN GIẢN

CHƯƠNG II. PHẦN MỀM HỌC TẬP

LỜI NÓI ĐẦU

Nhằm đáp ứng yêu cầu có một bộ sách Tin học với nội dung được cập nhật theo kịp sự phát triển của công nghệ, Nhà xuất bản Giáo dục Việt Nam đã phối hợp với các tác giả thực hiện việc chỉnh sửa, nâng cấp bộ sách *Tin học dành cho Trung học cơ sở*. Những nội dung liên quan đến các phần mềm phiên bản cũ và lạc hậu đã được viết lại trên cơ sở sử dụng những phiên bản phần mềm mới hơn đang được dùng phổ biến hiện nay. Trên tinh thần giảm tải, bên cạnh việc tăng cường các ví dụ và hoạt động định hướng kiến thức, phát triển năng lực, các tác giả đã cố gắng đưa vào một số nội dung mới nhằm tạo điều kiện tốt hơn cho việc dạy và học. Cụ thể như sau:

- Bổ sung một số hoạt động khởi động tại đầu mỗi bài học nhằm tạo tâm thế vui vẻ, kích thích trí tò mò, khơi gợi động cơ giúp học sinh mong muốn tham gia vào quá trình học tập. Các thầy cô giáo có thể thay thế bằng các nội dung khác phù hợp hơn với điều kiện cụ thể của nhà trường và địa phương.
- Thêm mục "Tìm hiểu mở rộng" ở cuối mỗi bài học nhằm giúp các em học sinh tìm hiểu và mở rộng thêm kiến thức của mình khi có nhu cầu và điều kiện, cũng như biết vận dụng những kiến thức đã học trên lớp vào giải quyết các vấn đề của cuộc sống hằng ngày. Nội dung phần này không yêu cầu thực hiện trong giờ học, là kiến thức không bắt buộc, chỉ khuyến khích thực hiện ở ngoài lớp học.

Nội dung chính của mỗi bài học được trình bày theo trật tự lôgic của vấn đề, đảm bảo chuẩn kiến thức và kĩ năng theo chương trình giáo dục hiện hành của Bộ Giáo dục và Đào tạo. Hiệu quả nhất để dạy những nội dung kiến thức này là được giảng dạy ngay tại phòng máy tính. Phần câu hỏi và bài tập, các thầy cô có thể hướng dẫn để các em trả lời hoặc thực hành ngay trên lớp hoặc bên ngoài thời gian lớp học.

Toàn bộ các phần mềm được sử dụng trong bộ sách này chỉ có tính minh hoạ cho chức năng mà học sinh cần được tiếp cận. Do vậy, các thầy cô giáo có thể sử dụng những phiên bản phần mềm phù hợp với điều kiện thực tế hoặc các phần mềm mã nguồn mở tương đương khác để phục vụ công việc giảng dạy của mình.

Giáo viên và học sinh có thể truy cập vào địa chỉ trang web ***http://sach24.vn/Tin-THCS*** để tìm một số học liệu điện tử giúp cho việc dạy và học hiệu quả hơn.

Mặc dù đã rất cố gắng, song bộ sách chắc chắn vẫn không tránh khỏi những điểm còn hạn chế, thiếu sót. Các tác giả mong nhận được những ý kiến đóng góp của các thầy cô giáo, các em học sinh và của các độc giả. Mọi góp ý xin gửi về địa chỉ:

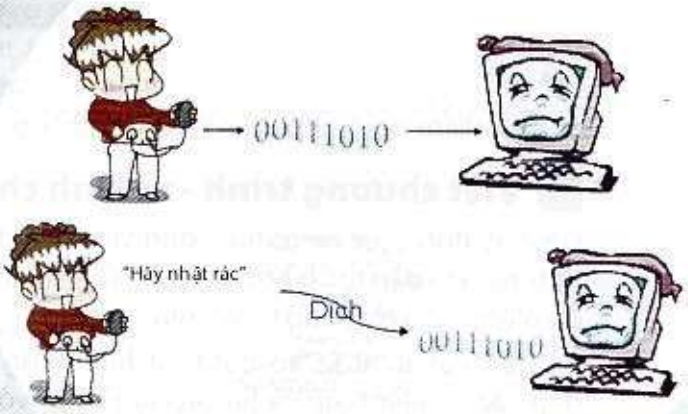
Ban Toán-Tin, Công ty Cổ phần Dịch vụ xuất bản Giáo dục Hà Nội – Nhà xuất bản Giáo dục Việt Nam, tầng 4, toà nhà Diamond Flower, số 1 Hoàng Đạo Thuý, quận Thanh Xuân, Hà Nội.

Xin trân trọng cảm ơn!

Các tác giả

CHƯƠNG I

LẬP TRÌNH ĐƠN GIẢN

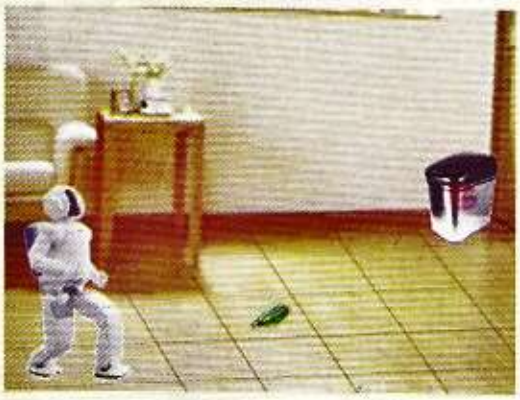


BÀI 1 MÁY TÍNH VÀ CHƯƠNG TRÌNH MÁY TÍNH

- ☑ ***Con người chỉ dẫn cho máy tính thực hiện công việc thông qua các lệnh.***
- ☑ ***Chương trình là bản hướng dẫn cho máy tính thực hiện những nhiệm vụ cụ thể.***

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 1.1 mô tả vị trí của rô-bốt, rác và thùng rác. Dưới đây là một 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 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 1.1. *Rô-bốt "nhặt rác"*

Em có cách hướng dẫn nào khác cho rô-bốt thực hiện công việc đó không?

1 Viết chương trình - ra lệnh cho máy tính làm việc

Để máy tính thực hiện một công việc nào đó, con người cần đưa cho máy tính các chỉ dẫn thích hợp (câu lệnh). Máy tính sẽ lần lượt thực hiện các lệnh đó giống như rô-bốt ở ví dụ trên.

Để tránh phải "nhắc" rô-bốt thực hiện từng câu lệnh, người ta thường tập hợp các câu lệnh đó và lưu trong rô-bốt với tên "Hãy nhặt rác". Khi đó chỉ cần ra lệnh "Hãy nhặt rác", rô-bốt sẽ tự động thực hiện các lệnh đó để bỏ rác vào thùng. Việc tập hợp các lệnh để điều khiển rô-bốt như vậy chính là viết chương trình. Tương tự, để điều khiển máy tính làm việc, chúng ta cũng viết chương trình máy tính.

Chương trình máy tính là một dãy các câu lệnh mà máy tính có thể hiểu và thực hiện được.

Tại sao cần viết chương trình?

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 hoàn thành công việc. Vì thế, để khai thác triệt để tốc độ của máy tính việc viết nhiều lệnh và tập hợp lại trong một chương trình giúp con người điều khiển máy tính một cách đơn giản và hiệu quả hơn.

Khi thực hiện chương trình, máy tính sẽ thực hiện các câu 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.

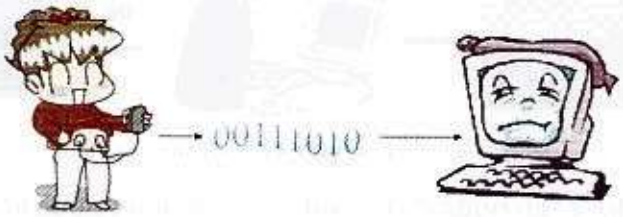
Trở lại ví dụ về rô-bốt nhặt rác, chương trình có thể có các lệnh như sau:

```
Hãy nhặt rác
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 1.2. *Ví dụ về chương trình*

2 Chương trình và ngôn ngữ lập trình

Chúng ta đã biết để máy tính có thể xử lí, thông tin đưa vào máy 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 1.3. *Minh hoạ ngôn ngữ máy*

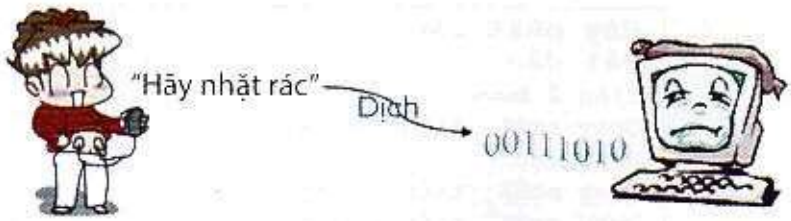
Tuy nhiên, việc viết chương trình bằng ngôn ngữ máy rất khó khăn và mất nhiều thời gian, công sức. Bởi lẽ, về mặt trực quan, các câu lệnh được viết dưới

dạng 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 bằng 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 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 1.4. *Minh hoạ vai trò chương trình dịch*

Như vậy, 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.

Chương trình viết bằng ngôn ngữ lập trình cần được chuyển thành ngôn ngữ máy.

Hình 1.5. *Các bước tạo chương trình máy tính*

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 thường được gọi chung là chương trình.

Chương trình có thể được soạn thảo nhờ một chương trình soạn thảo (tương tự như phần mềm soạn thảo Word). 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 lập trình phổ biến là Turbo Pascal và Free Pascal.

CÂU HỎI VÀ BÀI TẬP

1. Khi soạn thảo văn bản trên máy tính và yêu cầu chương trình tìm kiếm một cụm từ trong văn bản và thay thế bằng một cụm từ khác, thực chất ta đã yêu cầu máy tính thực hiện những lệnh gì? Có thể thay đổi thứ tự những lệnh đó mà vẫn không thay đổi kết quả được không?
2. 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 với thay đổi trên. Em hãy bổ sung hai lệnh để đưa rô-bốt trở lại vị trí ban đầu.
3. Hãy cho biết lí do cần phải viết chương trình để điều khiển máy tính.
4. Tại sao người ta phải tạo ra các ngôn ngữ lập trình trong khi có thể điều khiển máy tính bằng ngôn ngữ máy?
5. Chương trình dịch là gì?

TÌM HIỂU MỞ RỘNG

Có nhiều ngôn ngữ lập trình khác nhau, có thể kể tên một số ngôn ngữ phổ biến hiện nay như C, Java, Basic, Pascal,...

Em có thể kể tên một số ngôn ngữ lập trình khác không?

BÀI 2 LÀM QUEN VỚI CHƯƠNG TRÌNH VÀ NGÔN NGỮ LẬP TRÌNH

- ☑ *Ngôn ngữ lập trình là gì?*
- ☑ *Từ khoá của ngôn ngữ lập trình.*
- ☑ *Cấu trúc chung của một chương trình máy tính.*

Hình 1.6 minh hoạ một chương trình đơn giản được viết bằng ngôn ngữ lập trình Pascal.

Chương trình này chỉ có năm dòng lệnh, trong đó Program là câu lệnh khai báo tên chương trình; uses là câu lệnh khai báo tên công cụ có sẵn được sử dụng trong chương trình; writeln là câu lệnh in ra màn hình. Quan sát chương trình trên và trả lời các câu hỏi sau:

```
Program CT_Dau_tien
Uses Crt;
Begin
Writeln('Chao cac ban');
End.
```

Hình 1.6

1. Tên của chương trình là gì?
2. Công cụ có sẵn nào được sử dụng trong chương trình?
3. Dòng chữ nào sẽ được in ra màn hình?

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

1 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 ở hình 1.6 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. Câu lệnh đầu tiên trong ví dụ trên là câu lệnh đặt tên (khai báo) cho chương trình, câu lệnh thứ tư chỉ thị cho máy tính hiện ra màn hình dòng chữ *"Chao cac ban"*,...

Ngôn ngữ lập trình:
- Bảng chữ cái;
- Các quy tắc để viết các câu lệnh.

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.

2 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 tùy theo mỗi ngôn ngữ lập trình.

Từ khoá 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.

Trong ví dụ trên, ngoài các từ khoá **program**, **uses** em đã biết, các từ khoá **begin** và **end** dùng để thông báo điểm bắt đầu và kết thúc phần thân của chương trình.

Cùng với các từ khoá, chương trình ở hình 1.6 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 ở hình 1.6.

Từ khoá: Từ dành riêng của ngôn ngữ lập trình.

Tên: Do người lập trình đặt cho các đối tượng, đại lượng trong chương trình.

Tên do người lập trình đặt phải tuân thủ theo 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 người ta thường đặt tên sao cho *ngắn gọn, dễ nhớ* và *dễ hiểu*.

Ví dụ : 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.

3 Cấu trúc chung của chương trình

Cấu trúc chung của mọi chương trình gồm:

Cấu trúc chương trình:
Phần khai báo
Phần thân

- *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 1.6, ta có thể thấy:

- Phần khai báo gồm hai câu 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*"

Phần khai báo

```
Program CT_Dau_tien;
uses Crt;
Begin
Writeln('Chao cac ban');
End.
```

Phần thân chương trình

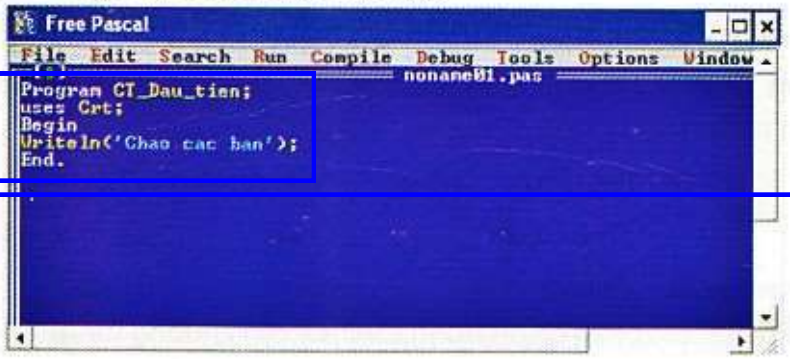
Hình 1.7

4 Ví dụ về ngôn ngữ lập trình

Trong mục 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 *Free Pascal*.

Khi khởi động phần mềm Free Pascal, cửa sổ soạn thảo chương trình hiện ra như hình 1.8 dưới đây. Ta có thể sử dụng bàn phím để soạn thảo chương trình tương tự như soạn thảo văn bản với Word.



Hình 1.8

Sau khi đã soạn thảo xong, nhấn tổ hợp phím **Alt+F9** để dịch chương trình. Chương trình yêu cầu nhập tên chương trình, em gõ tên **ctdt.pas**. 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 1.9.

Hình 1.9

Để chạy chương trình, ta nhấn tổ hợp phím **Ctrl+F9**. Trên cửa sổ sẽ hiện ra kết quả làm việc của chương trình, dòng chữ *"Chao cac ban"* như hình 1.10.

Hình 1.10

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 phần chính trong cấu trúc của chương trình.

5. Hãy cho biết các chương trình Pascal sau đây có hợp lệ không, tại sao?

Chương trình 1

```
begin
end.
```

Chương trình 2

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

TÌM HIỂU MỞ RỘNG

Pascal là một trong những ngôn ngữ lập trình hướng cấu trúc được dùng rộng rãi trong các nhà trường. Em hãy tìm hiểu để biết:

1. Các ưu điểm của ngôn ngữ lập trình Pascal.

2. Khái niệm ngôn ngữ lập trình hướng cấu trúc và tên của một vài ngôn ngữ lập trình loại này.

BÀI THỰC HÀNH 1 LÀM QUEN VỚI FREE PASCAL

1. Mục đích, yêu cầu

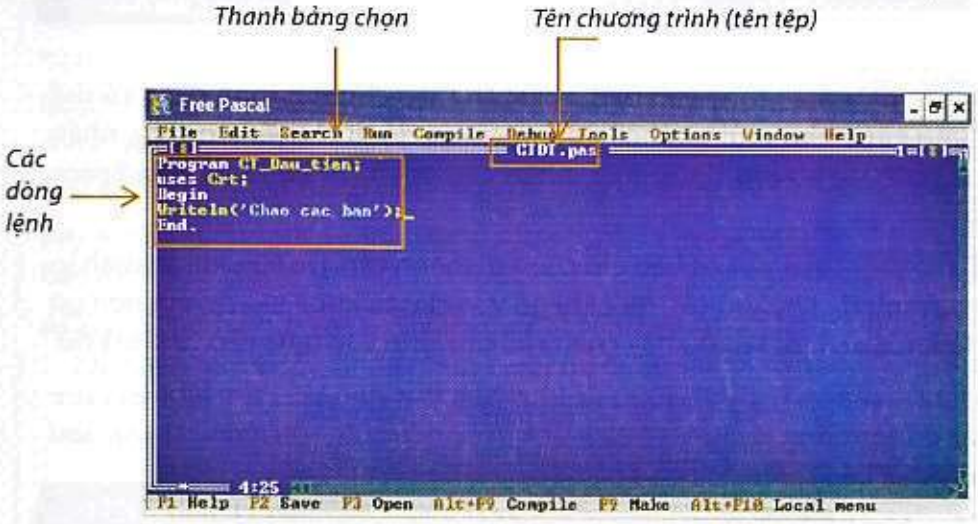
- Bước đầu làm quen với môi trường lập trình Free Pascal, nhận diện màn hình soạn thảo, cách mở các bảng chọn và chọn lệnh.
- Gõ được một chương trình Pascal đơn giản.
- Biết cách dịch, sửa lỗi trong chương trình, chạy chương trình và xem kết quả.

2. Nội dung

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

a) Khởi động Free Pascal bằng cách nháy đúp chuột lên biểu tượng  trên màn hình nền ;

b) Quan sát màn hình của Free Pascal và so sánh với hình 1.11 dưới đây:



Hình 1.11

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

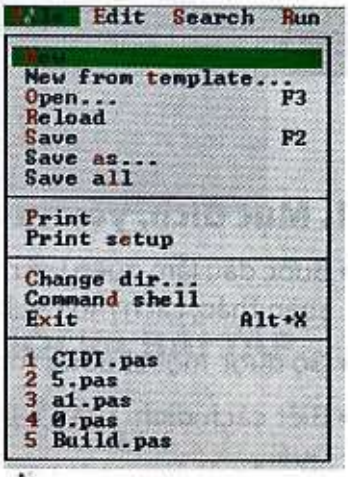
d) Nhấn phím **F10** để mở bảng chọn, sử dụng các phím mũi tên ← và → để di chuyển qua lại giữa các bảng chọn.

e) Nhấn phím **Enter** để mở một bảng chọn.

f) Quan sát các lệnh trong từng bảng chọn.

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

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



Hình 1.12

Có thể sử dụng 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,...) để mở bảng chọn tương ứng.

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

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

```
program CT_Dau_Tien;
uses crt;
begin
        clrscr;
        writeln('Chao cac ban');
        write('Minh la Free Pascal');
end.
```

Khi soạn thảo chương trình, cần gõ đúng, chính xác lệnh và không bỏ sót các dấu nháy đơn ('), dấu chấm phẩy (;), dấu chấm(.).

Chú ý

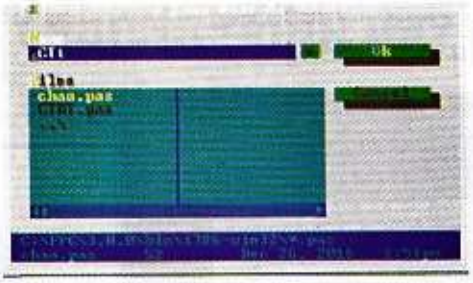
- Tương tự như soạn thảo văn bản, khi soạn thảo chương trình cũng có thể sử dụng các phím mũi tên hoặc dùng chuột để di chuyển con trỏ, nhấn phím **Enter** để xuống dòng mới, nhấn các phím **Delete** hoặc **BackSpace** để xoá.

- 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 cặp 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 **Save File As** hiện ra, gõ tên tệp (ví dụ CT1) trong ô **Name** (phần mở rộng ngầm định là **.pas**) và nhấn phím **Enter** (hoặc nháy **OK**).

Nên tắt chế độ gõ tiếng Việt khi soạn thảo chương trình.



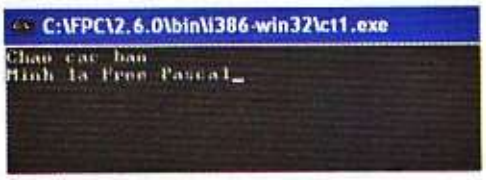
Hình 1.13

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 1.14 sau đây:

Hình 1.14

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 1.15

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.

Để quan sát được kết quả mà không cần nhấn tổ hợp phím Alt + F5, ta thêm lệnh `readln;` vào trước từ khoá `End`.

Lưu ý: Có thể dùng bảng chọn Run để chạy chương trình.

Bài 3. Tìm hiểu một số lỗi trong chương trình và thông báo lỗi.

a) Xoá dòng lệnh `begin`. Dịch chương trình và quan sát thông báo lỗi như hình 1.16 dưới đây, nhấn phím Enter để quan sát lỗi rõ hơn.

Hình 1.16

b) Nhấn phím bất kì và gõ lại lệnh **begin**. Xoá dấu chấm sau chữ **end**. Dịch chương trình và quan sát thông báo lỗi (h.1.17).

Hình 1.17

- Trong Pascal, dấu chấm phẩy để phân cách các lệnh. Sau câu lệnh trước từ khoá End có thể không cần dấu chấm phẩy.
- Sau từ khoá End kết thúc chương trình luôn có một dấu chấm đi kèm.

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

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

Chao cac ban
Toi ten la Pham Nhu Anh

TỔNG KẾT

1. Các bước đã thực hiện:
   ❶ Khởi động Free Pascal;
   ❷ Soạn thảo chương trình;
   ❸ Biên dịch chương trình: Alt + F9;
   ❹ 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**.
4. Dấu chấm phẩy (;) được dùng để phân cách các lệnh trong Pascal.
5. 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.
6. Lệnh **writeln** thông báo 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 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 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 3 CHƯƠNG TRÌNH MÁY TÍNH VÀ DỮ LIỆU

☑ *Một số kiểu dữ liệu cơ bản trong ngôn ngữ lập trình.*

☑ *Tương tác người-máy.*

Bằng những hiểu biết toán học thông thường, em có thể dễ dàng xác định được các phép toán sau có nghĩa hay không có nghĩa:

a) 5.1 > 5 b) 4 + 7

c) 20 - "Giai điệu tự hào" d) 6.5 mod 3.

Các ví dụ trên cho thấy, các phép toán số học như cộng, trừ, nhân, chia hoặc phép so sánh có thể được thực hiện với các số. Một số phép toán số học khác chỉ thực hiện được với các số nguyên như phép lấy phần dư, phép chia lấy phần nguyên. Những nguyên tắc kiểu như vậy cũng được quy định một cách chặt chẽ trong các ngôn ngữ lập trình.

1 Dữ liệu và kiểu dữ liệu

Em đã biết 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 1.18 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 1.18

Các kiểu dữ liệu 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 có thể của dữ liệu và các phép toán có thể thực hiện trên các dữ liệu 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" của ngôn ngữ lập trình.
- *Xâu kí tự* (hay *xâu*) là dãy các "chữ cái" lấy từ bảng chữ cái của ngôn ngữ lập trình, ví dụ: *"Chao cac ban"*, *"Lop 8E"*, "2/9/1945"...

Trong các ngôn ngữ lập trình, dữ liệu kiểu số nguyên còn 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 1.1 dưới đây liệt kê một số kiểu dữ liệu cơ bản của ngôn ngữ lập trình Pascal:

Tên kiểu dữ liệu	Phạm vi giá trị
`Byte`	Các số nguyên từ 0 đến 255
`Integer`	Số nguyên trong khoảng −32768 đến 32767
`real`	Số thực có giá trị tuyệt đối trong khoảng $1,5\times10^{-45}$ đến $3,4\times10^{38}$ và số 0.
`char`	Một kí tự trong bảng chữ cái
`String`	Xâu kí tự, tối đa gồm 255 kí tự

Bảng 1.1

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 1.2

Chúng ta đã quen thuộc với các phép toán cộng, trừ, nhân và chia. Tuy nhiên, cần lưu ý hầu hết các ngôn ngữ lập trình đều xem kết quả chia hai số n và m (tức là n/m) là số thực, cho dù n và m là các số nguyên và n 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)`
$\frac{x+5}{a+3} - \frac{y}{b+5}(x+2)^2$	`(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 đơn (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 đơn* 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 1.3 dưới đây cho biết các 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 1.3

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 ở bảng 1.3. 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 lại 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 Pascal

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

in kết quả tính diện tích hình tròn ra màn hình như hình 1.19 dưới đây:

Hình 1.19

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 nhập một số tự nhiên ứng với năm sinh. Sau khi nhấn phím Enter để xác nhận, chương trình sẽ tiếp tục hoạt động.

Hình 1.20

Hai câu lệnh Pascal dưới đây sẽ cho kết quả như hình 1.20:

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

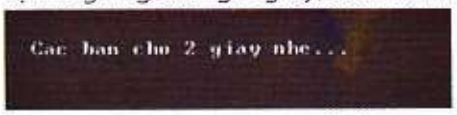
c) Tạm ngừng chương trình

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 ghi 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 1.21

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

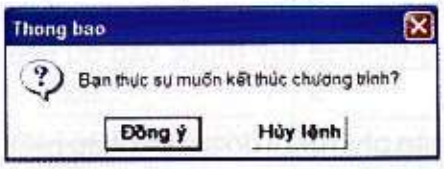
```
writeln('So Pi = ',Pi);
readln;
```

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 1.22

d) Hộp thoại

Một số môi trường lập trình cho phép sử dụng hộp thoại như một công cụ giao tiếp người-máy tính trong khi chạy chương trình. Ví dụ, khi người dùng muốn thoát khỏi một chương trình đang chạy, hộp thoại như sau có thể xuất hiện:



Hình 1.23

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.

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 2017 có thể thuộc những kiểu dữ liệu nào?
3. Cho hai xâu kí tự *"Lớp"* và *"8A"*. Hãy thử định nghĩa một "phép toán" có thể thực hiện được trên hai xâu kí tự đó.

4. 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);`

5. Viết các biểu thức toán dưới đây với 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) $(a^2+b)(1+d)^3$.

6. 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).`

7. 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) $x>10-3x$.

8. Viết các biểu thức ở bài tập 7 theo quy ước của Pascal.

TÌM HIỂU MỞ RỘNG

Khi học môn Toán em đã quen thuộc với các số nguyên, số thực cùng với các phép toán số học và phép so sánh trên tập hợp các số đó. Phép toán cộng và phép so sánh cũng có thể định nghĩa và có ý nghĩa trên tập hợp các kí tự và xâu kí tự. Em hãy tìm hiểu nhé.

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

Chỉ dùng dấu ngoặc đơn để nhóm các phép toán.

b) Khởi động Free 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.
```

Lệnh write hoặc writeln để in kết quả ra màn hình.

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 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.

Lệnh Readln trước từ khoá end để dừng màn hình xem kết quả.

Bài 3. Tìm hiểu thêm về cách ghi 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 làm tạm ngừng chương trình:
   - *delay(x)* tạm ngừng chương trình trong vòng *x* phần nghìn giây, sau đó tự động tiếp tục chạy.
   - *read* hoặc *readln* 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* quy định độ rộng in số, còn *m* là số chữ số thập phân. Lưu ý rằng các kết quả in ra màn hình được căn thẳng lề trái.

BÀI 4 SỬ DỤNG BIẾN VÀ HẰNG TRONG CHƯƠNG TRÌNH

- ☑ *Biến và hằng là gì?*
- ☑ *Cách sử dụng biến và hằng trong chương trình.*

Trong toán học em đã biết biến số (gọi tắt là biến) là một đại lượng có thể nhận các giá trị khác nhau và thường được dùng trong biểu diễn các hàm số, các biểu thức. Em có thể sử dụng các biến để viết công thức sau cho đơn giản hơn không?

$$\sqrt{\frac{15+\sqrt{20-4}}{\sqrt{20-4}}}\cdot\sqrt{\frac{11+\sqrt{20-4}}{\sqrt{20-4}}}+\sqrt{20-4}$$

Trong lập trình, biến cũng đóng một vai trò vô cùng quan trọng.

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* và dữ liệu được biến lưu trữ có thể thay đổi trong khi thực hiện chương trình.

Dữ liệu do biến lưu trữ được gọi là *giá trị của biến*.

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

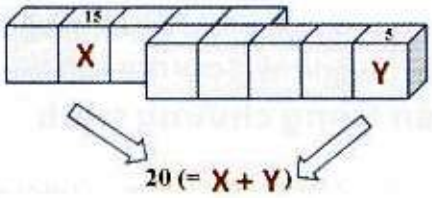
Trong Bài thực hành 2, em đã biết, để có kết quả của phép cộng 15 + 5 và in ra màn hình em có thể sử dụng câu lệnh Pascal sau đây:

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

Câu hỏi đặt ra ở đây là làm thế nào để in ra màn hình tổng của hai số mà giá trị của chúng không biết trước (các số là kết quả của một quá trình tính toán trung gian nào đó). Bằng cách sử dụng hai biến x, y để lưu giá trị của các số đó, câu lệnh sau đây sẽ in ra màn hình giá trị tổng của chúng:

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

Hình 1.24 minh hoạ trực quan việc lưu trữ các số 15 và 5 trong các ô nhớ có "tên" tương ứng là *X* và *Y* mà chương trình sẽ lấy ra để thực hiện phép cộng.



Hình 1.24

Ví dụ 1. Giả sử cần tính giá trị của các biểu thức $\frac{100+50}{3}$ và $\frac{100+50}{5}$ và 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.1.25):

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

Hình 1.25

2 Khai báo biến

Các biến dùng trong chương trình cần phải được khai báo ngay trong phần khai báo của chương trình.

Việc khai báo biến bao 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 2. Hình 1.26 là một ví dụ về cách khai báo biến trong Pascal:

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

Hình 1.26

Khai báo biến:
var tên biến : kiểu dữ liệu;

Trong ví dụ trên:

- *var* là từ khoá của Pascal dùng để khai báo biến,
- *m, n* là tên các biến có kiểu số nguyên (integer),
- *S, dientich* là tên các biến có kiểu số thực (real),
- *thong_bao* là tên biến có 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 các câu lệnh để tính toán hoặc xử lí chúng như với các giá trị dữ liệu (số, kí tự hay xâu,...). Điều phải lưu ý là để có các kết quả tính toán đúng mục tiêu của chương trình, cần phải gán các giá trị dữ liệu thích hợp cho các biến.

Như vậy các thao tác có thể thực hiện với các biến là:

- *Gán* giá trị cho biến;
- *Tính toán* với các biến.

Kiểu dữ liệu của giá trị được gán cho biến 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. Nói cách khác giá trị của biến có thể thay đổi.

Cú pháp của câu lệnh gán 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 bằng giá trị của biến y);

$i \leftarrow i+5$ (biến i nhận giá trị bằng giá trị hiện thời của i cộng thêm 5).

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

Ví dụ 3. 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*, *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*.

Câu lệnh gán:

Tên biến := **Biểu thức;**

Lưu ý giữa dấu : và dấu = không có kí tự trống, tức là phải viết liền nhau.

Giá trị của biến còn có thể gán nhờ các câu lệnh nhập dữ liệu `read` hoặc `readln`:

Ví dụ 4. Nhập giá trị cho các biến m, n bằng lệnh:

`read(m,n);` hoặc `readln(m,n);`

Khi gặp các câu lệnh trên trong chương trình, máy tính sẽ đợi ta gõ các giá trị tương ứng của các biến m và n từ bàn phím và nhấn phím Enter.

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;
```

Khai báo hằng:

const *tên hằng = giá trị;*

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 (trong ví dụ trên là 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 không thể dùng các 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;
```

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 và 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 3.1415 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 có thể 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*.

TÌM HIỂU MỞ RỘNG

1. Em đã biết để có các kết quả tính toán đúng mục đích của chương trình, cần phải gán các giá trị dữ liệu thích hợp cho các biến. Hãy chạy chương trình dưới đây để tìm hiểu ngay sau khi khai báo biến (trước khi gán giá trị dữ liệu cụ thể), biến có nhận giá trị dữ liệu ban đầu nào không? Nêu nhận xét của em về giá trị dữ liệu của biến ngay sau khi khai báo.

```
var A: integer; B: integer;
C: integer; D: integer;
begin
   writeln(A); writeln(B);
   writeln(C); writeln(D);
   readln;
end.
```

2. Với sự ra đời của khái niệm biến số toán học đã chuyển từ việc nghiên cứu các đại lượng cụ thể, rời rạc (các con số) sang việc nghiên cứu các đại lượng biến thiên. Do vậy, có thể nói không có khái niệm về biến không thể có sự phát triển của toán học hiện đại. Trong Tin học, biến cũng đóng vai trò vô cùng quan trọng. Không sử dụng biến không thể nào tự động hoá được quá trình xử lí thông tin. Điều thú vị là, nếu trong toán học, biến là một khái niệm trừu tượng, thì trong Tin học biến lại gắn với "khái niệm vật lí", cụ thể đó là địa chỉ một vùng nhớ hoàn toàn xác định được định danh bởi tên biến đã định nghĩa trong chương trình.

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

Xem lại các kiểu dữ liệu trong Free Pascal nêu trong Bài 3 để thực hành cách khai báo biến với các kiểu dữ liệu khác nhau.

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 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 trả 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.
```

Trong một chương trình chỉ cần khai báo một lần từ khoá var.

Ghi chú dùng khi người lập trình muốn đánh dấu, ghi nhớ đoạn chương trình đó làm gì, chúng được đặt ở giữa cặp dấu { và } hoặc (* và *). Khi chạy chương trình sẽ bỏ qua ghi chú.

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.
```

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. Kí hiệu := được sử dụng trong lệnh gán giá trị cho biến.

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, nói chung kết quả tính toán 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

- ☑ *Khái niệm về bài toán và xác định bài toán.*
- ☑ *Quá trình giải bài toán trên máy tính.*
- ☑ *Thuật toán và cách thức mô tả thuật 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 vận tốc 60 km/h là những ví dụ về bài toán.

Tuy nhiên, hàng ngày ta thường gặp và giải quyết các công việc đa dạng hơn nhiều nảy sinh từ nhu cầu thực tế: tính số gạch ít nhất phải mua để lát nền nhà, lập bảng điểm của lớp hoặc so sánh chiều cao của các bạn,... cũng là những ví dụ về bài toán.

Hãy nêu một vài bài toán em đã từng gặp và từng giải quyết trong cuộc sống thường ngày.

1 Xác định bài toán

Theo nghĩa rộng:

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

Xác định bài toán:
- Xác định điều kiện cho trước.
- Xác định kết quả cần thu được.

Để giải quyết được một bài toán cụ thể, người ta cần *xác định bài toán*, tức là phát biểu 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 nút 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 chiều cao tương ứng với cạnh đó;
- Kết quả cần thu được: Diện tích hình tam giác.

b) Đối với bài toán vượt qua 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 các bài toán.

2 Quá trình giải bài toán trên máy tính

Máy tính chỉ có thể thực hiện các công việc tiếp nhận, xử lí, biến đổi, tính toán, lưu trữ và biểu diễn thông tin thành dạng cần thiết dưới sự chỉ dẫn của con người thông qua các câu lệnh cụ thể.

Việc dùng máy tính giải một bài toán 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 thường 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 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. 1.27).

Hình 1.27

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

Như vậy, 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 Điều kiện cho trước - thông tin đã cho (INPUT) và đâu là Kết quả cần nhận được - 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 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 kèm theo 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.

Mô tả thuật toán: Liệt kê các bước theo thứ tự thực hiện.

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 khuấy mạnh để trộn trứng, muối và hành.

Bước 3. Cho một thìa dầu ăn vào chảo, đun nóng đều rồi đổ trứng 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.

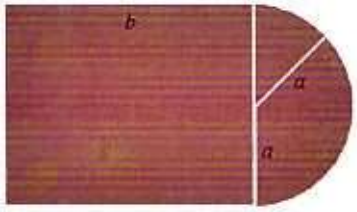
Rõ ràng, bất kì ai biết về các phép toán số học hay hiểu biết một chút về làm bếp, theo đúng trình tự và chỉ dẫn ở các bước trong các thuật toán nêu trên đều có thể tính ra nghiệm của phương trình đã cho hay tự làm cho mình món trứng tráng.

Tóm lại, có thể hiểu:

Thuật toán là dãy 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 1.28.



Hình 1.28

INPUT: Các 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 A.

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

Bước 1. $S_1 \leftarrow 2a \times b$ (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.

Trong biểu diễn thuật toán, người ta thường sử dụng kí hiệu ← để chỉ phép gán giá trị 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 $1 + 2 + \ldots + 100$.

Ở đây ta sẽ tính trực tiếp tổng cần tìm mà không sử dụng công thức toán học bằng cách 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.

...

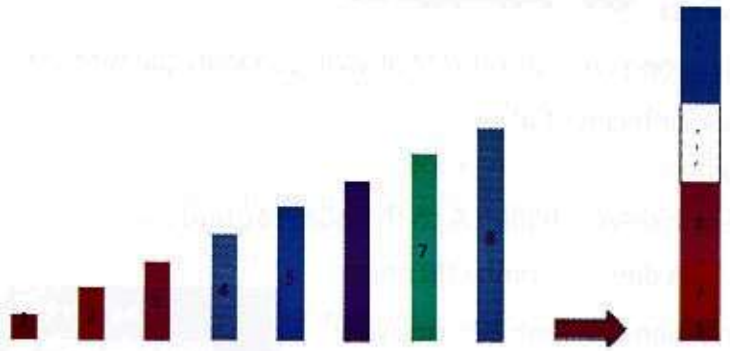
Bước 101. SUM ← SUM + 100.

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 ← 0.

Bước 2. SUM ← SUM + i; i ← i + 1.

Bước 3. Nếu $i \leq 100$, thì quay lại bước 2, ngược lại thông báo giá trị của SUM và kết thúc thuật toán.



Hình 1.29

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.

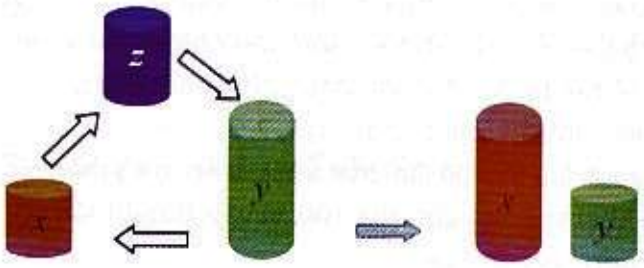
Trong Bài 2 của Bài thực hành 3, chúng ta đã tìm hiểu và viết chương trình hoán đổi các giá trị của hai biến x và y. Ví dụ này sẽ mô tả thuật toán để viết chương trình đó.

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 trước 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 1.30

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. Thoạt đầu ta thấy thuật toán sau đây có vẻ như có thể giải quyết bài toán này:

Bước 1. So sánh a và b. 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 là "a lớn hơn b", nhưng đến bước 2, khi kiểm tra $a < b$ không thoả mãn ta lại có tiếp kết quả là "a bằng b" và như thế ta nhận được hai kết quả.

Vì vậy, để có kết quả đúng, ta 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 giá trị phần tử 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*.

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

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

Từ đó, ta có thuật toán sau:

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

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

Bước 3. $i \leftarrow i + 1$.

Bước 4. Nếu $i \leq n$, 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 số bốn chú thỏ có trọng lượng tương ứng 2, 1, 5, 3 ki-lô-gam.

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

Hình 1.31

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.

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.

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.

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 các 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.

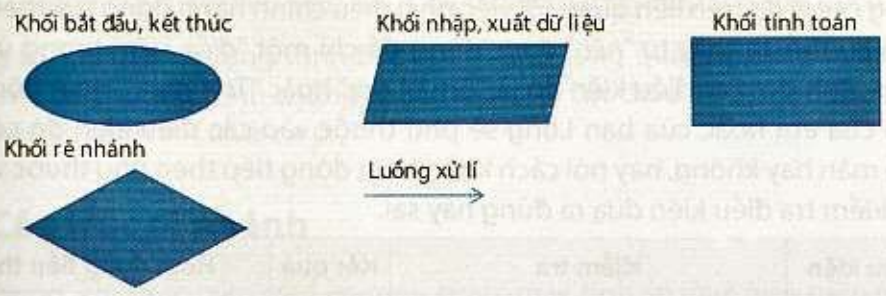
Bước 4. Thông báo giá trị *SUM* và kết thúc thuật toán.

6. Hãy mô tả thuật toán tính tổng các số dương trong dãy số $A = \{a_1, a_2, ..., a_n\}$ cho trước.

TÌM HIỂU MỞ RỘNG

1. Một trong những yêu cầu quan trọng của thuật toán và mô tả thuật toán là *tính dừng*, tức thuật toán phải được kết thúc sau một số *hữu hạn* bước. Việc mô tả thuật toán có *bước nhảy* (ví dụ, *chuyển đến bước 5*, *trở lại bước 2*) có thể gây khó khăn nhất định cho việc theo dõi tính dừng của thuật toán. Hãy tìm hiểu và cho ít nhất một ví dụ về thuật toán *không dừng*.

2. Để biểu diễn thuật toán theo sơ đồ khối, người ta thường phân biệt hai loại thao tác chính trong thuật toán: 1) Thao tác chọn lựa theo một điều kiện nào đó (được biểu diễn bằng khối hình thoi); 2) Các thao tác không thuộc loại chọn lựa được xếp vào loại hành động (được biểu diễn bằng các khối hình chữ nhật). Ngoài ra, người ta còn thường dùng các khối hình bình hành để biểu diễn thao tác nhập/xuất dữ liệu và khối elip để biểu diễn khối bắt đầu và kết thúc thuật toán (h.1.32).



Hình 1.32. Các khối quy ước dùng để biểu diễn thuật toán

Em có thể vẽ sơ đồ khối biểu diễn các thuật toán nêu trong bài học không?

BÀI 6 CÂU LỆNH ĐIỀU KIỆN

☑ ***Cấu trúc rẽ nhánh và hai dạng cấu trúc rẽ nhánh.***

☑ ***Câu lệnh điều kiện thể hiện cấu trúc rẽ nhánh.***

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 trước. Ví dụ:

- Mỗi sáng, em thức dậy, vệ sinh cá nhân, đến trường và vào lớp học,...
- Long thường đi đá bóng cùng các bạn vào sáng chủ nhật hàng tuần.

Tuy nhiên các hoạt động của con người thường bị tác động bởi sự thay đổi của 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 bố dọn dẹp nhà cửa.

Em có thể kể ra được những tình huống tương tự khác hay không?

1 Hoạt động phụ thuộc vào điều kiện

Trong các ví dụ trên liên quan tới việc phải điều chỉnh hành động tuỳ theo hoàn cảnh cụ thể, ta thấy từ "*nếu*" được dùng để chỉ một "*điều kiện*" tương ứng với hoàn cảnh đó. 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 được thoả mãn hay không, hay nói cách khác, hoạt động tiếp theo phụ thuộc vào kết quả kiểm tra điều kiện đưa ra đúng hay sai.

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*) in giá trị của X ra màn hình.
- *Nếu* nhấn phím **Pause/Break**, (*thì*) chương trình (sẽ bị) ngừng.

2 Điều kiện và phép so sánh

Để so sánh hai giá trị số hoặc hai biểu thức có giá trị số, chúng ta đã sử dụng các kí hiệu toán học như $=, \neq, <, \leq, >$ và $\geq$. Chúng ta cũng đã 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.

Điều kiện được biểu diễn bằng phép *so sánh*.

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 $c \neq 0$.

3 Cấu trúc rẽ nhánh

Nói chung, khi thực hiện một chương trình, máy tính sẽ *thực hiện tuần tự* các câu lệnh, từ trên xuống dưới. Để thay đổi trình tự ấy, ngôn ngữ lập trình có các câu lệnh cho phép máy tính thực hiện một 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 việc tính tiền cho khách hàng bằng các bước dưới đây:

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 trong ví dụ 2, nhưng chính sách khuyến mãi được thực hiện như sau: Nếu tổng số tiền mua từ 100 nghìn đồng trở lên, 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 giảm 10%.

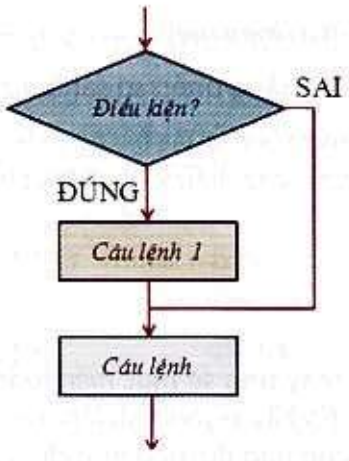
Khi đó, cần phải tính lại tiền cho khách trong cả hai trường hợp, tổng tiền không nhỏ hơn 100 nghìn đồng và tổng tiền nhỏ hơn 100 nghìn đồng. Thuật toán có thể được sửa lại như sau:

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

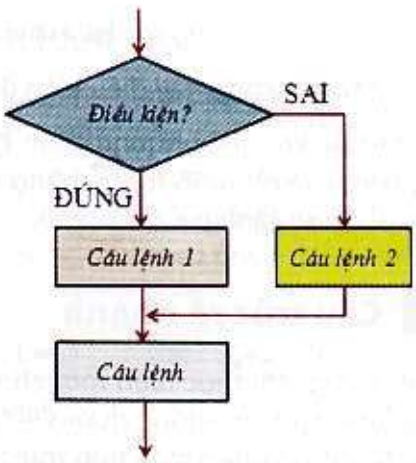
Bước 2. Nếu $T \geq 100000$, số tiền phải thanh toán là $70\% \times T$; Ngược lại, số tiền phải thanh toán là $90\% \times T$.

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. 1.33a), còn trong Ví dụ 3 được gọi là *cấu trúc rẽ nhánh dạng* ***đủ*** (h. 1.33b).



Hình 1.33a. Cấu trúc rẽ nhánh dạng thiếu



Hình 1.33b. Cấu trúc rẽ nhánh dạng đủ

Cấu trúc rẽ nhánh cho phép thay đổi *thứ tự thực hiện tuần tự* các bước trong thuật toán.

Cấu trúc rẽ nhánh giúp cho việc lập trình được linh hoạt hơn.

4 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. Giả sử cần in ra màn hình số lớn hơn trong hai số *a* và *b*:

Nếu a > b thì in ra màn hình giá trị của *a*.

Thể hiện bằng câu lệnh điều kiện dạng thiếu trong Pascal:

```
if a > b then write(a);
```

Ví dụ 5. Viết chương trình yêu cầu người dùng nhập một số không lớn hơn 5 từ bàn phím, chương trình sẽ kiểm tra tính hợp lệ, nếu không hợp lệ sẽ thông báo lỗi. Khi đó, chương trình có thể biểu diễn bằng thuật toán sau:

Bước 1. Nhập số *a*;

Bước 2. Nếu a > 5 thì thông báo lỗi;

Thể hiện bằng câu lệnh điều kiện dạng thiếu trong Pascal:

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

Ví dụ 6. 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 sẽ 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 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.

Các câu lệnh điều kiện trong ngôn ngữ lập trình thể hiện 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. Mỗi điều kiện hoặc biểu thức sau 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.

Hình 1.34

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 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;`

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

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

f) `if n>0 then begin a:=0; m:=-1 end else c:=a;`

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;`

7. Giả sử cần viết chương trình nhập một số tự nhiên vào máy tính và in ra màn hình kết quả số đã nhập là số chẵn hay lẻ, chẳng hạn *"5 là số lẻ"*, *"8 là số chẵn"*. Hãy mô tả các bước của thuật toán để giải quyết bài toán trên và viết chương trình Pascal để thực hiện thuật toán đó.

TÌM HIỂU MỞ RỘNG

1. Các câu lệnh điều kiện có thể sử dụng *lồng nhau* như trong ví dụ sau:

Ví dụ: Cho hai số thực *a* và *b*. Đoạn chương trình sau in kết quả so sánh hai số đó ra màn hình, chẳng hạn *"a > b"*, *"a < b"*, hoặc *"a = b"*:

```
if a>b then writeln ('a>b') else
    if a = b then writeln ('a = b') else writeln ('a<b');
```

2. Em hãy tìm hiểu thêm các ví dụ khác về các trường hợp sử dụng các câu lệnh lồng nhau.

3. Mỗi câu lệnh điều kiện đủ có thể được thay thế tương đương bằng hai câu lệnh điều kiện thiếu. Em hãy thử làm điều đó với một câu lệnh điều kiện đủ.

BÀI THỰC HÀNH 4 SỬ DỤNG CÂU LỆNH ĐIỀU KIỆN

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 một vài lần. Nhập 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 mot 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. 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*>

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 thành phần 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 là *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

- ☑ *Cấu trúc lặp.*
- ☑ *Câu lệnh* `for...do` *thể hiện cấu trúc lặp với số lần lặp cho trước.*

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

Có những hoạt động mà em thường thực hiện lặp lại với một số lần nhất định và biết trước, chẳng hạn đánh răng mỗi ngày hai lần, mỗi lần từ tầng 1 em phải bước lên 20 bậc cầu thang để tới phòng ngủ của em trên tầng 2,... Có những công việc em phải lặp đi lặp lại với số lần không thể xác định trước: học cho đến khi thuộc bài, nhặt từng cọng rau cho đến khi xong,...

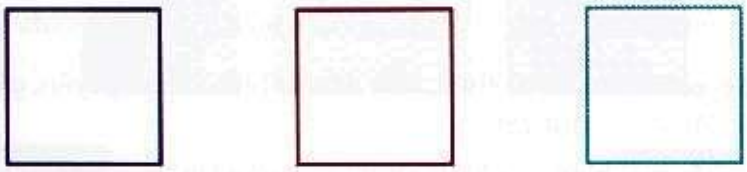
? Hãy nêu ví dụ khác về hoạt động lặp trong cuộc sống hằng ngày.

1 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 1.35. 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).

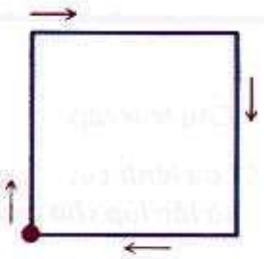
Bước 2. Nếu số hình vuông đã vẽ được ít hơn 3, di chuyển bút vẽ về bên phải 2 đơn vị và trở lại bước 1; ngược lại, kết thúc thuật toán.



Hình 1.35

Riêng với bài toán vẽ một hình vuông (h.1.36), 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 1.36

Bước 1. Đặt $k \leftarrow 0$ (k là số đoạn thẳng đã vẽ được).

Bước 2. Vẽ đoạn thẳng độ dài 1 đơn vị và quay thước 90° sang phải.

$k \leftarrow k+1$.

Bước 3. Nếu $k < 4$, 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.

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 các "cách" để chỉ thị cho máy tính thực hiện cấu trúc lặp với một câu lệnh. Đó là các *câu lệnh lặp*.

2 Câu lệnh lặp `for . . . do`

Các ngôn ngữ lập trình thường có nhiều dạng câu lệnh lặp. Một trong các 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

Câu lệnh trong vòng lặp không được phép làm thay đổi giá trị của biến đếm.

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:

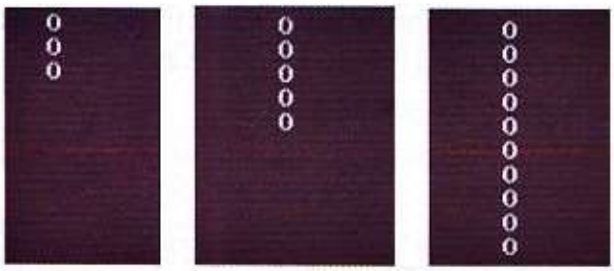
```
writeln('O');
```

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.
```

Câu lệnh ghép: Nhiều lệnh đặt trong cặp từ khoá `begin` và `end;`

Dịch và chạy chương trình này, ta sẽ thấy kết quả như ở hình1.37 dưới đây:



Hình 1.37

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.

3 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 là cũng 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:

$$N! = 1.2.3 \ldots N$$

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.
```

N! là số rất lớn so với N nên cần khai báo biến đủ lớn.

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. Chương trình Pascal sau đây thực hiện hoạt động nào?

```
var i: integer;
begin
for i:=1 to 1000 do;
end.
```

3. Hãy mô tả thuật toán để tính tổng A sau đây (n là số tự nhiên được nhập vào từ bàn phím):

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

TÌM HIỂU MỞ RỘNG

Ngoài lệnh lặp đã biết, Pascal còn có câu lệnh lặp tương tự:

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

Trong câu lệnh này, ban đầu *biến đếm* nhận *giá trị đầu*. Sau mỗi lần thực hiện *câu lệnh*, *biến đếm* bị *giảm* đi một đơn vị và *câu lệnh* được lặp lại tới khi *biến đếm* bằng *giá trị cuối*.

Ví dụ. Đoạn chương trình sau sẽ ghi trên màn hình các số từ 100 đến 1 theo thứ tự giảm dần:

```
writeln ('Dem nguoc');
for i := 100 downto 1 do writeln (i);
```

Nếu sử dụng lệnh **for...to** em phải làm thế nào? Hãy tìm hiểu cách thức sử dụng câu lệnh **for...downto** và thể hiện trong một chương trình cụ thể.

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 và tìm hiểu chương trình.

2. Nội dung

Bài 1. Viết chương trình in ra màn hình bản cửu chương của số *N* trong khoảng từ 1 đến 9, số được nhập từ bàn phím và dừng màn hình để có thể quan sát kết quả:

Hình 1.38

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 cú pháp, nếu có.

c) Chạy chương trình với các giá trị nhập vào lần lượt bằng 1, 2,..., 9. Quan sát kết quả nhận được trên màn hình.

Bài 2. Chỉnh sửa chương trình để làm đẹp kết quả trên màn hình.

Kết quả của chương trình nhận được trong bài 1 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.

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;
```

```
Nhap so N=7
Bang nhan 7
 7 x  1 =  7
 7 x  2 = 14
 7 x  3 = 21
 7 x  4 = 28
 7 x  5 = 35
 7 x  6 = 42
 7 x  7 = 49
 7 x  8 = 56
 7 x  9 = 63
 7 x 10 = 70
```

Hình 1.39

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 3. Cũng như câu lệnh `if`, có thể dùng câu lệnh `for` lồng bên 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:

```
 0  1  2  3  4  5  6  7  8  9
10 11 12 13 14 15 16 17 18 19
20 21 22 23 24 25 26 27 28 29
30 31 32 33 34 35 36 37 38 39
40 41 42 43 44 45 46 47 48 49
50 51 52 53 54 55 56 57 58 59
60 61 62 63 64 65 66 67 68 69
70 71 72 73 74 75 76 77 78 79
80 81 82 83 84 85 86 87 88 89
90 91 92 93 94 95 96 97 98 99
```

Hình 1.40

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.
```

Dữ liệu kiểu `byte` là kiểu số nguyên nhận giá trị từ 0 đến 255.

b) Gõ, dịch 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 cho 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

☑ *Cấu trúc lặp với số lần lặp không xác định trước.*

☑ *Câu lệnh lặp với số lần chưa biết trước* `While . . . do`.

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. Tuy nhiên, trong thực tế có nhiều hoạt động được thực hiện lặp đi lặp lại với số lần chưa được biết trước.

Ví dụ bạn Long gọi điện hẹn Trang tới thăm nhà cô giáo cũ vào chủ nhật tới. Long quyết định cứ 10 phút gọi điện một lần cho Trang cho đến khi có người thưa máy. Rõ ràng là không thể biết trước Long sẽ phải quay số điện thoại nhà Trang mấy lần: có thể một lần, có thể hai hoặc nhiều hơn nữa. Điều kiện để kết thúc hoạt động lặp đó là *có người thưa máy*.

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 nhất định nhiều lần.

? Em hãy phân tích và cho biết kết quả của phần thuật toán sau là gì ? Người dùng sẽ phải nhập số *n* từ bàn phím bao nhiêu lần?

Bước 1. Nhập số *n* từ bàn phím.

Bước 2. Nếu *n* <5 quay trở về bước *1*.

Bước 3. ...

1 Lệnh lặp với số lần chưa biết trước

Ví dụ 1. 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. $S \leftarrow 0, n \leftarrow 0$.

Bước 2. Nếu $S \leq 1000$ thì chuyển tới bước 3; Ngược lại ($S > 1000$) chuyển tới bước 4.

Bước 3. $n \leftarrow n + 1$; $S \leftarrow S + n$; 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 xác định trước, phụ thuộc vào một *điều kiện* ($S \leq 1000$) và chỉ dừng khi kết quả kiểm tra điều kiện đó là sai ($S > 1000$).

Để viết chương trình chỉ dẫn máy tính thực hiện các hoạt động lặp mà chưa xác định trước được số lần lặp, ta có thể sử dụng câu lệnh có dạng *lặp với số lần chưa xác định*.

Nói chung các ngôn ngữ lập trình đều có câu lệnh lặp dạng này. Cụ thể, câu lệnh lặp với số lần chưa xác định trước trong Pascal có dạng:

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

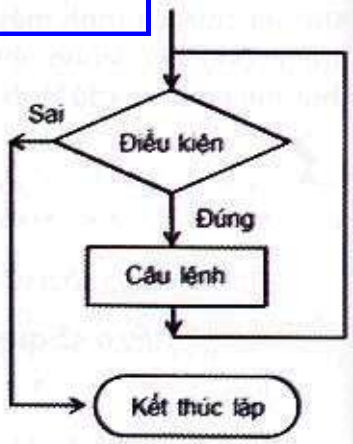
trong đó:

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

Câu lệnh 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 (h. 1.41).



Hình 1.41. Lệnh While-Do

Ví dụ 2. Chúng ta biết rằng, nếu n ($n > 0$) càng lớn thì $\frac{1}{n}$ càng nhỏ, nhưng 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 tìm 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:5:4, '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 3. Chương trình dưới đây thể hiện thuật toán tính số n trong Ví dụ 1:

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

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

Ví dụ 4. 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 cho 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`.

2 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 2, 3, 4 sau mỗi vòng lặp giá trị các biến *x*, *S*, *i* thay đổi (*x* giảm đi, còn *S* và *i* tăng lên) làm cho giá trị của điều kiện (*x*>= *sai_so* ở ví dụ 2; *S* <= 1000 ở ví dụ 3; *i*<=100 ở ví dụ 4) sẽ buộc phải thay đổi khi *x* đủ nhỏ, còn *S*, *i* đủ lớn.

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 lặp cho trước và câu lệnh lặp với số lần lặp 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. $n \leftarrow n + 3$, $S \leftarrow S - n$ và quay lại bước 2.

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

4. Hãy tìm hiểu mỗi đoạn chương trình Pascal 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;
```

TÌM HIỂU MỞ RỘNG

Câu lệnh lặp Repeat... until

Một câu lệnh lặp khác cũng thường hay được sử dụng trong Pascal là câu lệnh `repeat...until` có cú pháp như sau:

```
repeat
  <câu lệnh 1>;
  <câu lệnh 2>;...;
  <câu lệnh k>;
until <diều kiện>;
```

Khi gặp câu lệnh lặp này chương trình sẽ thực hiện các câu lệnh nằm giữa hai từ khoá `repeat` và `until`, sau đó kiểm tra *<điều kiện>*, nếu *<điều kiện>* sai thì tiếp tục thực hiện vòng lặp. Quá trình đó được lặp đi lặp lại cho tới khi nào *<điều kiện>* đúng thì kết thúc.

Đoạn chương trình sau cho cùng kết quả như ví dụ 4 ở trên:

```
T:=0;
i:=1;
repeat
   T:=T+1/i;
   i:=i+1;
until i > 100;
writeln(T);
```

1. Em hãy viết chương trình Pascal với câu lệnh repeat..until thể hiện thuật toán tính số n trong Ví dụ 1.

2. Hãy tìm hiểu và rút ra những điểm giống và khác nhau giữa các câu lệnh lặp `While...do` và `Repeat... until`.

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 không xác định 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 của *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 phai 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>*;

Đọc thêm. Tính gần đúng số π với độ chính xác cho trước

Người ta đã tìm ra công thức

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \ldots + \frac{1}{2n-1} - \frac{1}{2n+1} + \ldots$$

để tính gần đúng số π với *n* số hạng cho trước. Sử dụng lệnh **while...do**, ta còn có thể viết chương trình để tính gần đúng số π với độ chính xác theo yêu cầu (được nhập từ bàn phím):

```
uses crt;
var SoPi,saiso,Epsilon:real;
    n,i, dau: integer;
begin
clrscr;
write('Hay cho sai so de tinh gan dung so Pi =');
  readln(saiso);
  SoPi:=0; Epsilon:=3; i:=0; dau:=-1;
  while Epsilon>=saiso do
    begin dau:=dau*(-1); SoPi:=SoPi+dau*1/(2*i+1);
    Epsilon:=Abs(4*SoPi-Pi);i:=i+1 end; {Pi la ham chuan}
 writeln('So Pi gan bang ',SoPi*4);
readln;
end.
```

Lưu ý. Chương trình trên đã sử dụng hàm chuẩn **Abs** của Pascal. Hàm **Abs** cho kết quả là giá trị tuyệt đối của một số, tức **Abs(x)** cho giá trị **x**, nếu **x** ≥ 0; ngược lại **Abs** cho kết quả **−x**.

BÀI 9 LÀM VIỆC VỚI DÃY SỐ

- ☑ *Dữ liệu kiểu mảng.*
- ☑ *Làm việc với biến mảng.*
- ☑ *Sử dụng các biến kiểu mảng và câu lệnh lặp.*

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.

Việc giải bài toán trên gồm hai bước cơ bản:

1. Tính thu nhập trung bình bằng cách tính tổng thu nhập của tất cả các hộ gia đình rồi 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:

```
var a, Thunhap_TB: real; i: Integer;
begin
    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 la: ', a - Thunhap_TB)
    end;
end.
```

Em hãy tìm hiểu tác dụng của từng câu lệnh trong đoạn chương trình này và rút ra nhận xét của em.

1 Dãy số và biến mảng

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.

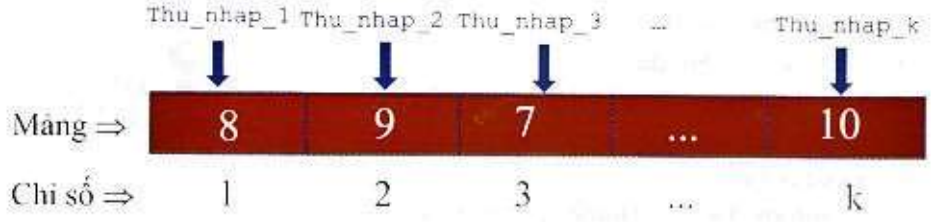
Để chỉ phải nhập dữ liệu một lần, ta có thể khai báo nhiều biến, mỗi biến dùng để lưu trữ thu nhập của một hộ gia đình. Ví dụ, trong Pascal ta cần nhiều câu lệnh khai báo và nhập dữ liệu như sau:

```
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 không hề thú vị.

Để tránh phải thực hiện điều tương tự như vậy, các ngôn ngữ lập trình đã đưa ra một kiểu dữ liệu đặc biệt – *kiểu mảng* để lưu nhiều dữ liệu liên quan đến nhau (như Thunhap_1, Thunhap_2,... ở trên) bằng một biến duy nhất và đánh "số thứ tự" cho các dữ liệu đó giúp cho việc xử lí các dữ liệu ấy đơn giản hơn.

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



Hình 1.42

Các phần tử có thể có cùng kiểu dữ liệu bất kì. 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 xếp 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 số tương ứng trong phần khai báo của chương trình.

Cách thức khai báo biến mảng trong các ngôn ngữ lập trình có thể khác nhau, nhưng 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 cùng kiểu dữ liệu số thực.

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

`var` <*tên biến 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 1. Tiếp tục với ví dụ trên, 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 từng 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 2. 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 nhiều 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 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 1.43

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.

Việc truy cập tới phần tử bất kì của mảng được thực hiện thông qua chỉ số tương ứng của phần tử đó trong mảng.

Chẳng hạn, ta có biến mảng *A*. Khi đó *A[i]* là phần tử thứ *i* của mảng đó.

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

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

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 3. Viết chương trình nhập *N* số nguyên từ bàn phím và in ra màn hình số nhỏ nhất và số lớn nhất cùng độ lệch của giá trị đó so với giá trị trung bình của *N* số đã nhập. *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; giatri_TB: Real;
  A: array[1..100] of integer;
```

Phần thân chương trình sẽ tương tự dưới đây:

```
Begin
  clrscr;
  write('Hay nhap do dai 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]; giatri_TB:=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];
             giatri_TB:= giatri_TB + a[i];
       end;
giatri_TB:= giatri_TB/N;
writeln('Gia tri Max:',Max,' hon gia tri TB:', max - giatri_TB:5:2);
writeln('Gia tri Min:',Min,' kem gia tri TB:', giatri_TB - min:5:2);
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 số được nhập vào sau này có thể nhỏ hơn nhiều so với 100).

CÂU HỎI VÀ BÀI TẬP

1. "Có thể xem biến mảng là một biến được tạo từ nhiều biến có cùng kiểu, nhưng chỉ có một tên duy nhất." Phát biểu đó đúng hay sai?
2. Hãy nêu các lợi ích của việc sử dụng biến mảng trong chương trình.
3. 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[4 .. 10] Of Integer;`

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

TÌM HIỂU MỞ RỘNG

Kiểu dữ liệu của biến mảng trong Pascal có thể là kiểu dữ liệu bất kì, không chỉ là dữ liệu kiểu số nguyên và số thực. Ví dụ sau đây là biến mảng có kiểu dữ liệu là kiểu xâu:

```
var Danhsach: array[1..20] of string;
```

Hãy tìm hiểu về biến mảng có các kiểu dữ liệu khác kiểu số và ứng dụng của chúng để giải quyết các bài toán thực tế.

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 > 8 điểm: Giỏi, từ 6,5 điểm đến 7,9 điểm: Khá, từ 5 điểm đến 6,5 điểm: Trung bình và dưới 5 điểm: Kém).

a) Xem lại các ví dụ 2 và ví dụ 3, Bài 9 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 và chạy thử 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]

CHƯƠNG II

PHẦN MỀM HỌC TẬP

BÀI 10 LÀM QUEN VỚI GIẢI PHẪU CƠ THỂ NGƯỜI BẰNG PHẦN MỀM ANATOMY

- ☑ *Quan sát các hệ giải phẫu cơ thể người như hệ xương, hệ cơ, hệ thần kinh,...*
- ☑ *Khám phá chức năng của một số bộ phận cơ thể người.*

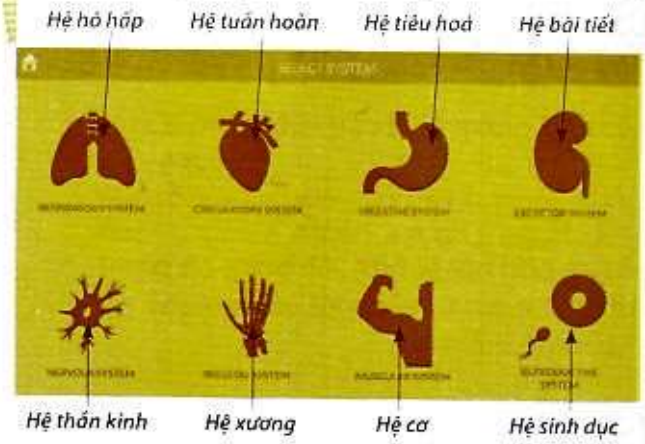
Trong bộ môn sinh học "Giải phẫu cơ thể người" người ta thường dùng hình ảnh hoặc mô hình để học sinh quan sát và qua đó bài học sẽ sống động, dễ hiểu hơn. Theo em, với mô hình như vậy ta có thể làm được gì?

Quan sát chi tiết bằng cách phóng to, thu nhỏ hình.	
Xoay mô hình để quan sát được từ nhiều hướng các bộ phận của cơ thể người.	
Quan sát từ bên ngoài hoặc từ bên trong các bộ phận cơ thể người.	
Tháo rời hoặc lắp ghép các bộ phận.	

Trong bài này chúng ta tìm hiểu về "giải phẫu cơ thể người" với phần mềm mô phỏng Anatomy. Phần mềm này đáp ứng tất cả các điều nói trên.

1 Cùng làm quen với phần mềm Anatomy

Mỗi khi khởi động phần mềm, màn hình chính có chứa hai nút lệnh **LEARN** (HỌC) và **EXERCISES** (BÀI TẬP) xuất hiện. Nháy nút lệnh LEARN để vào xem và học chi tiết giải phẫu cơ thể người. Màn hình có dạng như sau:

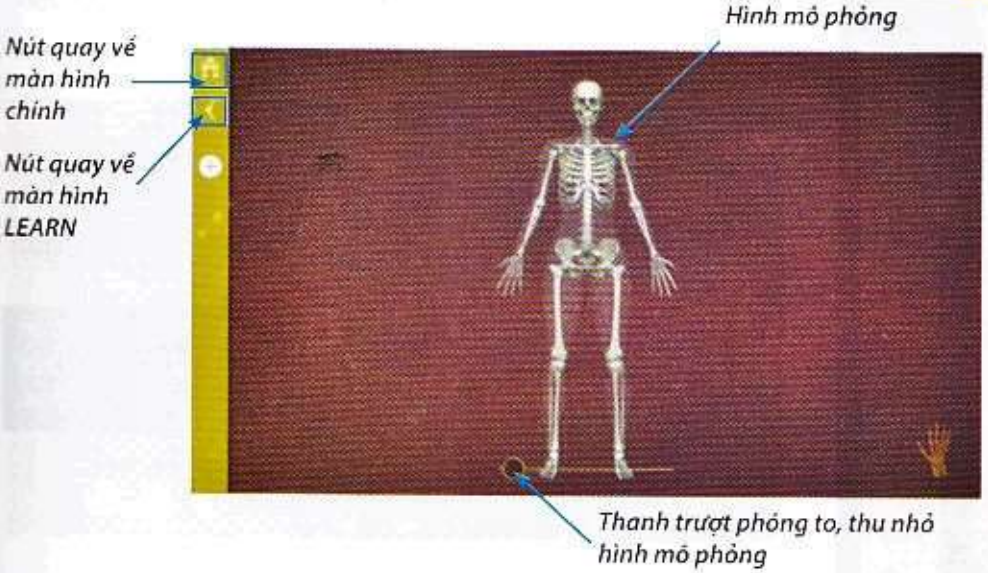


Hình 2.1. *Các hệ giải phẫu cơ thể người*

Chúng ta sẽ thấy tám biểu tượng tương ứng với tám chủ đề. Nháy chuột lên biểu tượng để vào chủ đề tương ứng.

2 Hệ xương

Nháy chuột vào biểu tượng có dòng chữ SKELETAL SYSTEM để tìm hiểu về hệ xương của con người. Màn hình xuất hiện như sau:



Hình 2.2. *Mô phỏng hệ xương*

a) Các thao tác trực tiếp trên hình mô phỏng

- Dịch chuyển mô hình lên, xuống trên màn hình: Kéo thả chuột theo chiều thẳng đứng (lên, xuống).
- Xoay mô hình xung quanh trục của mình: Kéo thả chuột theo chiều ngang, từ trái sang phải và ngược lại.
- Phóng to, thu nhỏ hình mô phỏng: Di chuyển nút tròn trên thanh trượt hoặc dùng nút cuộn của chuột.

Bằng các thao tác trên, ta có thể xem chi tiết bất cứ một phần hay bộ phận nào của hệ xương con người mà ta muốn khám phá. Ví dụ:

Các xương sườn *Xương chậu* *Xương đầu gối chân*

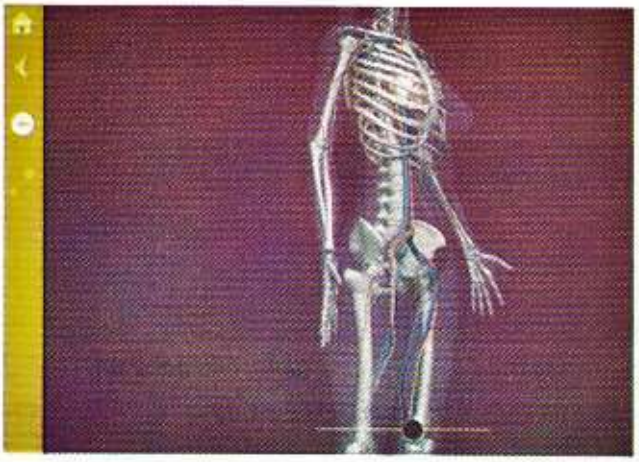
Hình 2.3. *Mô phỏng chi tiết một số bộ phận của hệ xương*

b) Bổ sung thêm các hệ khác vào hình mô phỏng

Trong quá trình đang quan sát và học về hệ xương của người, ta có thể hiển thị thêm các hệ khác. Đây là tính năng đặc biệt của phần mềm. Để kích hoạt tính năng này, ta nháy chuột vào nút  ở bên trái màn hình. Xuất hiện một bảng chọn ngay bên cạnh cho phép ta chọn bổ sung thêm các hệ cần xem.

Hình 2.4. *Bảng chọn xem nhiều hệ cơ quan cùng lúc*

Nháy chuột vào biểu tượng cần bổ sung trong bảng chọn trên để xem đồng thời các hệ trên hình mô phỏng. Em có thể chọn nhiều hệ cùng lúc bằng cách nháy chuột chọn. Ví dụ, em hiển thị thêm hệ tuần hoàn cùng hệ xương, sẽ quan sát được toàn bộ hệ xương có thêm hình ảnh trái tim, các mạch máu, động mạnh, tĩnh mạch.



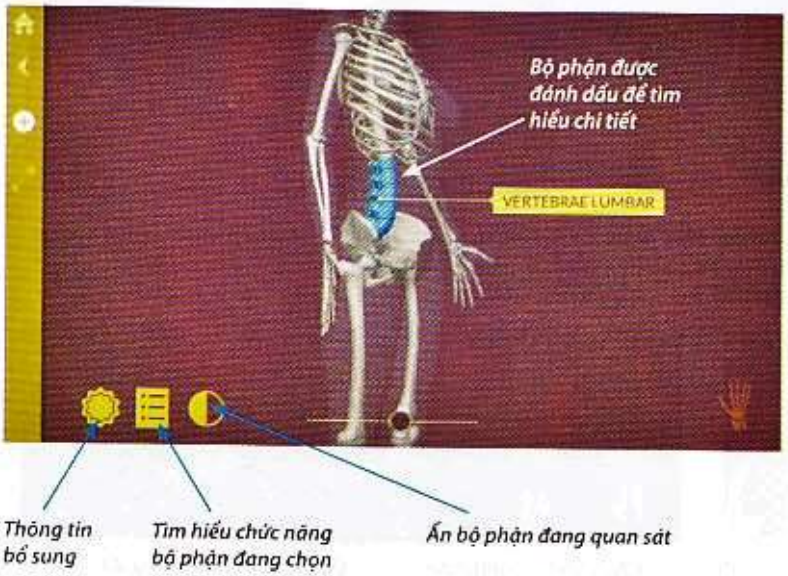
Hình 2.5. *Hiển thị đồng thời hệ xương và hệ tuần hoàn*

c) Quan sát chi tiết các hệ giải phẫu cơ thể người

Phần mềm cho phép quan sát chi tiết từng bộ phận nhỏ của các hệ giải phẫu cơ thể người. Không những có thể quan sát kĩ hơn mà em còn có thể xem thêm các thông tin chi tiết về từng bộ phận.

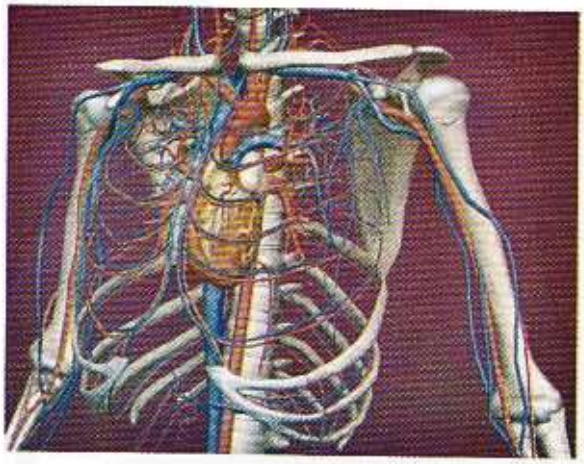
Nháy chuột vào bộ phận muốn quan sát, bộ phận này sẽ đổi màu. Muốn huỷ việc này cần nháy đúp chuột bên ngoài khu vực có mô phỏng.

Dưới đây là hình ảnh khi chọn phần đốt sống phía dưới để quan sát:



Hình 2.6. *Quan sát chi tiết xương sống thắt lưng*

Sau khi tìm hiểu xong một bộ phận nào đó, ta có thể ẩn bộ phận này khỏi mô hình. Chức năng này cho phép ta quan sát được các bộ phận bên trong bộ phận khác.



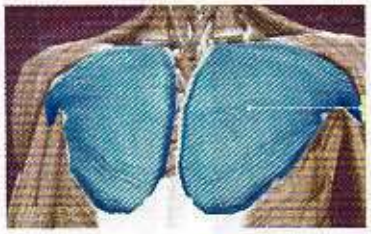
Hình 2.7. *Xương ức và bộ năm xương sườn phía trên được ẩn đi để có thể quan sát được các bộ phận bên trong lồng ngực như tim và các động mạch, tĩnh mạch chính từ tim.*

3 Hệ cơ

Nháy chuột vào biểu tượng có dòng chữ MUSCULAR SYSTEM ở màn hình LEARN để tìm hiểu hệ cơ của con người.

Chúng ta đã biết là cơ được cấu tạo bám vào xương có chức năng co, dãn để làm cho xương chuyển động.

Ví dụ một vài bộ phận quan trọng của hệ cơ:



Cơ ngực có chức năng làm cho lồng ngực có thể nở ra hoặc thu hẹp lại theo sự thở của con người

Cơ bắp tay phía trước - cơ hai đầu

Cơ bắp tay phía sau - cơ ba đầu

Cơ vai có nhiệm vụ nâng cánh tay khi chuyển động

Cơ đùi có bốn mảnh, bốn đầu là nhóm cơ khoẻ nhất của con người

Cơ mông có nhiệm vụ đỡ cho xương chậu và giúp xương đùi xoay được. Đây là bộ cơ lớn nhất về thể tích

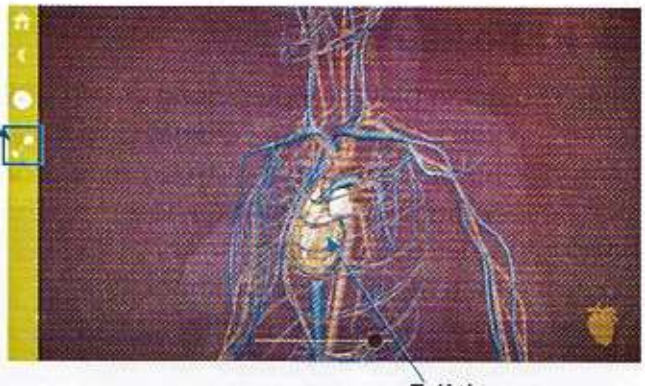
Hình 2.8. *Mô phỏng chi tiết một số bộ phận của hệ cơ*

Lưu ý hệ cơ gắn liền với xương, nên khi tìm hiểu hệ cơ, ta nên cho hiển thị cùng với hệ xương để quan sát được chính xác hơn.

4 Hệ tuần hoàn

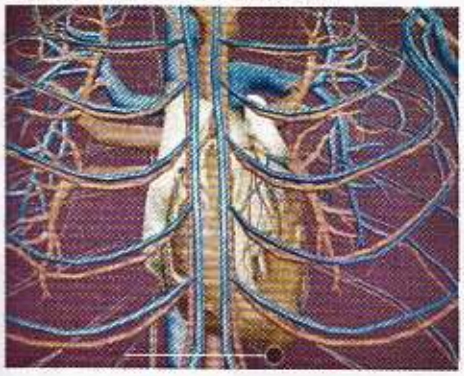
Nháy chuột vào biểu tượng có dòng chữ CIRCULATORY SYSTEM ở màn hình LEARN để tìm hiểu hệ tuần hoàn của con người.

Hệ tuần hoàn với cơ quan chính là trái tim giúp lưu thông máu đi khắp cơ thể để nuôi từng tế bào. Hình mô phỏng hệ tuần hoàn như sau:



Hình 2.9. *Mô phỏng hệ tuần hoàn*

Sử dụng các chức năng chính của phần mềm, em có thể tìm hiểu cấu tạo, hoạt động quả tim của người.

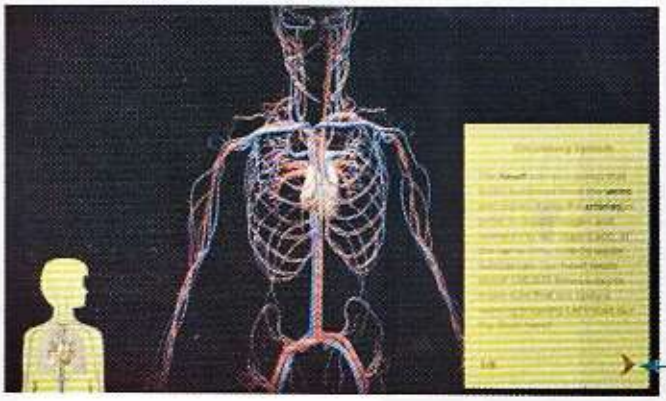


Hình 2.10. *Tim người được chia làm hai phần riêng biệt, vách ngăn trái và vách ngăn phải. Với mỗi bên lại chia làm hai ngăn trên và dưới. Do vậy có thể hình dung trái tim con người được chia làm bốn ngăn. Hai ngăn trên được gọi là tâm nhĩ (atria) trái, phải; hai ngăn dưới được gọi là tâm thất (ventricle) trái và phải.*

Chức năng mô phỏng hệ tuần hoàn

MÔ PHỎNG hoạt động một hệ giải phẫu là chức năng rất đặc biệt của phần mềm. Chức năng này sẽ đưa ra một bộ phim hoạt hình mô tả chi tiết toàn bộ hoạt động của vòng tuần hoàn trong cơ thể người. Để bắt đầu chức năng mô phỏng, nháy chuột vào nút mô phỏng, màn hình có dạng như sau:

Nút mô phỏng



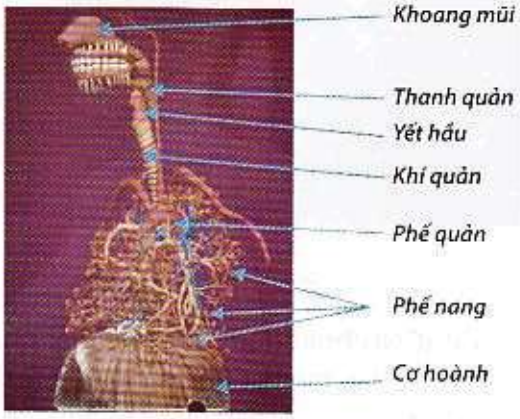
Hình 2.11. *Có tất cả tám màn hình mô phỏng hoạt động của hệ tuần hoàn*

5 Hệ hô hấp

Nháy chuột vào biểu tượng có dòng chữ RESPIRATORY SYSTEM trên màn hình LEARN để tìm hiểu hệ hô hấp. Hệ hô hấp có chức năng đặc biệt là làm giàu ô-xi trong máu thông qua trao đổi chất với bên ngoài, ví dụ hít thở không khí. Thông qua hít thở, hệ hô hấp sẽ lấy ô-xi trong không khí và đưa vào máu, sau đó lấy CO_2 trong máu để thải ra ngoài không khí.

Hình 2.12. *Phổi bao gồm màng phổi bên ngoài, hai lá phổi bên trong (phải, trái). Phổi được bảo vệ trong lồng ngực bởi hệ thống các xương sườn.*

Hệ hô hấp bao gồm khoang mũi, yết hầu, thanh quản, khí quản và các phế quản, phế nang trong phổi.



Hình 2.13. *Các bộ phận của hệ hô hấp*

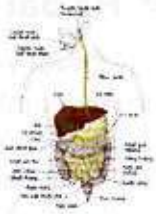
Chức năng mô phỏng hoạt động của hệ hô hấp

Khi nháy chuột vào nút mô phỏng, xuất hiện màn hình có dạng như sau :

Hình 2.14. *Hoạt động của hệ hô hấp, từ lúc hít thở không khí cho đến khi nạp thành công ô-xi vào máu và thải CO_2 ra ngoài.*

6 Hệ tiêu hoá

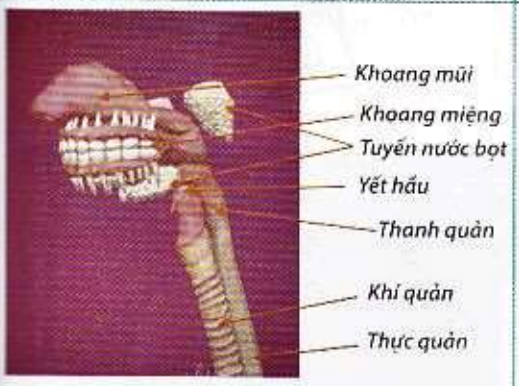
Nháy chuột vào biểu tượng có dòng chữ DIGESTIVE SYSTEM ở màn hình LEARN để tìm hiểu hệ tiêu hoá. Màn hình chính của hệ tiêu hoá có dạng như sau:

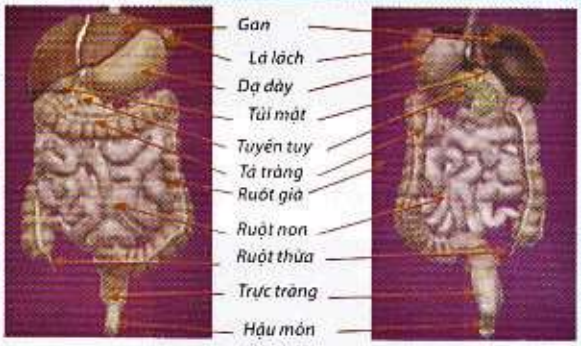


Hình 2.15. Mô phỏng hệ tiêu hoá.

Hệ tiêu hoá có chức năng chính là tiếp quản thức ăn từ miệng và tiêu hoá, hấp thụ, biến thức ăn thành năng lượng đi nuôi cơ thể. Các cơ quan thuộc hệ thống tiêu hoá chia thành hai phần: phần khoang miệng và phần khoang bụng, được nối với nhau bởi thực quản.

Các bộ phận ở phần khoang miệng	Các bộ phận ở phần khoang bụng

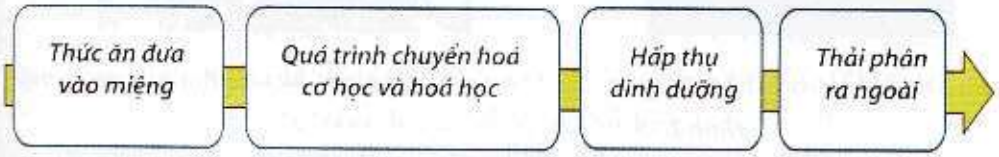




Hình 2.16

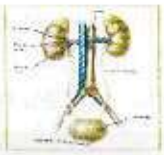
Em có thể nháy chuột vào nút mô phỏng để xem quá trình tiêu hoá của con người, bắt đầu từ thức ăn đưa vào miệng đến biến đổi, chuyển hoá, hấp thụ dinh dưỡng.

Có thể tóm tắt chức năng chính của hệ tiêu hoá của người như sau:



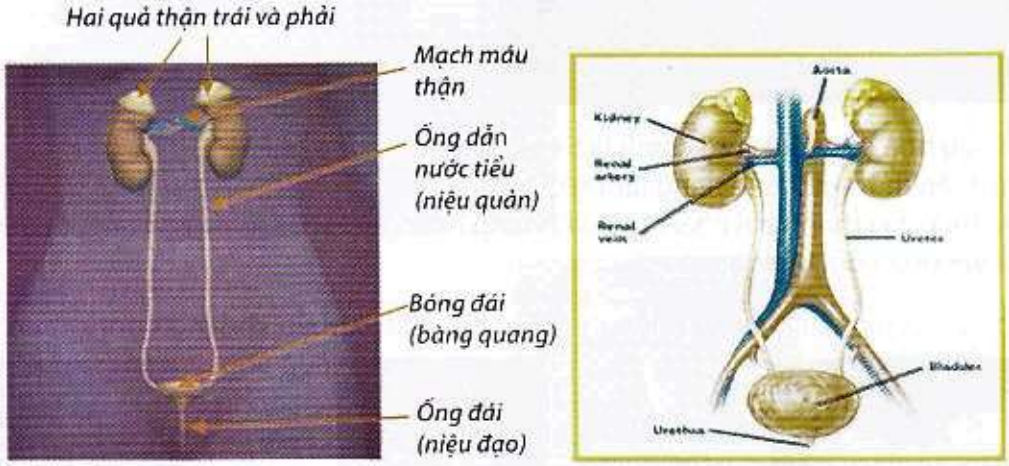
Hình 2.17

7 Hệ bài tiết



Nháy chuột vào biểu tượng có dòng chữ EXCRETOR SYSTEM trên màn hình LEARN để vào tìm hiểu hệ bài tiết. Hệ thống bài tiết có chức năng thải các chất độc ra bên ngoài cơ thể. Hệ thống này bao gồm thải khí CO_2 thông qua hít thở, thải mồ hôi qua da và thải nước tiểu qua thận.

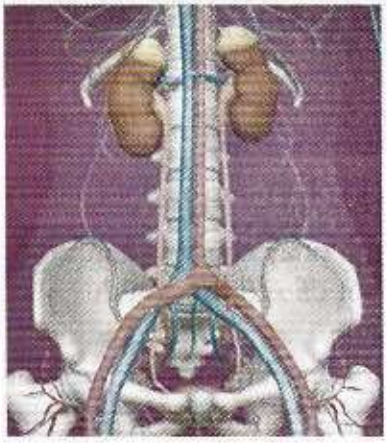
Hệ thống bài tiết nước tiểu bao gồm các cấu thành: hai quả thận, ống dẫn nước tiểu (niệu quản), bóng đái (bàng quang) và ống đái (niệu đạo). Hệ thống được mô tả trong phần mềm có hình ảnh sơ đồ như sau:



Hình 2.18. *Mô phỏng hệ bài tiết*

Để quan sát rõ hơn, em có thể hiển thị hệ bài tiết cùng với các hệ khác.

a) Hiển thị hệ bài tiết và hệ tuần hoàn



b) Hiển thị hệ bài tiết, hệ tuần hoàn và hệ xương

Hình 2.19. *Hệ bài tiết cùng với các hệ khác*

Phần mềm có chức năng mô phỏng hoạt động bài tiết lọc nước tiểu của thận.

8 Hệ thần kinh

Nháy chuột vào biểu tượng có dòng chữ NERVOUS SYSTEM trên màn hình LEARN để tìm hiểu về hệ thần kinh. Các bộ phận chính liên quan đến hệ thần kinh bao gồm não, tuỷ sống và các dây thần kinh.

Hình 2.20. *Mô phỏng hệ thần kinh*

Hệ thần kinh của con người được chia làm hai phần:

- **Hệ thần kinh trung ương** : gồm não và tuỷ sống.
- **Hệ thần kinh ngoại biên:** gồm các dây và mạch thần kinh toả đi khắp cơ thể.

Chức năng mô phỏng hoạt động của một phản xạ thần kinh không điều kiện

Nháy chuột vào nút mô phỏng, xuất hiện màn hình có dạng như sau:

Hình 2.21. *Mô phỏng hoạt động của phản xạ không điều kiện*

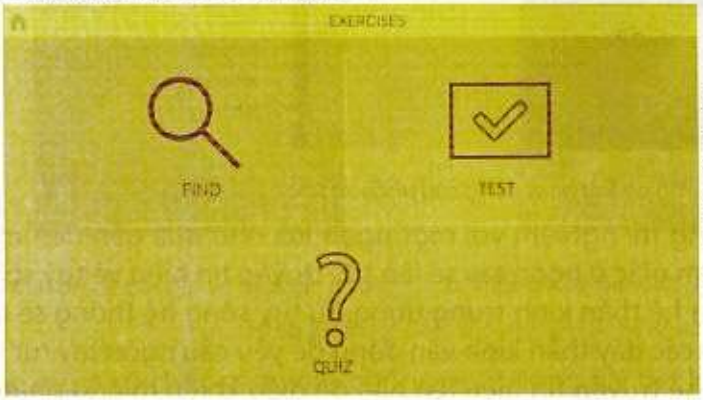
Chương trình mô phỏng thí nghiệm với một ngọn lửa nhỏ đưa gần đến một ngón tay. Thần kinh cảm giác ở ngón tay sẽ lập tức truyền tín hiệu về tuỷ sống là cơ quan đầu não của hệ thần kinh trung ương. Từ tuỷ sống hệ thống sẽ lập tức truyền tín hiệu cho các dây thần kinh vận động để yêu cầu ngón tay rụt lại. Đồng thời từ tuỷ sống sẽ truyền tín hiệu lên não để điều khiển miệng phát ra tiếng kêu "ối". Tất cả những việc đó được thực hiện gần như tức thời.

CÂU HỎI VÀ BÀI TẬP

1. Dựa trên các hoạt động mô phỏng của từng hệ thống của phần mềm, em hãy trình bày lại hoạt động của các hệ thống này:
   - Hệ tuần hoàn.
   - Hệ hô hấp.
   - Hệ tiêu hoá.
   - Hệ bài tiết.
   - Hệ thần kinh.
2. Trong hệ xương của con người, xương nào dài nhất, xương nào dài thứ hai?
3. Trong quả tim người có mấy cái van lớn? Các van này nằm ở bộ phận nào trong trái tim? Công dụng của các van này là gì?
4. Vì sao thức ăn qua đường miệng không bị chui vào khí quản?
5. Em hãy tra cứu từ điển để tìm tên tiếng Việt tương ứng cho các bộ phận sau của ruột già: ileum - cecum - ascending colon - traverse colon - descending colon - sigmoid colon rectum.
6. Thận đóng vai trò gì trong hệ bài tiết? Em hãy giải thích vì sao trong các hình vẽ mô tả chức năng của thận, các động mạnh đi vào được tô màu đỏ, tĩnh mạch đi ra màu xanh? Ngược lại với phổi, động mạnh đi vào được tô màu xanh, tĩnh mạch đi ra thì tô màu đỏ?
7. Trong cơ thể người, cơ nào là khoẻ nhất? Cơ nào dài nhất?

TÌM HIỂU MỞ RỘNG

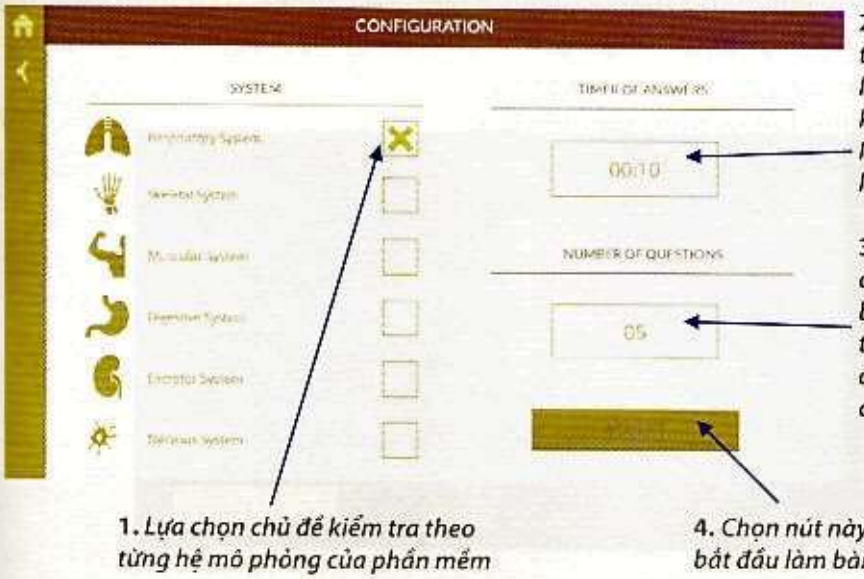
Ở màn hình chính của phần mềm, nháy chuột vào biểu tượng có chữ EXERCISES để vào chức năng kiểm tra kiến thức của phần mềm. Màn hình kiểm tra của phần mềm có dạng sau:



***Hình 2.22. Phần mềm có ba dạng bài kiểm tra:* Find, Quiz *và* Test.**

Các dạng bài này chỉ khác nhau ở cách đặt câu hỏi.

Nháy chuột chọn một trong biểu tượng trong màn hình kiểm tra, màn hình như sau xuất hiện để thực hiện các lựa chọn trước khi làm bài.



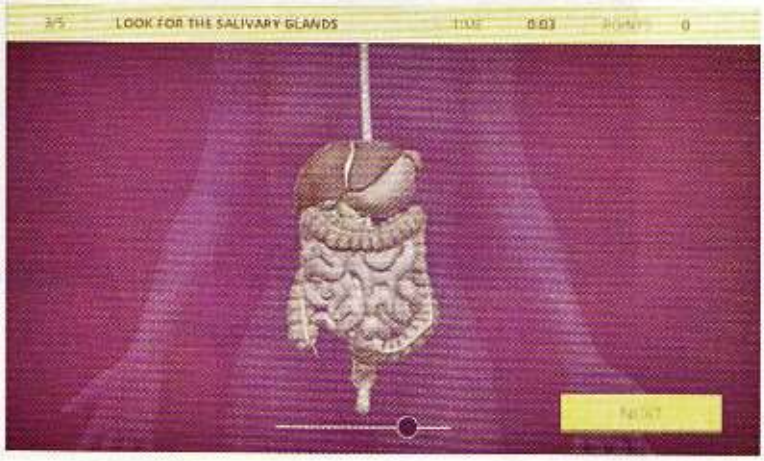
Hình 2.23

Khi làm xong phần mềm sẽ thông báo ngay kết quả trên màn hình và em có thể làm lại hoặc tiếp tục.

Các dạng câu hỏi kiểm tra của phần mềm

1. FIND: Tìm bộ phận theo tên

Đây là câu hỏi đơn giản nhất, có dạng **Look for <tên bộ phận>**. Người làm bài cần tìm trên hình ảnh (có thể xoay, dịch chuyển, phóng to, thu nhỏ) và nháy chuột vào vùng, bộ phận cần tìm.



Hình 2.24

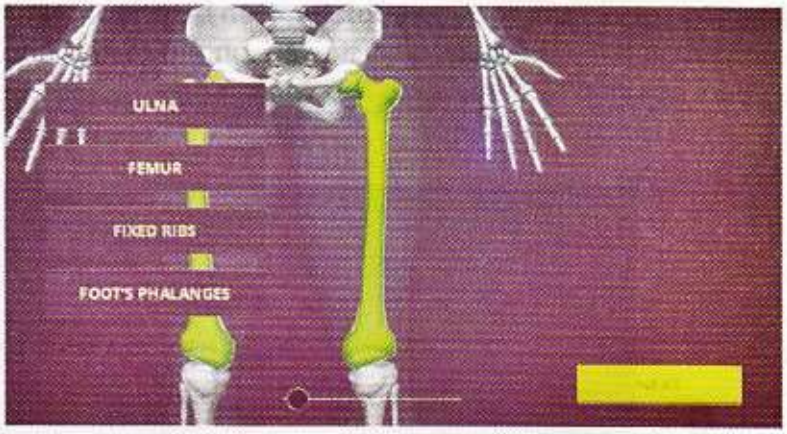
2. QUIZ: Tìm bộ phận theo chức năng

Đây là câu hỏi có dạng một câu hỏi ngắn, yêu cầu người dùng tìm một bộ phận theo một tính năng nào đó. Người làm bài cần tìm trên hình ảnh (có thể xoay, dịch chuyển, phóng to, thu nhỏ) và nháy chuột vào vùng, bộ phận cần tìm.

Hình 2.25

3. TEST: nhận dạng bộ phận đã đánh dấu trên màn hình

Câu hỏi dạng này như sau: trên màn hình xuất hiện một hình ảnh, trong đó có một bộ phận đã được đánh dấu. Có sẵn 4 đáp án, người làm bài cần chọn đáp án đúng.

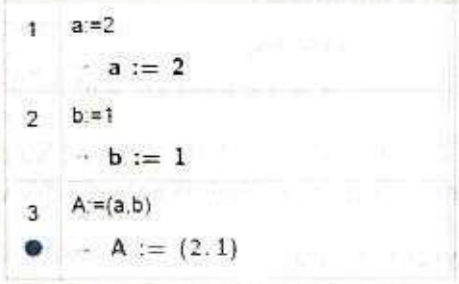


Hình 2.26

BÀI 11 GIẢI TOÁN VÀ VẼ HÌNH PHẲNG VỚI GEOGEBRA

- ☑ *Tính toán với đa thức, phân thức đại số, giải phương trình và bất phương trình bậc nhất một ẩn số với GeoGebra.*
- ☑ *Vẽ hình phẳng theo nội dung Hình học 8.*

1. Quan sát hình sau và trả lời các câu hỏi:



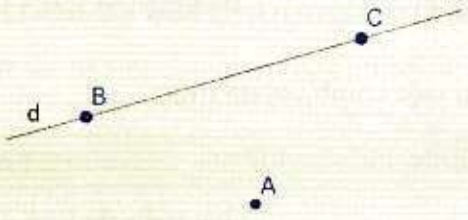
Hình 2.27

a) Trong hình 2.27 có bao nhiêu đối tượng toán học đã được tạo ra?

b) a, b là các đối tượng gì? Tự do hay phụ thuộc?

c) Điểm A là đối tượng tự do hay phụ thuộc?

2. Hình 2.28 có các đối tượng: các điểm A, B, C, đường thẳng d.



Hình 2.28

Mệnh đề nào dưới đây là đúng?

(A) A, B, C là các điểm tự do, d đi qua B, C nên không là tự do.

(B) d là tự do, B, C nằm trên d nên không là tự do. Điểm A là tự do.

(C) Đường thẳng d không là tự do. Các điểm A, B, C thì chưa thể kết luận là tự do hay phụ thuộc.

1 Các phép tính trên đa thức

Toàn bộ các tính toán với đa thức của mục này và các mục sau đều làm việc trên cửa sổ CAS và phải được thực hiện trong chế độ tính toán chính xác.

Nháy nút = để thiết lập chế độ tính toán chính xác.

– Các phép cộng, trừ, nhân đa thức: nhập biểu thức trên dòng lệnh của cửa sổ CAS, chúng ta có ngay kết quả.

1 `x^2*y - y*(x^2 + y^2) + (1/3x^2*y - x)*(1+x)`

$$\frac{1}{3}x^3y - y^3 + \frac{1}{3}x^2y - x^2 - x$$

Cần ghi đủ phép nhân giữa hai biến của đa thức khi viết lệnh.

Ví dụ: 2xy phải viết là 2x*y.

– Khai triển các biểu thức có chứa tích hoặc luỹ thừa: Sử dụng lệnh

Expand[<đa thức cần triển khai>]

1 `Expand[(a+b)^3]`

$$a^3 + 3a^2b + 3ab^2 + b^3$$

– Phân tích đa thức thành tích của các biểu thức: Sử dụng lệnh **Factor**[ ] cho việc phân tích trong số hữu tỉ và lệnh **iFactor**[ ] đối với số vô tỉ.

1 `Factor[x^3+2x^2*y+2x*y^2 + y^3]`

$$(x+y)(x^2+xy+y^2)$$

1 `IFactor[x^2-2]`

$$\left(x-\sqrt{2}\right)\left(x+\sqrt{2}\right)$$

– Các phép chia đa thức: Sử dụng ba lệnh là **Div** (tính thương), **Mod** (tính số dư) và **Division** (tính cả thương và số dư) của hai đa thức.

1 `Division[x^3+x^2-1, x-1]`

$$\{x^2+2x+2, 1\}$$

Phép chia $(x^3 + x^2 - 1) : (x - 1)$, thương là $x^2 + 2x + 2$, số dư là 1.

Bảng một số lệnh làm việc chính với đa thức:

Cú pháp lệnh	Ý nghĩa
Factor[<đa thức>]	Khai triển đa thức thành tích các thừa số trong phạm vi các số hữu tỉ.
iFactor[<đa thức>]	Khai triển đa thức thành tích các thừa số trong phạm vi các số vô tỉ.
Expand[<đa thức>]	Khai triển biểu thức tính toán đa thức.
Simplify[<đa thức>]	Rút gọn biểu thức tính của đa thức.

Cú pháp lệnh	Ý nghĩa
Div[<đa thức 1>, <đa thức 2>]	Cho thương của phép chia đa thức 1 cho đa thức 2.
Mod[<đa thức 1>, <đa thức 2>]	Cho số dư của phép chia đa thức 1 cho đa thức 2.
Division[<đa thức 1>, <đa thức 2>]	Cho thương và số dư của phép chia đa thức 1 cho đa thức 2.

2 Các phép tính trên phân thức đại số

Tương tự như với đa thức, chúng ta nhập trực tiếp phân thức cần tính toán trên dòng lệnh của cửa sổ CAS, chúng ta sẽ nhìn thấy ngay kết quả.

Ta gõ (x^2 + 2x + 1)/(x^2 – 1) để biểu thị $\frac{x^2+2x+1}{x^2-1}$.

Khi đó màn hình thể hiện như sau:

Dấu luỹ thừa được dùng với kí hiệu ^.

1 (x^2+2x+1)/(x^2-1)

$$\frac{x+1}{x-1}$$

Với các phép tính trên phân thức đại số, phần mềm sẽ tự động tính toán, khai triển và rút gọn nếu có thể được, chẳng hạn:

Phải thêm dấu ngoặc đơn đối với tử và mẫu là đa thức khi viết lệnh.

2 (a-(x^2+a^2)/(x+a))*(2a/x-4a/(x-a))

$$2a$$

3 Giải phương trình và bất phương trình bậc nhất một ẩn

Để giải phương trình và bất phương trình, chúng ta sẽ sử dụng các lệnh Solve[] và Solutions[]. Cú pháp và ý nghĩa các lệnh này như sau:

– Lệnh **Solve[<phương trình x>]** hoặc **Solve[<bất phương trình x>]** cho kết quả là các nghiệm của phương trình hoặc bất phương trình.

– Lệnh **Solutions[<phương trình x>]** hoặc **Solutions[< bất phương trình x>]** cho kết quả là tất cả các giá trị nghiệm của phương trình, bất phương trình.

1 Solve[1/3*(x-1)+2x = 0]

$$\left\{x = \frac{1}{7}\right\}$$

2 Solutions[3x + 12 = 5]

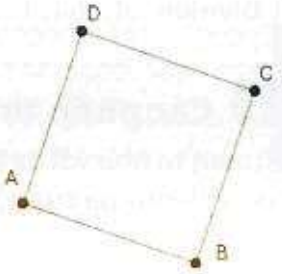
$$\left\{-\frac{7}{3}\right\}$$

Các lệnh Solve và Solutions cũng dùng để giải các phương trình, bất phương trình bậc cao và có nhiều ẩn số.

1	Solve[-3/2(1-x)+3x < 1]	2	Solutions[(x-1)/2 + 3/4x > 1/3]
	$\left\{\frac{5}{9} > x\right\}$		$\left\{x > \frac{2}{3}\right\}$

4 Quan hệ toán học và các công cụ tạo quan hệ toán học trong GeoGebra

Chúng ta đã biết rằng mỗi tệp GeoGebra sẽ có rất nhiều đối tượng toán học. Các đối tượng này được chia làm hai loại: tự do và phụ thuộc. Quan hệ phụ thuộc ở đây hiểu là các phụ thuộc toán học và là phụ thuộc một chiều. Xét ví dụ sau:



Hình 2.29

Hình vuông ABCD được xây dựng từ hai điểm tự do A, B ban đầu. Các điểm C, D không tự do và phụ thuộc vào A, B. Có thể dùng chuột di chuyển các điểm A, B trên mặt phẳng thì các điểm C, D sẽ phải chuyển động theo để đảm bảo ABCD luôn là hình vuông.

Trong ví dụ trên, quan hệ nào là phụ thuộc toán học?

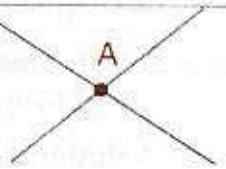
(A) song song (B) vuông góc

(C) giao nhau (D) đi qua.

Tất cả các công cụ (đại số và hình học) của GeoGebra đều có chức năng chính là thiết lập các đối tượng toán học thông qua các quan hệ toán học. Chúng ta xét một số công cụ tạo quan hệ phụ thuộc sau:

a) Công cụ tạo điểm

Chọn công cụ  để tạo ra các điểm tự do và điểm phụ thuộc. Cụ thể tạo đối tượng điểm như sau:

Tạo điểm A	Tạo điểm A nằm trên một đường (đoạn, tia)	Tạo điểm A là giao điểm
		
Nháy chuột lên một vị trí trống của màn hình để tạo ra một điểm tự do.	Nháy chuột lên một đường thẳng (đoạn thẳng, tia) sẽ tạo ra một điểm luôn nằm trên đường thẳng (đoạn thẳng, tia) này. Điểm này phụ thuộc.	Nháy chuột tại vị trí giao điểm.

b) Công cụ đoạn thẳng , đường thẳng , tia

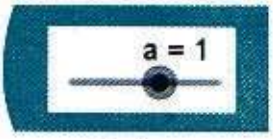
Các công cụ này đều có chung đặc điểm là phải đi qua hai điểm cho trước, thao tác là nháy chuột lần lượt lên hai điểm này.

c) Công cụ vẽ các đường song song, vuông góc, phân giác, trung trực

Công cụ	Tên và chức năng	Thao tác
	Tạo đường song song	– Chọn công cụ. – Chọn điểm, sau đó chọn đường thẳng (đoạn thẳng, tia) muốn vẽ song song hoặc chọn đường thẳng (đoạn thẳng, tia) trước, chọn điểm sau.
	Tạo đường vuông góc	– Chọn công cụ. – Chọn điểm, sau đó là đường thẳng (đoạn thẳng, tia) mốn vẽ vuông góc hoặc chọn đường thẳng (đoạn thẳng, tia) trước, chọn điểm sau.
	Tạo đường phân giác	*Cách 1: Tạo một đường phân giác* Chọn công cụ, sau đó chọn lần lượt ba điểm, sẽ tạo ra một đường phân giác của góc tạo bởi ba điểm trên. *Cách 2: Tạo hai đường phân giác* Chọn công cụ, chọn hai đường thẳng (đoạn thẳng, tia) để tạo hai đường phân giác của góc tạo bởi hai đường thẳng (đoạn thẳng, tia) này.
	Tạo đường trung trực	– Chọn công cụ. – Chọn đoạn thẳng hoặc hai điểm để tạo ra đường trung trực của đoạn thẳng này.

d) Tạo đối tượng số trực tiếp từ dòng nhập lệnh

Ta đã biết cách tạo ra một đối tượng số tự do từ ngay dòng lệnh của phần mềm GeoGebra. Ví dụ có thể nhập vào dòng lệnh như sau:



Hình 2.30

a := 1

Phần mềm sẽ tạo ngay một đối tượng số tự do có tên là a, giá trị bằng 1. Bây giờ chúng ta có thể tạo ra các đối tượng số khác phụ thuộc vào a, ví dụ:

b := a/2; c := a^2.

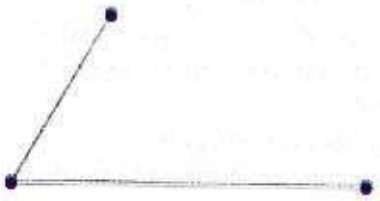
5 Các công cụ biến đổi hình học trong GeoGebra

Trong mục này chúng ta sẽ làm quen với hai công cụ thực hiện phép biến đổi hình học của GeoGebra là lấy đối xứng trục và đối xứng tâm.

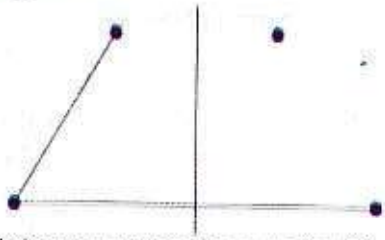
Công cụ	Chức năng	Thao tác
	Tạo ra đối tượng mới là đối xứng của một đối tượng cho trước, qua một trục cho trước.	– Chọn công cụ. – Chọn các đối tượng cần tạo đối xứng (có thể chọn một hoặc nhiều). – Nháy chuột chọn trục đối xứng (đoạn thẳng, đường thẳng, tia).
	Tạo ra đối tượng mới là đối xứng của một đối tượng cho trước, qua một tâm cho trước.	– Chọn công cụ. – Chọn các đối tượng cần tạo đối xứng (có thể chọn một hoặc nhiều). – Nháy chuột chọn điểm là tâm trục đối xứng.

Một số ví dụ cụ thể ứng dụng các công cụ này.

a) Vẽ hình thang cân biết cạnh đáy và một cạnh bên



1. Vị trí ban đầu, biết một cạnh đáy và một cạnh bên



2. Kẻ đường trung trực của cạnh đáy. Lấy đường này làm trục đối xứng, tạo điểm là đối xứng với đỉnh của cạnh bên. Sau đó ẩn đường trung trực và nối các đỉnh để tạo ra hình thang cân

Hình 2.31. *Vẽ hình thang cân*

b) Vẽ hình bình hành biết một cạnh và tâm

1. Vị trí ban đầu, biết một cạnh và tâm hình bình hành (điểm có màu đỏ)

2. Lấy điểm màu đỏ làm tâm đối xứng. Vẽ hai điểm đối xứng với hai đỉnh ban đầu qua tâm màu đỏ. Sau đó nối lại các cạnh ta thu được hình bình hành

Hình 2.32. *Vẽ hình bình hành*

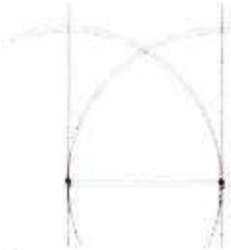
6 Công cụ đường tròn và cách vẽ một số hình đặc biệt

Công cụ	Chức năng	Thao tác
	Vẽ đường tròn biết tâm và một điểm trên đường tròn.	– Nháy chuột chọn một điểm làm tâm. – Nháy chuột chọn một điểm nằm trên đường tròn.
	Vẽ đường tròn biết tâm và độ dài bán kính.	– Nháy chuột chọn một điểm làm tâm. – Xuất hiện cửa sổ nhập bán kính, có thể nhập một số dương bất kì hoặc nhập tên của đối tượng số hoặc tên của đoạn thẳng.

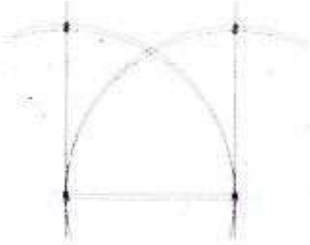
a) Vẽ hình vuông biết một cạnh (không dùng công cụ đa giác đều)

1. Vẽ đoạn thẳng là cạnh cho trước của hình vuông

2. Sử dụng công cụ đường vuông góc để vẽ hai đường thẳng vuông góc với đoạn thẳng đã cho và đi qua hai điểm đầu mút của đoạn thẳng này.



3. Sử dụng công cụ  *lần lượt lấy hai điểm đầu mút của đoạn thẳng làm tâm vẽ hai đường tròn đi qua đỉnh còn lại.*



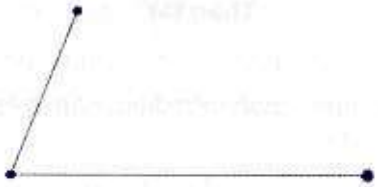
4. Sử dụng công cụ điểm để xác định giao điểm của hai đường tròn này với hai đường thẳng vuông góc đã vẽ tại bước 2.

5. Bây giờ ẩn đi hai đường tròn và hai đường vuông góc.

6. Sử dụng công cụ đoạn thẳng nối các cạnh còn thiếu để tạo thành hình vuông.

Hình 2.33. *Vẽ hình vuông*

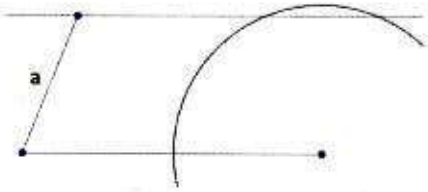
b) Vẽ hình thang cân biết trước một cạnh đáy và một cạnh bên



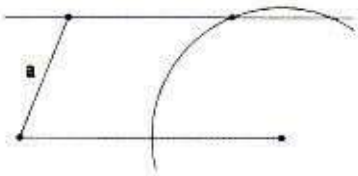
1. Cho trước hai cạnh của hình thang với ba đỉnh tự do.

2. Từ một đỉnh tự do của cạnh bên, sử dụng công cụ  *kẻ đường thẳng song song với cạnh đáy hình thang.*

3. Kiểm tra lại tên của cạnh bên, chẳng hạn là a. Khi đó a sẽ là độ dài cạnh bên.



4. Dùng công cụ đường tròn  *, vẽ đường tròn có tâm là đỉnh thuộc đáy chưa xác định và bán kính là a để xác định cạnh bên còn lại.*



5. Sử dụng công cụ 

để xác định giao điểm (bên trong) của đường tròn này với đường thẳng đã vẽ trong bước 4.

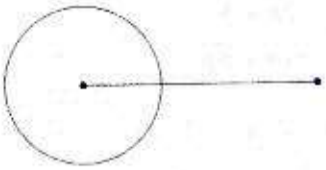
6. Ẩn đi các đối tượng phụ là đường thẳng song song và đường tròn.

7. Sử dụng công cụ đoạn thẳng nối các cạnh còn thiếu để tạo thành hình thang cân.

8. Ẩn tên cạnh bên a, ta được hình thang cân cần vẽ.

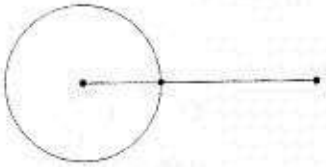
Hình 2.34. *Vẽ hình thang cân*

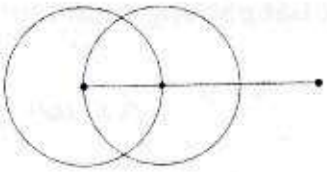
c) Chia ba một đoạn thẳng



1. Vẽ đoạn thẳng đi qua hai điểm tự do A, B. Xem trên cửa sổ Danh sách đối tượng để biết tên đoạn thẳng, chẳng hạn là d.

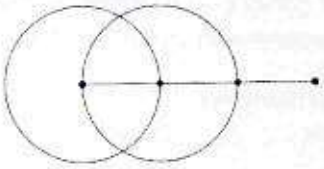
2. Sử dụng công cụ để vẽ đường tròn với tâm là điểm bên trái đoạn thẳng. Khi xuất hiện hộp thoại nhập bán kính, gõ d/3 và nhấn phím Enter.





3. Dùng công cụ điểm xác định giao điểm của đường tròn này với đoạn thẳng AB.

4. Sử dụng công cụ để vẽ đường tròn với tâm là giao điểm vừa xác định trong bước 3. Khi xuất hiện hộp thoại nhập bán kính, gõ d/3 và nhấn phím Enter.



5. Lại dùng công cụ điểm xác định giao điểm của đường tròn này với đoạn thẳng AB.

6. Bây giờ ẩn đi hai đường tròn, ta chia đoạn thẳng AB thành ba phần bằng nhau.

Hình 2.35. *Chia ba đoạn thẳng*

CÂU HỎI VÀ BÀI TẬP

1. Tính

a) $1^5 + 2^5 + 3^5 + \ldots + 10^5$

b) $(x-y)(x^3+xy+y^3)$

2. Phân tích các đa thức sau thành nhân tử

a) $x^3y^2 + x^2y^3 + x^2y + xy^2 + x^3 + y^3 + x + y$

b) $x^3 + 2x^2y + xy^2 - 9x$

3. Tính

a) $\frac{x+1}{2x+6}+\frac{2x+3}{x(x+3)}$

b) $\frac{x-1}{x}.\left(x^2+x+1+\frac{x^3}{x-1}\right)$

c) $\left(\frac{x^2}{y^2}+\frac{y}{x}\right):\left(\frac{x}{y^2}-\frac{1}{y}+\frac{1}{x}\right)$

4. Giải các phương trình sau:

a) $(x+1)^2=4(x^2-2x+1)$

b) $\frac{1}{2x-3}+\frac{3}{x(2x-3)}=\frac{5}{x}$

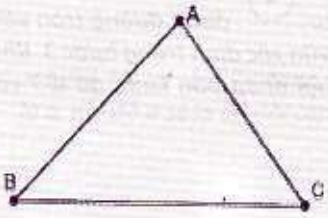
5. Giải các bất phương trình sau:

a) $\frac{2x+3}{-2}\geq\frac{4-x}{-3}$

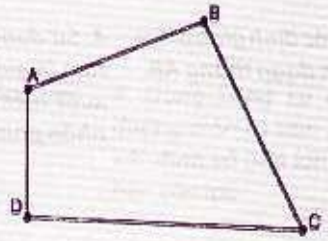
b) $(x-3)^2<x^2-3$

c) $\frac{3}{-x}+\frac{10}{x}<1$

.am giác, tứ giác



Dùng công cụ đoạn thẳng nối các cạnh của tam giác

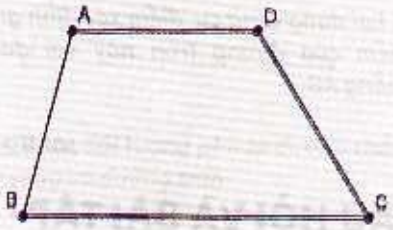


Dùng công cụ đoạn thẳng nối các cạnh của tứ giác

Hình 2.36

7. Vẽ hình thang

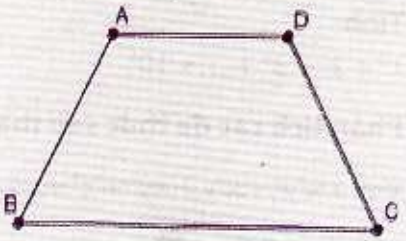
Cho ba đỉnh A, B, C. Dựng đỉnh D của hình thang ABCD dựa trên các công cụ đoạn thẳng và đường song song.



Hình 2.37

8. Vẽ hình thang cân

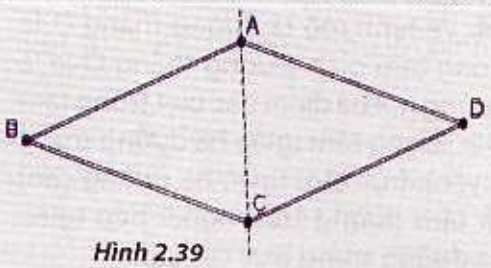
Cho ba đỉnh A, B, C. Dựng đỉnh D của hình thang cân ABCD dựa trên các công cụ đoạn thẳng, đường trung trực và phép biến đổi đối xứng qua trục



Hình 2.38

9. Vẽ hình thoi

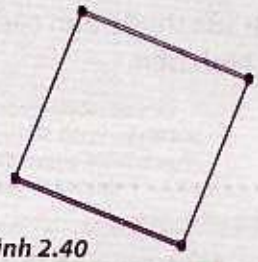
Cho trước cạnh AB và một đường thẳng đi qua A. Hãy vẽ hình thoi ABCD lấy đường thẳng đã cho là đường chéo. Sử dụng các công cụ thích hợp đã học để dựng các đỉnh C, D của hình thoi.



Hình 2.39

10. Vẽ hình vuông

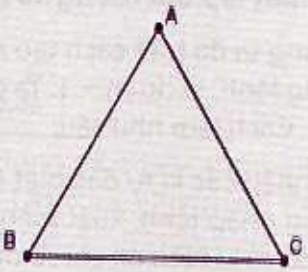
Sử dụng các công cụ thích hợp để vẽ một hình vuông nếu biết trước một cạnh.



Hình 2.40

11. Vẽ tam giác đều

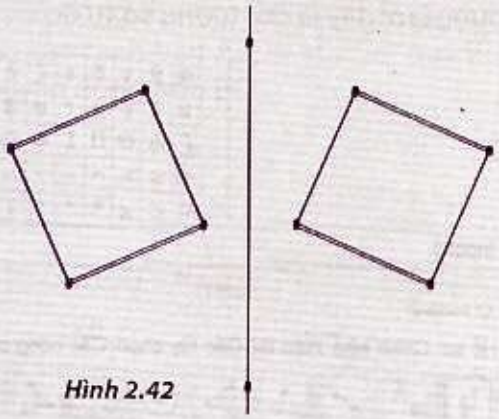
Cho trước cạnh BC, hãy vẽ tam giác đều ABC.



Hình 2.41

12. Vẽ một hình là đối xứng trục của một đối tượng cho trước trên màn hình

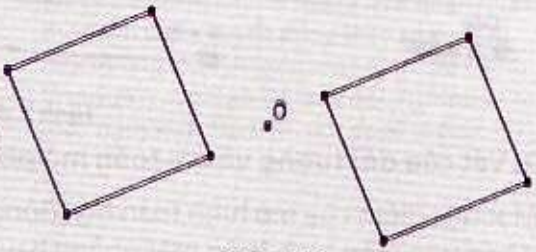
Cho một hình và một đường thẳng trên mặt phẳng. Hãy dựng hình mới là đối xứng của hình đã cho qua trục đường thẳng trên. Sử dụng công cụ đối xứng trục để vẽ hình.



Hình 2.42

13. Vẽ một hình là đối xứng qua tâm của một đối tượng cho trước

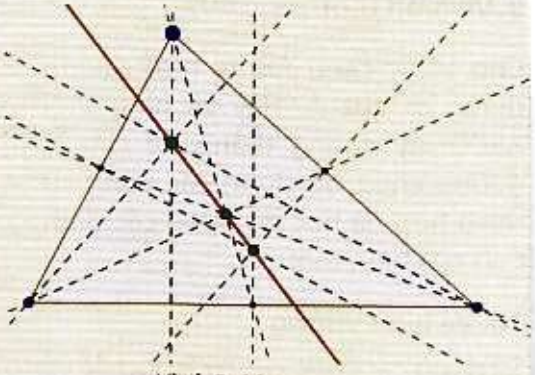
Cho trước một hình và một điểm O. Hãy dựng hình mới là đối xứng qua tâm O của hình đã cho. Sử dụng công cụ đối xứng tâm để vẽ hình.



Hình 2.43

14. Vẽ hình mô tả đường thẳng Ơ-le trong tam giác. Đường thẳng Ơ-le là đường nối ba điểm đặc biệt trong tam giác: trọng tâm (giao ba đường trung tuyến), trực tâm (giao ba đường cao) và tâm đường tròn ngoại tiếp (giao ba đường trung trực các cạnh).

Vẽ và chỉnh sửa thuộc tính các đường như trong hình bên.



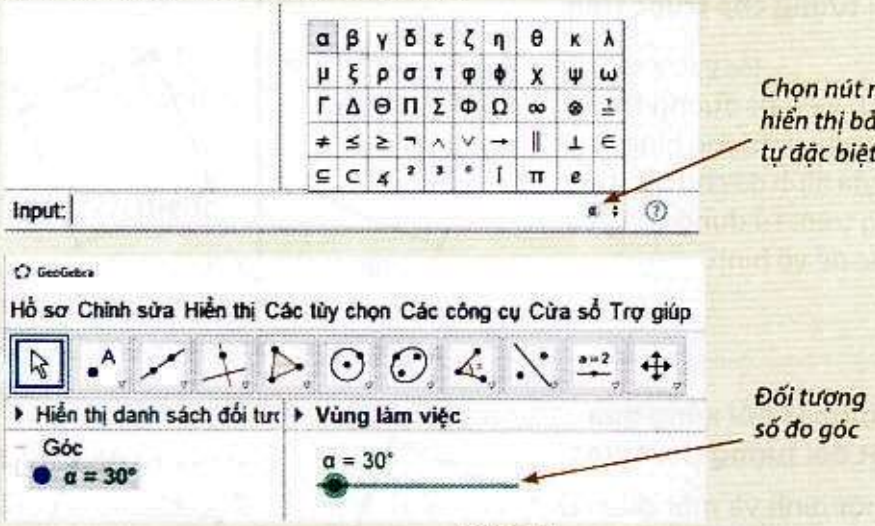
Hình 2.44

TÌM HIỂU MỞ RỘNG

1. Thiết lập đối tượng số là số đo góc

Chúng ta đã biết cách tạo một đối tượng tự do là số bằng cách gõ trực tiếp từ dòng lệnh, ví dụ m:= 1. Ta có thể làm tương tự để tạo đối tượng tự do là số đo góc. Cách làm như sau:

Để nhập các kí tự đặc biệt trên dòng lệnh ta nháy chuột vào nút nhỏ α ở cuối dòng nhập lệnh, xuất hiện một bảng cho phép chọn các kí tự đặc biệt, trong đó có cả kí hiệu để chỉ số đo góc, ví dụ $\alpha := 30°$. Một đối tượng là số đo góc α được tạo, đây là đối tượng số tự do.



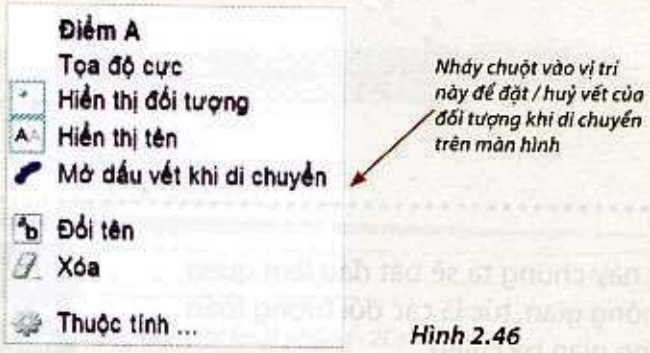
Hình 2.45

2. Vết của đối tượng và bài toán mô phỏng quỹ tích

Một đặc điểm của mô hình toán học động là các đối tượng có thể được chuyển động trên màn hình (trên mặt phẳng hay trong không gian). Ví dụ với các điểm

tự do, chúng ta có thể điều khiển các điểm này chuyển động trên màn hình, khi đó các đối tượng khác có quan hệ phụ thuộc sẽ chuyển động theo. Đó chính là ý nghĩa thực của khái niệm "toán học động".

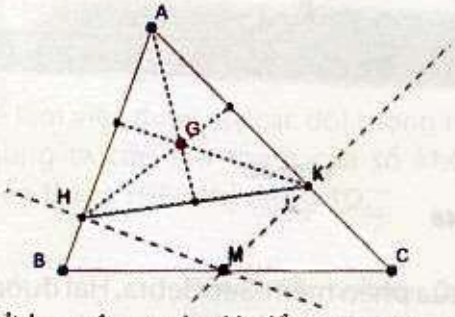
Trong GeoGebra có một tính năng cho phép khi các đối tượng chuyển động sẽ để lại **vết** trên màn hình, hay nói cách khác các đối tượng này sẽ có tác dụng như cái bút vẽ trên màn hình. Để thiết lập chế độ hiển thị vết của một đối tượng nháy nút phải chuột lên đối tượng đó và chọn lệnh **Mở dấu vết khi di chuyển**. Chọn lệnh này lần thứ hai sẽ kết thúc chế độ hiển thị vết.



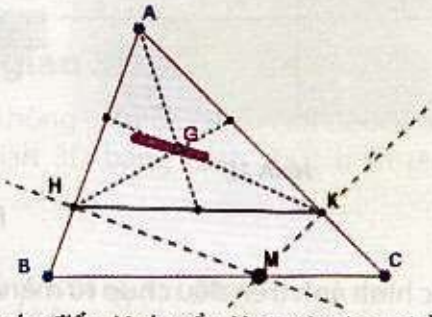
Hình 2.46

Chức năng hiển thị vết này của GeoGebra được áp dụng rất nhiều trong các bài toán thực tế, ví dụ bài toán tìm quỹ tích. Ta có thể bật chức năng hiển thị vết của một đối tượng để quan sát và dự đoán quỹ tích của đối tượng này.

Ví dụ: Cho tam giác ABC. Điểm M chuyển động theo cạnh BC. Từ M lần lượt kẻ các đường vuông góc MH, MK xuống các cạnh bên AB và AC. Tìm quỹ tích trọng tâm G của tam giác AHK khi M chuyển động trên BC.



Sử dụng công cụ đa giác để vẽ tam giác ABC. Dùng công cụ điểm để tạo một điểm M thuộc BC (chuyển động tự do trên đoạn này). Từ M kẻ các đường vuông góc xuống AB, AC. Dùng các công cụ trung điểm và đoạn thẳng vẽ ba đường trung tuyến tam giác AHK. Đặt vết cho điểm G.



Khi cho điểm M chuyển động trên đoạn thẳng BC, điểm G sẽ chuyển động theo và vạch ra một đường màu đỏ trên màn hình. Đó chính là quỹ tích của G khi M thay đổi. Từ hình vẽ ta thấy ngay quỹ tích G là một đoạn thẳng.

Hình 2.47

BÀI 12 VẼ HÌNH KHÔNG GIAN VỚI GEOGEBRA

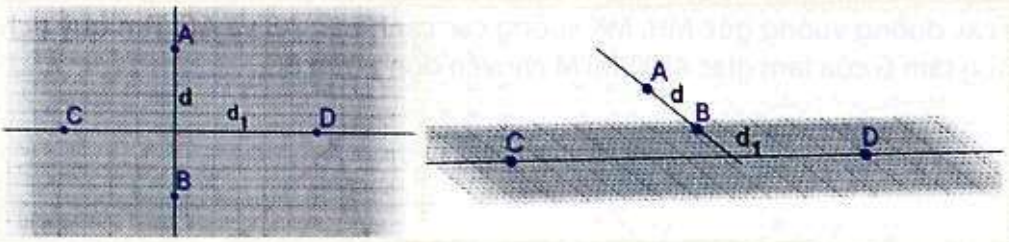
- ☑ ***Làm quen với mô hình không gian ba chiều của GeoGebra.***
- ☑ ***Vẽ hình hộp, hình lăng trụ, hình chóp với GeoGebra.***

Trong bài học này chúng ta sẽ bắt đầu làm quen với các hình không gian, tức là các đối tượng toán học trong không gian ba chiều.

Mặt phẳng là hai chiều hay 2D, không gian là ba chiều hay 3D.

Hãy quan sát hình 2D và hình 3D sau :

Có hai đường thẳng d và d_1. Các điểm A, B nằm trên d ; điểm C, D nằm trên d_1.

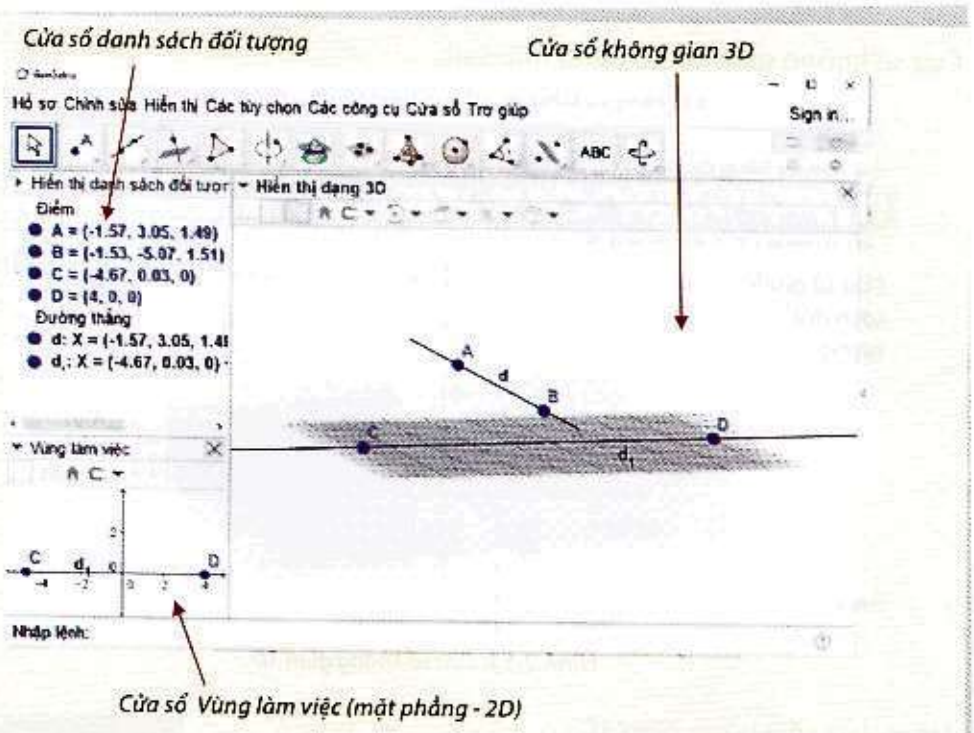


Hình 2D *Hình 3D*

Hình 2.48

Các hình ảnh trên đều chụp từ màn hình của phần mềm GeoGebra. Hai đường thẳng d và d_1 ở hình 2D giao nhau nhưng ở hình 3D là không giao nhau. Các điểm A, B, C, D ở hình 2D nằm trên cùng mặt phẳng, còn ở hình 3D các điểm này không cùng trên mặt phẳng. Như vậy chúng ta đã thấy và hình dung được cách mà phần mềm thể hiện các hình trong không gian.

Hãy quan sát hình sau và nêu nhận xét của mình, chú ý tập trung quan sát các đối tượng hình học trên màn hình.

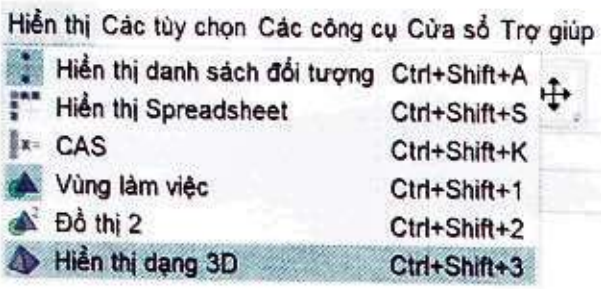


Hình 2.49

Trên hình này, trong cửa sổ không gian 3D, chúng ta thấy rõ hai đường thẳng d và d_1 không cùng nằm trên một mặt phẳng, không cắt nhau. Hai đường thẳng này chéo nhau.

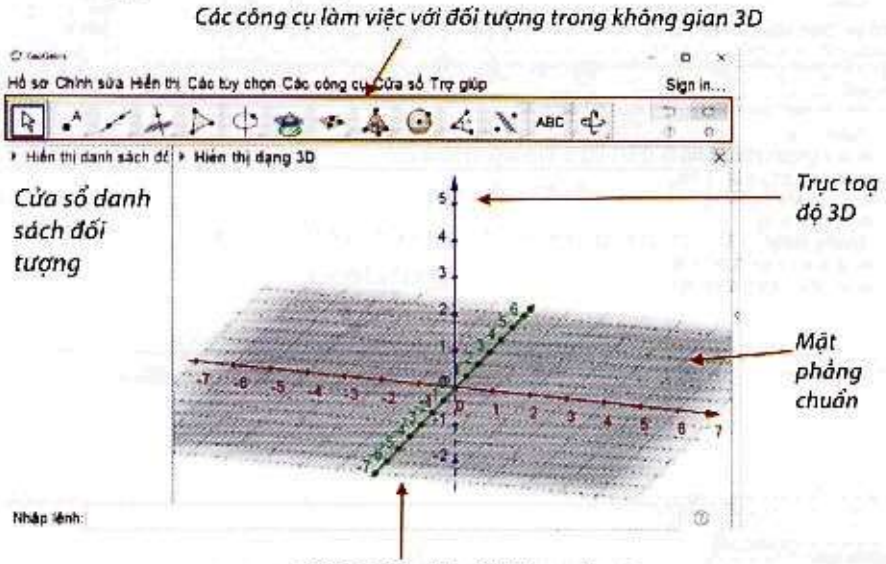
1 Làm quen với cửa sổ không gian 3D

Để làm việc được với các đối tượng trong không gian ba chiều của GeoGebra, chúng ta cần mở thêm cửa sổ không gian 3D bằng cách thực hiện lệnh **Hiển thị → Hiển thị dạng 3D**.



Hình 2.50

Cửa sổ không gian 3D có dạng như sau:



Hình 2.51. *Cửa sổ không gian 3D*

Trong cửa sổ không gian 3D có:

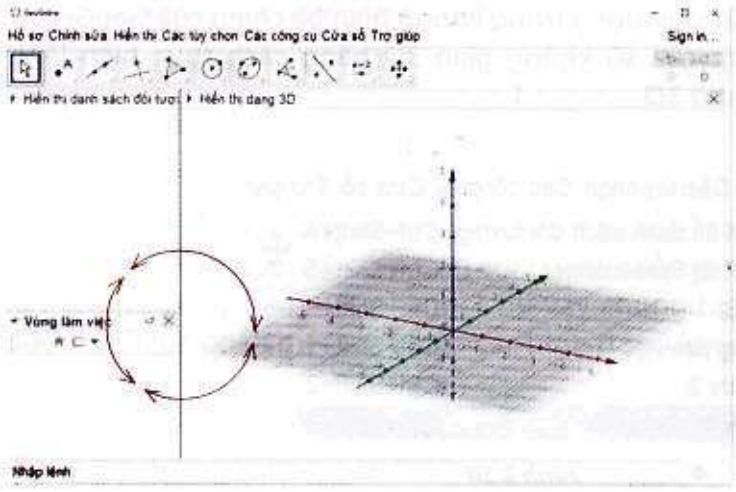
Khi làm việc với hình không gian 3D nên đóng cửa sổ CAS

- Các công cụ làm việc với không gian 3D.

- Hệ trục toạ độ tương ứng với ba trục toạ độ x, y, z trong không gian 3D.

- Mặt phẳng chuẩn luôn hiện chính giữa màn hình làm việc. Ta chỉ có thể làm việc được với các đối tượng xuất phát từ mặt phẳng chuẩn này.

Em có thể thiết lập chế độ quan sát hiển thị cả ba cửa sổ: Danh sách đối tượng, vùng làm việc (2D) và không gian 3D như sau:



Hình 2.52. *Các đối tượng trên ba cửa sổ này có quan hệ toán học chặt chẽ với nhau*

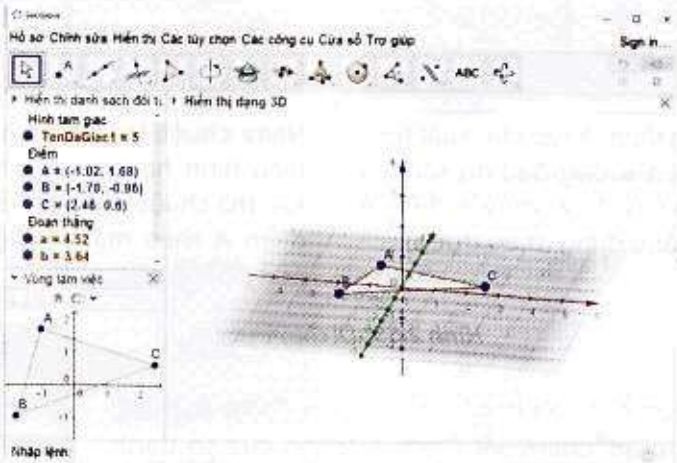
Đưa cửa sổ mặt phẳng lồng vào giao diện làm việc với không gian 3D.

Hãy thực hiện thao tác sau:

- Nháy chuột lên *Vùng làm việc* để kích hoạt cửa sổ này.

- Chọn công cụ  , vẽ một tam giác ABC trên *Vùng làm việc*.

Em sẽ thấy trong cửa sổ không gian 3D, tam giác ABC sẽ xuất hiện trên mặt phẳng chuẩn như sau:



***Hình 2.53.** Tam giác ABC sẽ hiện chính xác trên mặt phẳng chuẩn của không gian 3D*

Như vậy mặt phẳng chuẩn của màn hình không gian 3D chính là *Vùng làm việc* trên mặt phẳng mà ta đã biết.

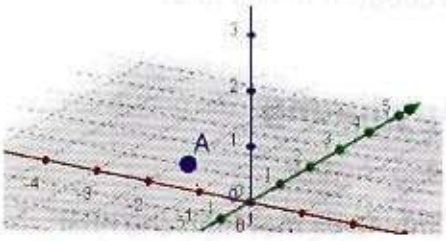
2 Điểm và di chuyển điểm trong không gian

a) Tạo đối tượng điểm

- Kích hoạt cửa sổ không gian 3D, chọn công cụ  .

- Nháy chuột lên vị trí bất kì trên mặt phẳng chuẩn.

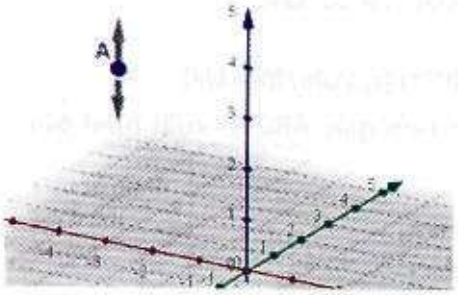
Nháy chuột vào cửa sổ không gian 3D để kích hoạt.



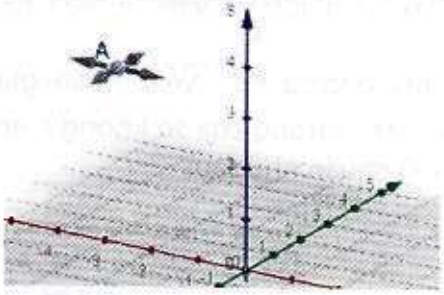
***Hình 2.54.** Điểm A xuất hiện trên mặt phẳng chuẩn*

Hãy quan sát điểm A trong cửa sổ danh sách đối tượng, em có nhận xét gì?

b) Di chuyển điểm trong không gian



Nháy chuột lên điểm A sao cho xuất hiện hình mũi tên lên xuống. Sau đó kéo thả chuột tại điểm này để di chuyển điểm A theo hướng thẳng đứng (theo trục z).



Nháy chuột lên điểm A sao cho xuất hiện hình hai mũi tên ngang. Sau đó kéo thả chuột tại điểm này để di chuyển điểm A theo mặt phẳng ngang (theo mặt phẳng x-y).

Hình 2.55. *Di chuyển điểm*

Hãy thực hiện thao tác di chuyển điểm A trong không gian. Quan sát điểm A trong cửa sổ danh sách đối tượng, em có nhận xét gì?

Điểm A có thể di chuyển theo hai cách: theo chiều thẳng đứng và theo chiều mặt phẳng ngang.

3 Xoay hình trong không gian

Trong không gian 3D, một trong những thao tác quan trọng nhất là "xoay" hình, hay nói cách khác là thay đổi góc và hướng nhìn vào các hình. Thao tác này thực hiện theo các cách sau:

Cách 1: Nhấn giữ nút phải chuột và đồng thời rê chuột.

Cách 2: Chuyển về chế độ chọn, kéo thả chuột trên màn hình.

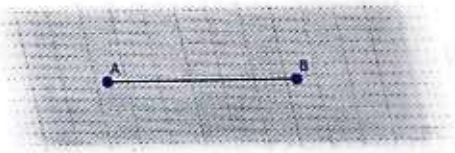
Cách 3: Chọn công cụ quay , rồi kéo thả chuột trên màn hình.

Chọn nút lệnh hoặc nhấn phím ESC để chuyển về chế độ chọn.

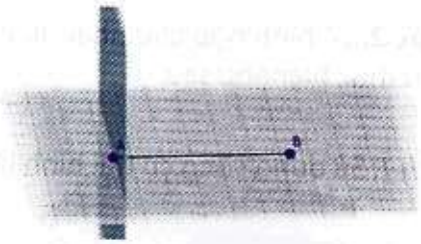
4 Vẽ hình hộp chữ nhật, hình lập phương

a) Vẽ hình hộp chữ nhật (đáy nằm trên mặt phẳng chuẩn)

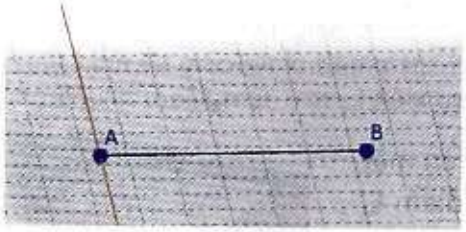
Bước 1: Trong mặt phẳng chuẩn cho trước cạnh AB vẽ một hình chữ nhật



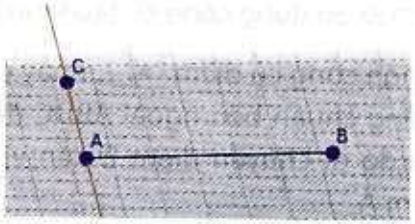
1. Sử dụng công cụ đoạn thẳng  vẽ một đoạn thẳng AB đi qua hai điểm tự do A và B.



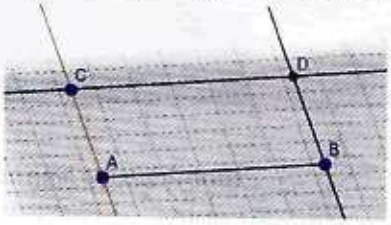
2. Thiết lập một mặt phẳng qua điểm A và vuông góc với AB. Sử dụng công cụ  nháy chuột vào điểm A, sau đó nháy chuột lên đoạn AB.



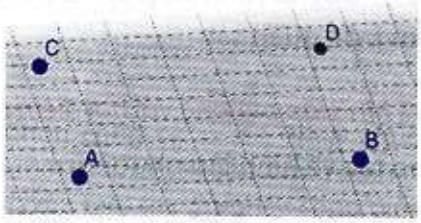
3. Xác định giao của hai mặt phẳng vừa tạo và mặt phẳng chuẩn: Chọn công cụ  và nháy chuột lên mặt phẳng vừa tạo rồi làm ẩn đi mặt phẳng vừa tạo. Chúng ta đã tạo được đường thẳng vuông góc với AB.



4. Sử dụng công cụ  tạo một điểm C nằm trên đường thẳng vuông góc với AB vừa tạo ra ở bước trên.

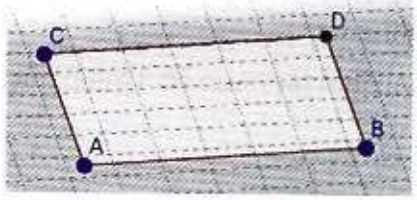


5. Sử dụng công cụ  tương tự như trong mặt phẳng kẻ các đường thẳng từ C song song với AB và từ B song song với AC. Hai đường này cắt nhau tại D.



6. Bây giờ chúng ta làm ẩn đi tất cả các đường thẳng có trên hình, chỉ để lại bốn điểm A, B, C, D.

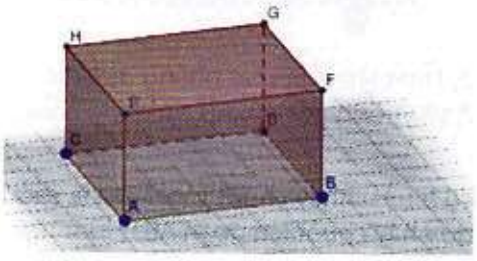
7. Sử dụng công cụ  , lần lượt nháy chuột lên các điểm A, B, D, C, A để tạo thành một hình chữ nhật như trong hình.



Hình 2.56. *Vẽ hình hộp chữ nhật*

Bước 2: Vẽ hình hộp chữ nhật dựa trên hình chữ nhật đã tạo ở bước 1. Có hai cách thực hiệnnhư sau:

Cách 1: Sử dụng công cụ trải hình lăng trụ



Hình 2.57

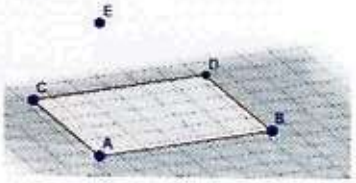
- Chọn công cụ 

- Đưa chuột vào bên trong hình ABDC, kéo thả chuột theo hướng thẳng đứng để tạo hình hộp.

Cách 2: Sử dụng công cụ tạo hình lăng trụ xiên

- Chọn công cụ điểm 

, nháy chuột lên mặt phẳng chuẩn bên ngoài ABDC để tạo điểm E, sau đó di chuyển điểm E đến vị trí phía trên điểm A.

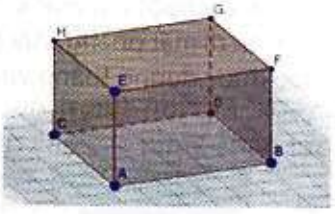


Hình 2.58

- Chọn công cụ  . Nháy chuột lên một vị trí bất kì bên trong hình ABDC.

- Nháy chuột chọn điểm E.

Lưu ý: Đây là hình hộp xiên, có thể di chuyển điểm E để quan sát.



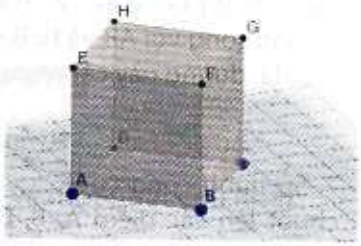
***Hình 2.59.** Tạo hình hộp xiên*

b) Vẽ hình lập phương với hai điểm tự do

Thao tác vẽ hình lập phương:

- Chọn công cụ  .

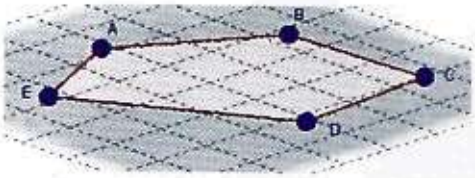
- Nháy chuột chọn hai điểm bất kì để tạo hình khối lập phương, lấy hai điểm này làm một cạnh.

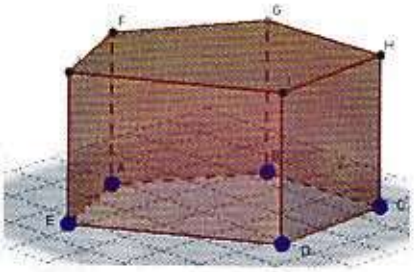


Hình 2.60

5 Vẽ hình lăng trụ đứng

Cách 1. Sử dụng công cụ trải hình lăng trụ đứng



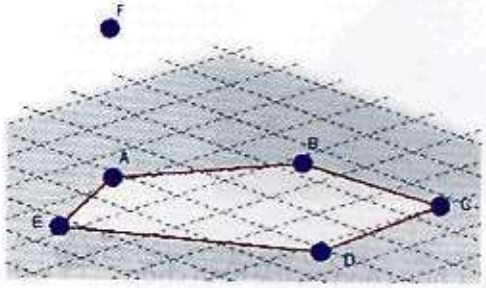


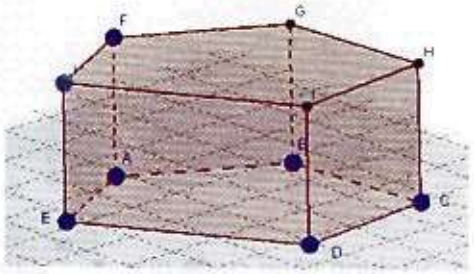
1. Sử dụng công cụ đa giác  tạo một hình đa giác bất kì trên mặt phẳng chuẩn (hoặc một mặt phẳng bất kì).

2. Chọn công cụ trải hình lăng trụ  đưa chuột vào bên trong hình đa giác, sau đó kéo thả chuột để tạo hình lăng trụ có đáy là hình đa giác.

Hình 2.61

Cách 2. Sử dụng công cụ tạo hình lăng trụ xiên





1. Sử dụng công cụ đa giác  tạo một hình đa giác bất kì trên mặt phẳng chuẩn (hoặc một mặt phẳng bất kì). Giả sử đa giác này bắt đầu từ đỉnh A. Sử dụng công cụ  để tạo một điểm nằm trong không gian phía trên A (trong hình này là điểm F).

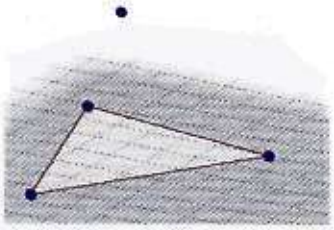
2. Chọn công cụ  . Nháy chuột lên một vị trí bất kì bên trong hình đa giác vừa tạo, sau đó nháy chọn điểm vừa tạo phía trên A (điểm F).

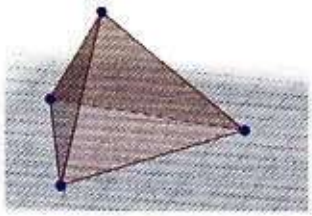
Hình 2.62

Lưu ý: Đây là hình lăng trụ xiên, có thể di chuyển điểm F để quan sát hình lăng trụ xiên này.

6 Vẽ hình chóp

a) Sử dụng công cụ hình chóp





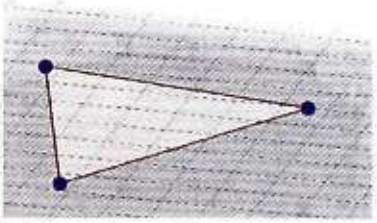
1. Sử dụng công cụ đa giác tạo một hình đa giác bất kì trên mặt phẳng chuẩn (hoặc một mặt phẳng bất kì). Sử dụng công cụ A để tạo một điểm nằm trong không gian ở phía trên của đa giác.

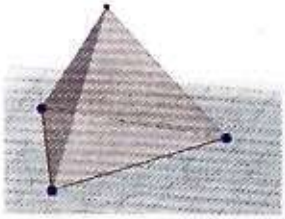
2. Chọn công cụ .

Nháy chuột lên một vị trí bất kì bên trong hình đa giác vừa tạo ra, sau đó chọn điểm vừa tạo phía trên.

Hình 2.63

b) Sử dụng công cụ trải hình chóp





1. Sử dụng công cụ đa giác tạo một hình đa giác bất kì trên mặt phẳng chuẩn (hoặc một mặt phẳng bất kì).

2. Chọn công cụ trải hình chóp , đưa chuột vào trong hình đa giác, sau đó kéo thả chuột để tạo hình chóp có đáy là hình đa giác này.

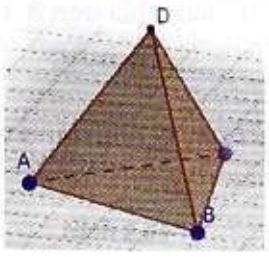
Hình 2.64

c) Công cụ vẽ hình chóp tam giác đều

Công cụ có chức năng tạo nhanh một hình chóp tam giác đều, hay còn gọi là tứ diện đều.

Thao tác: Chọn công cụ , sau đó nháy chuột chọn lên hai điểm là đỉnh của một cạnh của tứ diện đều cần tạo.

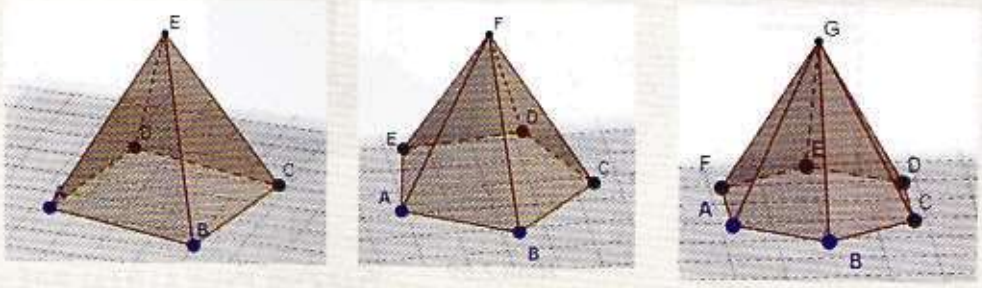
Hình chóp này có đỉnh thẳng chiếu xuống trọng tâm của đáy.



Hình 2.65

CÂU HỎI VÀ BÀI TẬP

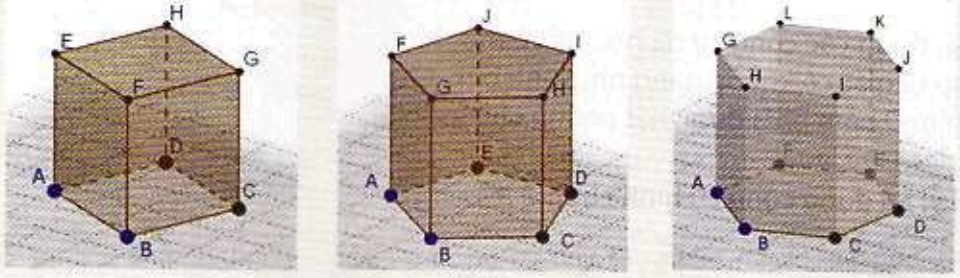
1. Vẽ một số hình chóp đa giác đều sau:



Hình 2.66

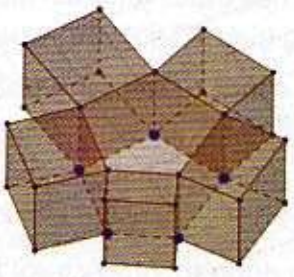
2. Theo em mặt phẳng chuẩn có phải là một đối tượng toán học nằm trong danh sách đối tượng của GeoGebra không?

3. Vẽ một số hình lăng trụ đa giác đều sau:



Hình 2.67

4. Sử dụng các công cụ đã học để vẽ hình khối có dạng hình 2.68:



Hình 2.68

5. Cho một điểm tự do trong không gian. Em có thể dùng chuột để di chuyển điểm này theo một hướng bất kì hay không?

6. Trong GeoGebra có nhiều cửa sổ cùng có chức năng thể hiện các đối tượng trong tệp. Chúng ta có thể sắp xếp các cửa sổ này sao cho hợp lí, dễ quan sát nhất.

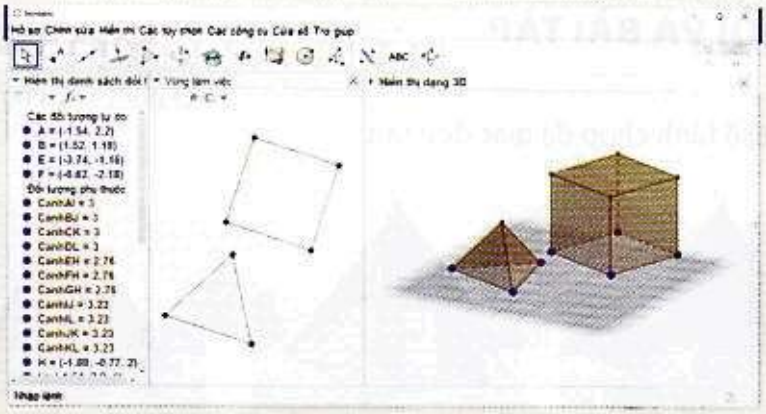
Em hãy sắp xếp các cửa sổ

(i) danh sách đối tượng,

(ii) vùng làm việc ,

(iii) không gian 3D

như trong hình 2.69:

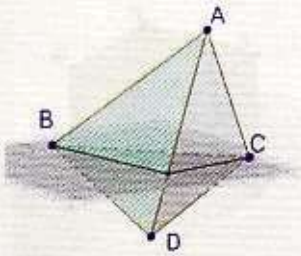


Hình 2.69

Trong màn hình làm việc này, em có thể dùng các công cụ của mặt phẳng để làm việc với các đối tượng phẳng, sau đó mở rộng trong không gian. Ví dụ trong cửa sổ *Vùng làm việc*, em tạo các hình tam giác đều, hình vuông, sau đó trong cửa sổ không gian 3D tạo ra các hình hộp chữ nhật và hình chóp đa giác đều.

7. Sử dụng các công cụ đã học để vẽ một hình chóp tứ giác có khuôn dạng như hình bên. Chú ý điểm D nằm phía dưới mặt phẳng chuẩn.

Gợi ý: Sử dụng công cụ hình chóp  .



Hình 2.70

8. Em hãy thực hiện thao tác sau sẽ làm cho các đối tượng hình học trên cửa sổ không gian 3D xoay tròn xung quanh trục thẳng đứng.

- Kéo chuột từ trái sang phải và thả nhanh chuột : Hình sẽ xoay ngược chiều kim đồng hồ.

- Kéo chuột từ phải sang trái và thả nhanh chuột: Hình sẽ xoay theo chiều kim đồng hồ.

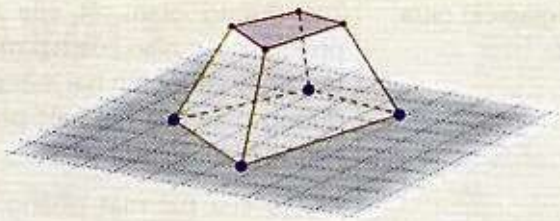
Muốn dừng thì nháy chuột lên một vị trí bất kì trên màn hình.

9. Từ một điểm nằm phía trên mặt phẳng chuẩn, em hãy vẽ một mặt phẳng song song với mặt phẳng chuẩn.

Gợi ý: Sử dụng công cụ  .

Hình 2.71

10. Sử dụng các công cụ đã biết, hãy vẽ hình chóp cụt có dạng như hình sau:



Hình 2.72

TÌM HIỂU MỞ RỘNG

Chúng ta tìm hiểu thêm một số công cụ vẽ hình trong không gian 3D.

1. Công cụ tạo đường thẳng trong không gian 3D

Công cụ	Chức năng	Thao tác
	Các công cụ đoạn, đường, tia thẳng	Thao tác của các công cụ này hoàn toàn tương tự như trong mặt phẳng. Chọn công cụ, sau đó lần lượt nháy chuột lên hai điểm trong không gian để tạo ra đoạn thẳng, đường thẳng, tia.
	Đường thẳng vuông góc	Công cụ này tạo quan hệ vuông góc, từ một điểm kẻ đường thẳng vuông góc với một đường thẳng khác hay với một mặt phẳng. Thao tác: Chọn điểm, sau đó là đường thẳng / mặt phẳng hoặc theo thứ tự ngược lại.
	Đường thẳng song song	Công cụ này tương tự như trong mặt phẳng, từ một điểm kẻ đường thẳng song song với một đường thẳng cho trước. Thao tác: Chọn điểm, sau đó nháy chuột lên đường thẳng hoặc theo thứ tự ngược lại.
	Đường thẳng phân giác	Công cụ này tương tự như trong mặt phẳng. Chức năng tạo đường phân giác của một góc tạo bởi ba điểm hoặc góc tạo bởi hai đường thẳng giao nhau.

Công cụ	Chức năng	Thao tác
	Đường giao cắt giữa hai mặt	Tạo đường giao cắt của hai đối tượng phẳng, ví dụ giao của hai mặt phẳng. Thao tác: nháy chuột lên hai mặt phẳng sẽ tạo ra đường giao của hai mặt phẳng này trong không gian. Ví dụ: - Giao của hai mặt phẳng là một đường thẳng. - Giao của mặt phẳng và hình cầu là một đường tròn.

2. Công cụ tạo mặt phẳng trong không gian 3D

Công cụ	Chức năng chính	Thao tác cụ thể
	Tạo mặt phẳng đi qua ba điểm trong không gian.	Thao tác: nháy chuột lần lượt lên ba điểm không thẳng hàng trong không gian để tạo mặt phẳng đi qua ba điểm này.
	Lệnh tạo mặt phẳng (tổng quát)	Công cụ này tạo mặt phẳng trong các trường hợp sau: - Đi qua một điểm và một đường thẳng. - Đi qua hai đường thẳng song song trong không gian. - Đi qua hai đường thẳng giao nhau trong không gian.
	Tạo mặt phẳng vuông góc.	Tạo mặt phẳng đi qua một điểm và vuông góc với một đường thẳng trong không gian. Thao tác: lần lượt nháy chuột lên điểm và đường thẳng.
	Tạo mặt phẳng song song.	Tạo mặt phẳng đi qua một điểm và song song với một mặt phẳng. Thao tác: lần lượt nháy chuột lên điểm và mặt phẳng.

INDEX

MỤC LỤC

CHƯƠNG I. LẬP TRÌNH ĐƠN GIẢN

CHƯƠNG II. PHẦN MỀM HỌC TẬP