§7. CÁC THỦ TỤC CHUẨN VÀO/RA ĐƠN GIẢN

A. Mục đích, yêu cầu

- Biết các lệnh vào/ra đơn giản để nhập dữ liệu từ bàn phím hoặc đưa dữ liệu ra màn hình.
- Viết được một số lệnh vào/ra đơn giản.

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

a) Sử dụng thủ tục chuẩn vào/ra đơn giản là công việc thường xuyên gặp khi lập trình. Trong suốt quá trình dạy lập trình, giáo viên nên quan tâm hướng dẫn sử dụng các thủ tục này. Tuy nhiên, bước đầu cũng không nên trình bày quá nhiều ví dụ minh hoạ về mọi trường hợp sử dụng khác nhau đối với các

thủ tục này. Chỉ nên giới thiệu một số trường hợp như sách giáo khoa đã trình bày, quá trình học sau này sẽ giới thiệu bổ sung dần. Có thể yêu cầu một số học sinh dự đoán kết quả thực hiện một số lệnh vào/ra cụ thể (giáo viên ghi trên bảng). Nên minh hoạ màn hình trên bảng như một hình chữ nhật có dòng và cột để có thể chỉ rõ vị trí tương đối của các giá trị được in trên màn hình.

b) Khi gặp câu lệnh *read* (hoặc *readln*), chương trình sẽ chờ người dùng nhập giá trị cho danh sách biến và nhấn phím **Enter**, chỉ sau khi nhấn phím **Enter** thì việc nhập giá trị cho danh sách biến mới kết thúc và lệnh tiếp theo được thực hiện.

c) Khi nhập giá trị cho danh sách biến phải chú ý các giá trị được nhập có kiểu tương ứng với các biến trong danh sách, giữa hai giá trị liên tiếp phải nhấn phím **Space** hoặc phím **Enter**.

d) Việc nhập giá trị của biến từ bàn phím được kết thúc bởi việc nhấn phím **Enter** nên không phân biệt *read* và *readln*. Do đó, khi nhập từ bàn phím nên dùng *readln*. Ngoài ra, *readln* không tham số có tác dụng chờ người dùng nhấn phím **Enter** trước khi tiếp tục thực hiện chương trình, nhờ đó người dùng có thể xem kết quả trước đó còn lưu trên màn hình. Tuy nhiên, lưu ý khi không cần xem kết quả trên màn hình thì không nên dùng *readln* vì khi dùng *readln* không tham số và không có thông báo hướng dẫn thì người dùng có thể nhầm tưởng chương trình đang chạy và không nhấn phím **Enter**.

e) Về việc đưa ra kết quả theo quy cách: giáo viên không nên đưa ra quá nhiều ví dụ về các trường hợp đưa kết quả ra với những quy cách khác nhau, chỉ cần nêu ví dụ như sách giáo khoa là đủ. Điều quan trọng cần lưu ý ở đây là ý nghĩa của việc đưa ra theo quy cách nhằm tạo giao diện với người dùng tốt hơn.

C. Kiến thức bổ sung

Với Pascal, có hai cách trình bày dữ liệu trên màn hình gồm dạng không quy cách và dạng có quy cách.

- Dạng không quy cách sẽ căn lề theo bên trái. Số thực được viết ra dưới dạng dấu phẩy động.

  Ví dụ

  ```
  x:= 12.87;
  writeln(x);
  ```

  thì kết quả trên màn hình là:

  ```
  1.2870000000E+01
  ```

- Dạng có quy cách sẽ căn lề theo bên phải. Dạng này có quy định độ rộng của biểu diễn, riêng trong trường hợp biểu diễn số thực còn quy định cả số chữ số phần phân trong biểu diễn dữ liệu.

  Ví dụ

  ```
  write(x:m:n);
  ```

  với *x* là biến kiểu thực, *m* là số nguyên thể hiện số kí tự trong biểu diễn số *x* (*m* gọi là độ rộng của biểu diễn), *n* là số chữ số thập phân trong biểu diễn của *x*.

  Nếu phần dành cho các chữ số phần nguyên (*m* – (*n* + 1)) lớn hơn số chữ số phần nguyên của *x* thì trình dịch của Pascal thêm các dấu cách vào bên trái biểu diễn của *x* sao cho đủ. Trong trường hợp ngược lại, Pascal vẫn hiển thị đủ các chữ số phần nguyên của *x*.

  Nếu *n* nhỏ hơn số chữ số phần thập phân đã cho của *x* thì chỉ viết giá trị đã làm tròn của *x* chính xác đến *n* chữ số thập phân. Nếu *n* lớn hơn số chữ số phần thập phân của *x* thì các số 0 được điền thêm vào cuối phần thập phân cho đủ *n* chữ số phần thập phân.

  Ví dụ

  ```
  x:= 12.87;
  writeln(x:5:1);
  ```

  trên màn hình sẽ là

  ```
  _12.9
  ```

  chứa năm kí tự, kí tự đầu là dấu cách.

  Ví dụ

  ```
  writeln(x:7:4);
  ```

  trên màn hình sẽ là

  ```
  12.8700
  ```

  chứa bảy kí tự.

  Trong C++ sử dụng dòng nhập, xuất dữ liệu là *cin* (luồng vào) và *cout* (luồng ra) cùng các toán tử đưa dữ liệu vào (>>) và đưa dữ liệu ra (<<). Luồng vào hoặc luồng ra quản lí dãy các kí tự liên tiếp được đưa vào hoặc đưa ra và có nhiều thuộc tính để tạo điều kiện nhập/xuất dữ liệu có đầy đủ chức năng và thuận lợi. Các toán tử >> và << có thể được dùng liên tiếp trong cùng một câu lệnh để thực hiện tuần tự nhiều thao tác đưa dữ liệu ra hoặc nhiều thao tác đưa dữ liệu vào. Ví dụ:

  ```
  Int x=10;
  Cout.fill('0');
  Cout.width(5);
  Cout<<x;
  ```

  Giá trị *x* bằng 10 được xuất ra màn hình với khuôn dạng năm kí tự (sử dụng thuộc tính *width* của luồng ra *cout*), trong đó ba kí tự đầu được thay thế (sử dụng thuộc tính *fill* của luồng ra *cout*) bằng các số 0.