II - HƯỚNG DẪN CHI TIẾT

§9. CẤU TRÚC RẼ NHÁNH

A. Mục đích, yêu cầu

- Hiểu nhu cầu của cấu trúc rẽ nhánh trong biểu diễn thuật toán;
- Hiểu câu lệnh rẽ nhánh (dạng thiếu và dạng đủ);
- Hiểu câu lệnh ghép;
- Sử dụng cấu trúc rẽ nhánh trong mô tả thuật toán của một số bài toán đơn giản.
- Viết được các lệnh rẽ nhánh dạng thiếu, rẽ nhánh dạng đầy đủ và áp dụng để thể hiện được thuật toán của một số bài toán đơn giản.

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

§9 được dạy trong 1 tiết.

- *Về cấu trúc rẽ nhánh*

– Giáo viên cần nêu rõ ý nghĩa của cấu trúc rẽ nhánh (thông qua ví dụ dẫn dắt như sách giáo khoa đã trình bày trong mục *1. Rẽ nhánh*).

— Ý nghĩa của cấu trúc rẽ nhánh: Cấu trúc rẽ nhánh là một điều khiển chọn thực hiện hay không thực hiện công việc phù hợp một điều kiện đang xảy ra. Thực chất là "dạy máy" học cách xử lí tình huống.

– Nếu đưa vào khái niệm câu lệnh rỗng là câu lệnh không thực hiện thao tác nào thì dạng thiếu có thể viết thành dạng đủ nếu coi *câu lệnh 2* là câu lệnh rỗng.

– Dạng đủ có thể lồng nhau.

Ví dụ

Khi kết luận về nghiệm của phương trình bậc hai một ẩn:

$$ax^2 + bx + c = 0 \ (a \neq 0)$$

dựa theo biệt số delta (D) có thể viết đoạn chương trình là:

```
if D<0 then writeln('PT vo nghiem ')
else
   if D=0 then writeln('PT co nghiem kep ',-b/(2*a):6:2)
   else
```

```
begin
    x:=(-b-sqrt(D))/(2*a);
    writeln('PT co 2 nghiem ',x:6:2,' va ',(-b/a-x):6:2);
end;
```

– Một lỗi thường gặp của người lập trình khi sử dụng câu lệnh *if* là không tạo câu lệnh ghép trong trường hợp cần thiết. Đây là lỗi mà người lập trình khó phát hiện vì trình dịch không phát hiện giúp được. Khi có nhiều câu lệnh *if* đi liền nhau thường tạo ra nhiều tiềm ẩn dẫn tới lỗi (đặc biệt xảy ra khi có nhiều *if* hơn *else*, người ta thường gọi hiện tượng này là hiện tượng *else lỏng lẻo*). Để tránh hiện tượng này khi viết dấu bắt đầu của lệnh ghép thì cũng viết luôn dấu kết thúc của lệnh ghép, sau đó viết chèn dãy lệnh cần thực hiện vào giữa hai dấu này. Ví dụ hai đoạn chương trình sau đây có nội dung khác nhau:

Đoạn 1:

```
if <biểu thức> then
  begin
   <câu lệnh 1>;
    if <biểu thức 2> then
    <câu lệnh 2>
  end
else <câu lệnh 3>;
```

Trong đoạn 1, *else* tương ứng với *if* trước.

Đoạn 2:

```
if <biểu thức> then
    <câu lệnh 1>;
if <biểu thức 2> then
    <câu lệnh 2>
else <câu lệnh 3>;
```

Trong đoạn 2, *else* tương ứng với *if* sau.

- *Về phương pháp giảng dạy*

– Giáo viên nên chuẩn bị sẵn minh hoạ cú pháp và sơ đồ cấu trúc của câu lệnh *if-then* (hai dạng) trên giấy khổ lớn. Dựa vào minh hoạ này, giáo viên hướng dẫn học sinh so sánh dạng thiếu và dạng đủ.

– Nên tập trung phần lớn thời gian để học sinh tìm hiểu hai ví dụ minh hoạ (mục 4. *Một số ví dụ*) của sách giáo khoa. Giáo viên dành thời gian cho học sinh đọc lại sách giáo khoa tại lớp, giáo viên phân tích một số câu lệnh chủ yếu trong ví dụ.

– Có thể tiến hành giảng dạy theo phương pháp giao tiếp bằng cách giới thiệu các ví dụ rồi để học sinh nghiên cứu, từ đó giáo viên dẫn tới các khái niệm cần truyền đạt như cấu trúc rẽ nhánh, câu lệnh *if-then*,... Phương pháp này dễ tạo sự cuốn hút học sinh vào việc học.

- Phần củng cố bài học, giáo viên có thể chọn thêm một ví dụ đơn giản để học sinh tự giải, đồng thời giáo viên nhấn mạnh ý nghĩa của câu lệnh rẽ nhánh như đã nêu trên.
- Bài về nhà: các câu hỏi và bài tập 1, 2, 4 ở cuối chương.

C. Kiến thức bổ sung

Mỗi ngôn ngữ lập trình có quy định cú pháp của dạng *if đầy đủ* là khác nhau. Trong C/C++ nếu *câu lệnh 1* không là câu lệnh ghép thì nó phải được kết thúc bởi dấu chấm phẩy trước từ khoá *else*. Điều này trong Pascal lại là không hợp lệ. Cụ thể trong C/C++, dạng *if đầy đủ* là:

if <*biểu thức*>

    câu lệnh 1; *//nếu câu lệnh 1 là câu lệnh đơn*

else *câu lệnh* 2;