§2. CÁC THÀNH PHẦN CỦA NGÔN NGỮ LẬP TRÌNH

A. Mục đích, yêu cầu

- Biết ngôn ngữ lập trình có ba thành phần cơ bản là: bảng chữ cái, cú pháp và ngữ nghĩa. Hiểu và phân biệt được ba thành phần này.
- Biết một số khái niệm: tên, tên chuẩn, tên dành riêng (từ khoá), hằng và biến.
- Yêu cầu học sinh ghi nhớ các quy định về tên, hằng và biến trong một ngôn ngữ lập trình. Biết cách đặt tên đúng và nhận biết được tên sai quy định.

B. Những điểm cần lưu ý và gợi ý tổ chức dạy học

Khi trình bày ba thành phần của một ngôn ngữ lập trình là *bảng chữ cái*, *cú pháp* và *ngữ nghĩa*, giáo viên nên minh hoạ bằng một vài ví dụ cụ thể. Giáo viên không nên dành quá nhiều thời gian phân tích quá sâu về cú pháp và ngữ nghĩa. Cần lưu ý chương trình dịch phát hiện được lỗi về cú pháp nhưng không phát hiện được lỗi ngữ nghĩa.

Giáo viên có thể dùng một đoạn chương trình Pascal đơn giản (hoặc C/C++) để dẫn tới các khái niệm bảng chữ cái, tên, biến, hằng,..., sau đó mới nêu định nghĩa các khái niệm này.

- *Về bảng chữ cái*. Giáo viên nên chuẩn bị sẵn một số minh hoạ trên giấy khổ lớn, ví dụ có thể giới thiệu bảng chữ cái trong Pascal như sau:

Loại kí tự	Biểu diễn của kí tự	Mã ASCII
Kí tự chữ cái in thường và in hoa	'A'.. 'Z'	65.. 90
	'a'.. 'z'	97.. 122
Kí tự chữ số (thập phân Ả Rập)	'0'.. '9'	48.. 57
Kí tự dấu cách	' '	32
Kí tự gạch dưới	'_'	
Kí tự các phép toán	'+', '-', '*', '/', '=', '>', '<'	
Kí tự dấu ngoặc	'(', ')', '{', '}', '[', ']'	
Kí tự khác	Dấu chấm '.' dấu phẩy ','	
	Dấu hai chấm ':' dấu chấm phẩy ';'	
	'"', '@', '^', '$', '#', '&'	

Giáo viên có thể so sánh bảng chữ cái trong ngôn ngữ lập trình và bảng chữ cái trong ngôn ngữ tự nhiên. Từ các kí hiệu (kí tự) trong bảng chữ cái và cú pháp của ngôn ngữ lập trình có thể tạo thành các câu lệnh và chương trình tương tự như từ bảng chữ cái và ngữ pháp của ngôn ngữ tự nhiên có thể diễn đạt thành câu và văn bản. Không yêu cầu học sinh nhớ hết các kí tự trong bảng chữ cái. Mã ASCII của các kí tự cũng chỉ giới thiệu để biết.

- *Về cú pháp*. Cần nêu đầy đủ ý nghĩa của cú pháp như sách giáo khoa trình bày. Có thể lấy thêm một vài ví dụ đơn giản trong Pascal để minh hoạ một số tổ hợp viết không đúng cú pháp.
- *Về ngữ nghĩa*. Ngữ nghĩa xác định ý nghĩa của các tổ hợp kí tự trong chương trình. Giáo viên minh hoạ bằng ví dụ trong sách giáo khoa là đủ.
- *Tên*. Cần nêu ý nghĩa của việc đặt tên và khai báo tên cho các đối tượng:

– Để quản lí và phân biệt các đối tượng trong chương trình;

– Để gợi nhớ nội dung của đối tượng.

Trong Pascal, tên do người lập trình đặt cần phải khai báo trước khi sử dụng, tuy nhiên một số ngôn ngữ khác không cần trực tiếp khai báo trước mà gián tiếp qua khởi trị ban đầu. Nhưng vì lí do sư phạm nên sách giáo khoa đưa ra yêu cầu *cần khai báo trước* đối với những tên này.

- *Hằng và biến*. Giáo viên nên hướng dẫn học sinh tìm sự khác biệt giữa hằng và biến. Giáo viên chuẩn bị sẵn bảng các tên dành riêng, tên chuẩn của một ngôn ngữ lập trình (Pascal hoặc C++), một số ví dụ về tên, hằng, biến đúng và sai trên giấy khổ lớn. Nên dùng các bài tập trắc nghiệm ngay trong giờ để học sinh hiểu rõ hơn các quy định về tên và hằng. Giáo viên

nên phát vấn học sinh thông qua các ví dụ minh hoạ trong bảng nêu trên, đồng thời động viên học sinh lên bảng ghi thêm một số tên, hằng đúng. Giáo viên cũng lưu ý sự khác biệt giữa hằng và biến trong khai báo và cách sử dụng chúng.

- Lưu ý thêm trong Pascal:
  - Để biểu diễn *hằng kí tự dấu nháy đơn* cần phải coi hằng này là hai dấu nháy đơn đặt trong một cặp nháy đơn (bốn dấu nháy đơn liền nhau) dễ nhầm với diễn đạt kí tự nháy kép bằng cách đặt kí tự nháy kép trong cặp nháy đơn.
  - Có thể sử dụng hằng số dạng hexa, cần thêm kí tự $ trước giá trị số, ví dụ $A1 biểu diễn số 161 trong hệ thập phân.
- Ngoài ra, giáo viên cũng lưu ý học sinh, dạng dấu phẩy động của số thực nêu trong sách giáo khoa Tin học 10 là dạng được xem xét khi bàn về lưu trữ số thực trong bộ nhớ (số 12,345 được biểu diễn là 0.12345E2) khác với cách biểu diễn số thực trong Pascal (12,345 được hiển thị trên màn hình là 1.2345E1).
- *Về chú thích trong chương trình*. Các chú thích trong chương trình được bỏ qua khi dịch chương trình. Giáo viên nêu rõ ý nghĩa của việc dùng đoạn chú thích. Cách sử dụng chú thích của Pascal hoặc C++ được mô tả như trong bảng sau:

Ngôn ngữ	Chú thích một dòng	Chú thích nhiều dòng
Pascal	{chú thích}	(* chú thích ... *) {chú thích ... }
C++	// chú thích	/* chú thích ... */

- *Về tiết bài tập*

Dùng để chữa các câu hỏi và bài tập cuối chương I:

– Các bài 1 và 2 có thể phát vấn hoặc xây dựng thành dạng trắc nghiệm.

– Các bài 3 và 4 nên hệ thống thành bảng so sánh đối chiếu.

– Các bài 5 và 6 khuyến khích dạng trắc nghiệm.

Giáo viên có thể nêu các câu hỏi và bài tập tương tự để học sinh giải bài mới ngay tại lớp.

C. Kiến thức bổ sung

- Chương trình dịch thường có các chức năng sau:

– *Phân tích từ vựng.* Nhằm phát hiện các kí tự không thuộc bảng chữ cái, nhận dạng kí tự và mỗi từ thuộc loại nào (tên dành riêng, tên chuẩn, tên biến, tên hằng, từ khoá, dấu phép toán, dấu ngăn cách các lệnh,…). *Phân tích từ vựng* còn có chức năng giao diện với người dùng (bỏ qua các dòng chú thích, lưu số dòng, số cột, hiện các từ khoá, biến, chữ số, đoạn chú thích với màu khác,…).

– *Phân tích cú pháp.* Sau phân tích từ vựng là phân tích cú pháp. Dựa trên cú pháp của ngôn ngữ để phân tích các từ và câu nhằm phát hiện lỗi. Các lỗi thường gặp là từ viết sai hoặc thiếu hoặc tổ hợp từ chưa đúng cú pháp.

Ví dụ

+ Trong Pascal, câu lệnh gán là dấu hai chấm đi liền dấu bằng (:=) nhưng trong C/C++ là một dấu bằng (=);
+ Trong Pascal, hằng xâu được đặt trong cặp nháy đơn nhưng trong C/C++ hằng xâu được đặt trong cặp nháy kép;
+ Trong Pascal, toán tử so sánh bằng là một dấu bằng (=) nhưng trong C/C++ là hai dấu bằng liền nhau (==);
+ Trong C/C++, điều kiện là một biểu thức lôgic thì bắt buộc phải đặt nó trong cặp ngoặc tròn nhưng trong Pascal thì không nhất thiết;
+ Trong Pascal trong câu lệnh ***if ... then … else*** thì trước ***else*** không có dấu chấm phẩy ngăn cách lệnh nhưng trong C/C++ thì bắt buộc phải có;
+ …

– *Phân tích ngữ nghĩa* chỉ được thực hiện sau khi chương trình không còn lỗi cú pháp. Một số nội dung của phân tích ngữ nghĩa là kiểm tra kiểu các toán hạng có phù hợp với các toán tử không, dòng lệnh có đúng quy định của các cấu trúc điều khiển không, các tên có nhất quán không, quan hệ giữa các tên có phù hợp không,… Như vậy, ngữ nghĩa xác định tính chất và thuộc tính của các tổ hợp kí tự tạo thành các dòng lệnh có trong chương trình.

– Các lỗi ngữ nghĩa thường được người lập trình phát hiện khi thực hiện chương trình trên bộ dữ liệu cụ thể và phụ thuộc vào ngữ nghĩa của từng trình dịch.

+ *Ví dụ,* câu lệnh *N*:= *i* + *j*; trong trình dịch Turbo Pascal (TP) là câu lệnh đúng về cú pháp nếu các biến *N*, *i*, *j* đã được khai báo trước đó. Nội dung câu lệnh này là tính tổng giá trị của *i* và *j* sau đó gán tổng này cho *N*. Nếu *i* và *j* khai báo kiểu *integer* (có giá trị trong khoảng −32768 đến 32767) và giả sử rằng *i* = 20000, *j* = 15000, thì cho dù *N* được khai báo là kiểu *longint* (có giá trị từ −2147483648 tới 2147483647) thì câu lệnh vẫn sai về ngữ nghĩa trong TP. Trong TP quy định kết quả phép cộng hai số kiểu integer phải trả về kiểu integer. Nhưng 20000 + 15000 = 35000 đã vượt qua giới hạn của

integer, nên kết quả trả về của phép cộng sẽ gán giá trị sai cho *N*. Nếu sửa lại khai báo *i* hoặc *j* kiểu longint thì câu lệnh lại đúng. Tuy nhiên, nếu khai báo *i*, *j* có kiểu integer và N có kiểu longint thì không gặp lỗi trong trình dịch Free Pascal. Vậy lỗi ngữ nghĩa sinh ra khi người lập trình thể hiện thuật toán hoặc trình bày đúng cú pháp nhưng lại không đáp ứng các yêu cầu về ngữ nghĩa của ngôn ngữ hoặc của chương trình dịch cụ thể.

- *Bảng chữ cái* là tập hữu hạn các kí hiệu gồm *một số* kí tự trong bộ mã ASCII. Mỗi kí tự trong bộ mã ASCII được mã hoá tương ứng với một số nguyên có giá trị từ 0 đến 255 (kiểu byte). Nên chú ý tới mã của những kí tự thường dùng như các kí tự chữ cái in hoa (tiếng Anh) 'A'..'Z' có mã ASCII từ 65 đến 90. Các kí tự chữ cái thường (tiếng Anh) 'a'..'z' có mã ASCII từ 97 đến 122. Dấu cách có mã 32. Các kí tự số '0'..'9' có mã từ 48 đến 57,... Các kí tự có mã từ 0 đến 31 là kí tự điều khiển.
- Mỗi từ là một dãy hữu hạn các kí hiệu thuộc bảng chữ cái được viết liền kề nhau (không chứa dấu cách). Các tổ hợp từ tuân theo các quy định về cú pháp tạo thành các câu lệnh có một ý nghĩa nào đó theo ngữ nghĩa của ngôn ngữ. Các ngôn ngữ khác nhau nói chung có bảng chữ cái tương tự như nhau (gồm các chữ cái tiếng Anh, các chữ số, các dấu phép toán và một vài kí tự thông dụng khác dùng trong diễn đạt ngôn ngữ) nhưng có thể khác nhau một vài kí tự nào đó. Ví dụ, trong bảng chữ cái của Pascal không có sáu kí tự: ? \ ! % | " nhưng trong C/C++ lại có. Trong chương trình, các tên và câu lệnh chỉ dùng những kí tự trong bảng chữ cái.
- Tên: Mỗi đối tượng (hằng, biến, hàm, kiểu dữ liệu,...) đều được đặt tên. Mỗi trình dịch đều có bộ chức năng quản lí các tên (Indentifier) đã gán cho các đối tượng. Cách đặt tên là những quy ước phải tuân theo trong mỗi ngôn ngữ lập trình và trình dịch tương ứng (quy định về độ dài, những kí tự thuộc tên, tên bắt đầu bằng kí tự loại nào, tên dành riêng (từ khoá), tên chuẩn, tên do người lập trình đặt,...). Tên không được trùng với từ khoá và không nên trùng với tên chuẩn. Các ngôn ngữ lập trình khác nhau có thể có những quy ước khác nhau:
  - Trong Pascal, quy định tên chỉ gồm chữ cái, chữ số và kí tự gạch dưới, tên không bắt đầu bằng chữ số, không phân biệt chữ hoa và chữ thường trong tên. Ví dụ, chương trình Pascal đặt tên bán kính đường tròn nội tiếp là *r*, bán kính đường tròn ngoại tiếp là *R* thì mắc lỗi trùng tên (Duplicate indentifier).
  - Trong C/C++ lại phân biệt chữ hoa và chữ thường trong tên.

Chương trình dịch sẽ thông báo lỗi về tên (nếu có) khi tiến hành dịch. Trong Pascal nếu gặp lỗi về đặt tên, thường có thông báo như: "*Cannot Access this symbol*" (không thể truy cập kí hiệu này) *hoặc* "*Unknown indentifier*" (tên chưa xác định). Trong C/C++ gặp các thông báo như: {$ *Ambiguous member name* $} (tên thành viên chưa rõ ràng) hoặc {$ '*identifier*' *cannot be declared in* ... $} (tên chưa được

khai báo trong …). Thông thường các trình dịch cho phép theo dõi giá trị của các tên (biến, hằng) trong quá trình thực hiện chương trình. Ví dụ, bật chức năng *Add Watch* (trong Pascal là **Ctrl+F7**, trong Dev C++ là **F4**).

- Mỗi ngôn ngữ lập trình thường có những quy định riêng về cách biểu diễn hằng:
  - Thông thường hằng số nguyên không có dấu chấm thập phân (.).
  - Hằng số thực có thể viết dưới dạng thập phân hoặc dạng dấu phẩy động. Ví dụ, dạng dấu phẩy động -3.12E+3 nghĩa là $-3{,}12\times10^{3}$ (bằng -3120), còn dạng dấu phẩy động 1.235E-4 nghĩa là $1{,}235\times10^{-4}$ (bằng 0,0001235).
  - Trong Pascal, hằng kí tự đặt trong cặp nháy đơn (có thể trong các ngôn ngữ lập trình cụ thể còn có những quy định khác, như trong C thường dùng '\141' để biểu diễn kí tự 'a' vì mã ASCII thập phân của 'a' là 97 đổi thành 141 trong hệ cơ số 8).
  - Hằng xâu đặt trong cặp nháy đơn (Pascal) hoặc trong cặp nháy kép (C/C++).
  - Hằng cũng thể hiện kiểu.

  Ví dụ, 23 là hằng số nguyên, 0.5 là hằng số thực, 'Ha Noi' là hằng xâu trong Pascal, "Ha Noi" là hằng xâu kí tự trong C++.

  Một số ngôn ngữ có thể còn những quy định chi tiết hơn, ví dụ trong C++ hằng 128u là hằng có kiểu nguyên không dấu, 024 là hằng nguyên viết trong hệ cơ số 8 (của số 20 trong hệ thập phân),… Trong ngôn ngữ lập trình còn quy định một số hằng chuẩn, như trong Pascal có *MaxInt* = 32767, *MaxLongInt* = 2147483647, *True*, *False*, *Nil*,…

  Hằng cũng có thể được đặt tên. Tên hằng luôn thay cho giá trị của hằng. Ví dụ trong C/C++ khai báo:

  #define MAX 1000

  thì ở mọi vị trí trong chương trình tên *MAX* luôn thay cho giá trị 1000, 2**MAX* sẽ là 2000,...

  Trong Pascal, hằng xâu rỗng thường được kí hiệu là cặp nháy đơn gồm hai nháy đơn sát cạnh nhau, còn kí tự dấu cách được kí hiệu là cặp nháy đơn cách nhau một khoảng trống (đủ viết một kí tự).

  Cần phân biệt hằng số và hằng xâu chữ số. Ví dụ 123 là hằng số còn '123' là hằng xâu trong Pascal.

- Biến là những đại lượng có thể thay đổi giá trị trong quá trình thực hiện chương trình. Biến cũng được đặt tên. Tên biến mang giá trị của biến tại từng thời điểm thực hiện chương trình. Giá trị của biến được lưu trữ trong ô nhớ, do đó có thể truy cập tới ô nhớ này để đọc giá trị của biến. Khi nói tới biến ta quan tâm tới địa chỉ ô nhớ của biến và cũng quan tâm tới giá trị hiện thời đang được lưu trữ trong ô nhớ đó.