II - HƯỚNG DẪN CHI TIẾT

§3. CẤU TRÚC CHƯƠNG TRÌNH

A. Mục đích, yêu cầu

- Hiểu chương trình là sự mô tả của thuật toán bằng một ngôn ngữ lập trình;
- Biết cấu trúc của một chương trình đơn giản: cấu trúc chung và các thành phần;
- Nhận biết được các phần của một chương trình đơn giản.

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

1. Về cấu trúc chương trình

Cấu trúc chương trình được diễn tả gồm:

[<*Phần khai báo*>]

<*Phần thân*>

Phần khai báo đặt trong cặp ngoặc vuông có nghĩa là có thể có hoặc không. Phần khai báo gồm tên chương trình, tên thư viện, hằng và biến (cách khai báo biến được trình bày ở §5). Khai báo tên chương trình chỉ có ý nghĩa để ghi nhớ tên bài toán cần giải. Một số ngôn ngữ lập trình không cần phần khai báo tên chương trình, có thể dùng các dòng chú thích đặt ở ngay đầu chương trình để tóm tắt nội dung bài toán được giải bằng chương trình này và một số thông tin cần thiết khác như tên thuật toán, tên lập trình viên, thời gian viết,… Tuỳ từng chương trình cụ thể, phần khai báo tên thư viện, hằng và biến có thể có hoặc không có. Khai báo các đối tượng nhằm tổ chức chúng trong bộ nhớ trong và quản lí được chúng khi dịch hoặc thực hiện chương trình. Chỉ những đối tượng được khai báo hoặc được gián tiếp khai báo qua khởi trị thì sau đó mới sử dụng được.

Phần thân chương trình bắt buộc phải có và được đánh dấu bắt đầu bằng một từ khoá (hoặc kí hiệu) và đánh dấu kết thúc bằng một từ khoá (hoặc kí hiệu) tương ứng.

Trong Pascal, phần thân chương trình được bắt đầu bằng từ khoá ***begin*** và đánh dấu kết thúc bằng từ khoá ***end*** (sau ***end*** có dấu chấm). Trong C++ dùng cặp ngoặc { và }. Trong thân chương trình chứa các câu lệnh cần thực hiện.

Chương trình Pascal đơn giản có dạng như cột bên phải của bảng 1 sau đây:

Bảng 1

<table>
<tr><th>Giải thích</th><th>Cấu trúc chương trình Pascal đơn giản</th></tr>
<tr><td>Phần khai báo</td><td rowspan="2">program <tên chương trình>;
uses <tên các thư viện>;
const <tên hằng> = <giá trị của hằng>;
var <tên biến> : <kiểu dữ liệu>;
(* có thể còn những khai báo khác *)
begin
[<dãy lệnh>]
end.</td></tr>
<tr><td>Phần thân chương trình chính (từ begin đến end)</td></tr>
</table>

2. Về các thành phần của chương trình

Các loại tên như tên chương trình, tên các thư viện, tên hằng phải được đặt và khai báo đúng cú pháp.

Khai báo hằng sẽ tạo điều kiện thuận lợi cho việc chỉnh sửa lại giá trị của hằng trong toàn bộ chương trình khi cần thiết. Khi sửa lại giá trị của hằng tại phần khai báo thì toàn bộ tên của hằng này ở mọi vị trí trong chương trình đều được nhận giá trị mới.

Trong chương trình có thể khai báo tên các thư viện hàm/thủ tục chuẩn hoặc các thư viện chứa các hàm người lập trình tự định nghĩa.

Pascal dùng dấu chấm phẩy để ngăn cách các câu lệnh và nếu còn câu lệnh khác nữa sau câu lệnh ghép thì sau ***end*** có dấu chấm phẩy (;). Kết thúc chương trình chính là ***end*** và dấu chấm (.). Mọi dòng lệnh sau ***end.*** không có ý nghĩa.

C. Kiến thức bổ sung

- Một số thư viện của Pascal là DOS (chứa các hàm chuẩn giao tiếp với DOS), SYSTEM (gồm các hàm cơ sở liên quan đến ngôn ngữ Pascal), CRT (chứa các hàm vào/ra chuẩn làm việc với màn hình và bàn phím), GRAPH (chứa các hàm đồ hoạ).

  Khai báo thư viện trong Pascal có dạng:

  uses *<danh sách thư viện>*;

  Trong C++ để khai báo thư viện, ngay đầu chương trình dùng lệnh:

  #include *<tên thư viện chuẩn>*

- Bộ tiền xử lí sẽ dịch thư viện, đưa các hàm trong thư viện vào phục vụ chương trình. Khi thực hiện chương trình, các thư viện này và chương trình đều được dịch thành mã đích. Các thư viện là những đơn vị độc lập, được dịch riêng trước khi chương trình chính được dịch. Trong môi trường lập trình của nhiều ngôn ngữ có hạn chế mỗi chương trình chính hoặc thư viện khi dịch và thực hiện chỉ sử dụng dung lượng bộ nhớ không vượt quá một giới hạn nào đó, chẳng hạn với Pascal giới hạn đó là 64KB. Vì vậy sử dụng thư viện có một tác dụng là mở rộng gián tiếp vùng nhớ dành cho chương trình. Các yếu tố quan trọng trong quá trình biên dịch, liên kết mã nguồn là số lượng các thư viện, các thành phần được biên dịch trước (tệp tiền tố header), độ phức tạp của mã lệnh, yêu cầu tối ưu trong biên dịch và liên kết, kích thước các môđun dịch.