II - HƯỚNG DẪN CHI TIẾT

§14. KIỂU DỮ LIỆU TỆP

§15. THAO TÁC VỚI TỆP

A. Mục đích, yêu cầu

- Biết khái niệm và vai trò của kiểu tệp;
- Biết hai cách phân loại tệp: theo cách tổ chức dữ liệu và theo cách truy cập;
- Hiểu bản chất của tệp văn bản;

- Biết các bước làm việc với tệp: gắn tên cho biến tệp, mở tệp, đọc/ghi tệp, đóng tệp;
- Biết khai báo biến tệp và các thao tác cơ bản với tệp văn bản;
- Biết sử dụng một số hàm và thủ tục chuẩn làm việc với tệp.

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

- Cách tổ chức tệp và nội dung các thao tác vào/ra tệp trong các ngôn ngữ lập trình là khác nhau. Tìm hiểu sâu về vấn đề này không thuộc phạm vi tin học phổ thông. Với mục tiêu đặt ra ở trên, chỉ cần đề cập những nội dung này trong phạm vi ngôn ngữ lập trình Pascal là đủ. Những ý kiến dưới đây liên quan đến ngôn ngữ Pascal.

– Về vai trò kiểu tệp: Dữ liệu kiểu tệp có nhiều điểm khác biệt so với các kiểu dữ liệu đã học ở các chương trước. Dữ liệu kiểu tệp được lưu trữ lâu dài ở bộ nhớ ngoài. Đặc điểm này mở ra khả năng rộng lớn cho việc lưu trữ thông tin (chỉ tuỳ thuộc vào dung lượng của đĩa).

– Phân loại tệp: Sách giáo khoa giới thiệu hai kiểu phân loại tệp, theo cách tổ chức dữ liệu của tệp trên bộ nhớ ngoài và theo cách thức truy cập tệp.

- Xét theo cách tổ chức dữ liệu:

– Tệp văn bản gồm các kí tự được tổ chức và quản lí theo từng dòng. Khi truy cập tệp, con trỏ tệp được di chuyển trên các dòng tuần tự từ đầu dòng về cuối dòng và từ dòng đầu đến dòng cuối của tệp. Cuối mỗi dòng có nhóm kí tự điều khiển báo hết dòng và chuyển tới đầu dòng tiếp theo. Cách truy cập tệp văn bản thường là truy cập tuần tự.

– Tệp có cấu trúc là tệp mà các thành phần của nó được tổ chức theo một cấu trúc nhất định (ví dụ như tổ chức theo các phần tử cùng kiểu). Cách truy cập tệp có cấu trúc thường là truy cập trực tiếp, do biết được kích thước của từng phần tử nên có thể định vị con trỏ tệp đến ngay vị trí (phần tử) cần truy cập.

- Khai báo biến tệp văn bản

  var <*tên biến tệp*>: **text**;

  Tên biến tệp phải tuân theo đúng quy cách đặt tên (để cho gọn, *tên biến tệp* còn được gọi là *biến tệp*). Khai báo *biến tệp* để sau đó có thể thực hiện các thao tác với tệp thông qua *biến tệp*. Cần phân biệt tên biến tệp khác tên tệp.

- Thao tác với tệp

a) Gắn tên tệp với biến tệp

Gắn tên tệp với biến tệp thực chất là tạo một tham chiếu giữa tệp trên đĩa (do tên tệp và đường dẫn tương ứng được hệ điều hành xác định) và biến tệp trong chương trình, làm cho biến tệp trở thành đại diện cho tệp. Biến tệp trở thành đối tượng trực tiếp trong chương trình để nhận các thao tác đối với tệp trên đĩa.

+ Câu lệnh sử dụng thủ tục *assign* gắn tên tệp với biến tệp.

assign(<*biến tệp*>, <*tên tệp*>);

Trong cú pháp, *tên tệp* là hằng xâu kí tự hoặc giá trị của một biểu thức kiểu xâu kí tự (để hệ điều hành có thể xác định được tệp). Tất cả các phép toán trên *biến tệp* sẽ tác động tới tệp *tên tệp*. Sau khi gọi thủ tục *assign*, sự liên kết giữa biến tệp và tệp chỉ kết thúc khi có lời gọi *assign* khác thực hiện cũng trên *biến tệp* này (nghĩa là lúc đó biến tệp được chuyển sang gắn với một tệp khác). *Tên tệp* có thể là đường dẫn chứa ổ đĩa, danh sách các thư mục liên tiếp cách nhau bởi dấu đường dẫn (\), cuối cùng là tên tệp:

<*ổ đĩa*>:\<*tên thư mục*>\<*tên thư mục*>\... \<*tên thư mục*>\<*tên tệp*>

Độ dài lớn nhất của tên tệp là 79 kí tự. Đặc biệt khi tên tệp là xâu rỗng (độ dài xâu bằng 0) thì biến tệp được gán cho các tệp vào/ra chuẩn. Các tệp vào/ra chuẩn được quy định tương ứng với thiết bị nào là tuỳ thuộc vào sự mở rộng của mỗi chương trình dịch Pascal, nhưng thường quy định tệp input chuẩn là bàn phím, tệp output chuẩn là màn hình.

b) Mở tệp

+ Câu lệnh sử dụng thủ tục *reset* mở tệp văn bản đã tồn tại để đọc dữ liệu.

Cú pháp:

reset(<*biến tệp*>);

Trong cú pháp, *biến tệp* phải đã được gắn với một tệp (dùng *assign*). Nếu tệp này không tồn tại thì thực hiện *reset* sẽ gặp lỗi. Nếu tệp đã mở thì nó sẽ đóng lại rồi sau đó mở lại. Vị trí con trỏ tệp sau lời gọi *reset* là đầu tệp.

Với học sinh khá, giáo viên có thể giới thiệu về dẫn hướng biên dịch {$I-} đặt trước lời gọi *reset* và hàm *IoResult*. Hàm *IoResult* sẽ trả lại giá trị 0 nếu thực hiện *reset* thành công, trái lại nó cho giá trị khác 0.

Ví dụ:

```
uses crt;
var f: text;
begin
 clrscr;
 assign(f,'songuyen.dat');
```

```
  {$I-} reset(f);
  if IoResult<>0 then
    begin
       write('Mo tep bi loi');
       readln;
       halt;
    end
  else write('Mo tep thanh cong');
  close(f);
  {$I+}
  readln
end.
```

+ Câu lệnh sử dụng thủ tục *rewrite* mở tệp để ghi dữ liệu.

Cú pháp:

rewrite(<*biến tệp*>);

Trong cú pháp, *biến tệp* cần phải đã được gắn với một tệp (dùng *assign*). *Rewrite* tạo ra một tệp mới với tên tệp đã gắn với biến tệp. Nếu đã có một tệp cùng tên (cùng thư mục) thì tệp này bị xoá và một tệp rỗng được tạo ra thay thế nó. Nếu tệp đã mở sẵn, nó sẽ được đóng lại sau đó được tạo lại là tệp rỗng.

Với học sinh khá, giáo viên có thể giới thiệu dẫn hướng biên dịch {$I-} đặt trước lời gọi *rewrite*. Hàm *IoResult* sẽ trả lại giá trị 0 nếu thực hiện *rewrite* thành công, trái lại nó cho giá trị khác 0.

c) Đọc/ghi tệp văn bản

Cú pháp đọc tệp văn bản:

read(< *biến tệp*>, <*danh sách biến*>);

hoặc

readln(<*biến tệp*>, <*danh sách biến*>);

Danh sách biến là dãy *tên biến 1*, *tên biến 2*, ..., *tên biến N*. Các dữ liệu cần đọc trong tệp gán vào danh sách biến phải lần lượt có kiểu tương ứng với kiểu của biến trong danh sách biến. Nếu sai kiểu thì chương trình mắc lỗi. Lỗi này thường gặp khi biến có kiểu số, dữ liệu đọc được lại là kiểu xâu.

Ví dụ, tệp THU.TXT chỉ có một dòng *abcdefghik*1234.

Xét chương trình

```
var f : text;
 s : string[9];
 x : longint;
begin
 assign(f,'THU.TXT');
 reset(f);
 read(f,s,x);
```

```
  writeln(s);
  writeln(x);
  readln
end.
```

Chương trình này mắc lỗi "*Invalid numeric format*" vì sau khi đọc được *s*='*abcdefghi*', tiếp theo đọc dữ liệu cho *x* thì mắc lỗi vì '*k*' không là dạng số. Nếu thay lại khai báo *s*:*string*[9] thành *s*:*string*[10] thì chương trình không mắc lỗi khi thực hiện đọc tệp, kết quả trên màn hình cho biết *s* = '*abcdefghik*' và *x* = 1234.

Với biến xâu, thủ tục *read* sẽ đọc các kí tự trên một dòng vào biến (loại trừ các kí tự đánh dấu hết dòng hoặc hết tệp). Số kí tự đọc vào biến xâu tối đa bằng độ dài đã khai báo của biến xâu.

Với biến kiểu nguyên hoặc thực, thủ tục *read* sẽ không đọc dấu cách, dấu tab hoặc dấu xuống dòng đứng trước xâu chữ số. Nếu xâu chữ số không phù hợp với kiểu của biến tương ứng thì xuất hiện lỗi vào/ra (I/O). Trong trường hợp ngược lại, giá trị kiểu số tương ứng của xâu chữ số sẽ được gán cho biến. Lệnh *read* tiếp theo sẽ được bắt đầu bằng dấu cách, dấu tab hoặc kí tự hết dòng và chúng cũng lại được bỏ qua. Những dấu này vạch định giới hạn cho các xâu chữ số.

Cú pháp ghi tệp văn bản:

write(<*biến tệp*>,<*danh sách kết quả*>);

hoặc

writeln(<*biến tệp*>,<*danh sách kết quả*>);

danh sách kết quả là dãy *kết quả 1*, *kết quả 2*, ..., *kết quả N*. Khi hai kết quả liền nhau cùng là kiểu số thì cần xen vào giữa hai kết quả số này một kết quả trung gian là hằng kí tự dấu cách. Ví dụ, *write*(*f*,*x*,' ',*y*); trong đó *f* là tên biến tệp, *x* và *y* là hai biểu thức số. Trước khi gọi thủ tục này, tệp tương ứng với biến tệp phải là đang mở. Thủ tục *write* sẽ ghi lần lượt các kết quả theo danh sách các kết quả vào tệp kể từ vị trí hiện thời của con trỏ tệp. Các giá trị được ghi vào tệp là các kết quả của các biểu thức có kiểu kí tự, nguyên, thực, lôgic hoặc xâu kí tự.

d) Đóng tệp

Cú pháp:

close(< *biến tệp*>);

Trong cú pháp, *biến tệp* đã được gắn với một tệp đang mở bằng *reset*, *rewrite* hoặc *append* (*append* chỉ dùng với tệp văn bản và không giới thiệu trong sách giáo khoa) ở thời điểm trước đó để mở tệp. Sau câu lệnh *close*, tệp gắn với biến tệp được hoàn thành cập nhật và sau đó được đóng lại, chương trình trả lại

quyền quản lí tệp cho hệ điều hành. Nếu thực hiện ghi dữ liệu vào tệp mà không đóng tệp thì không có dữ liệu nào được ghi hoặc chỉ ghi được một phần vào tệp, nguyên nhân do các dữ liệu chứa trong bộ nhớ đệm chưa chuyển kịp vào đĩa thì chương trình đã bị ngắt.

- Củng cố lại kiến thức cuối tiết. Sử dụng hình 16 trong sách giáo khoa Tin học 11 (giáo viên kẻ sẵn trên giấy khổ lớn) và câu hỏi trắc nghiệm.

C. Kiến thức bổ sung

Một số hàm và thủ tục chuẩn thường dùng trong thao tác tệp.

Hàm *EOF* trả lại giá trị cho biết con trỏ tệp đã ở vị trí cuối tệp hay chưa. Nếu con trỏ tệp ở vị trí cuối tệp thì hàm *EOF* trả lại giá trị *true*, ngược lại là giá trị *false*. Vị trí cuối tệp là vị trí ngay sau vị trí dữ liệu cuối cùng của tệp. Tại vị trí cuối tệp là một kí tự điều khiển (mã số 26).

Hàm *EOLN* trả lại giá trị cho biết con trỏ tệp đã ở vị trí cuối dòng chưa. Nếu con trỏ tệp ở vị trí cuối dòng thì hàm *EOLN* trả lại giá trị *true*, ngược lại là giá trị *false*. Vị trí cuối dòng là vị trí ngay sau vị trí dữ liệu cuối cùng của dòng. Tại vị trí cuối dòng là cặp kí tự điều khiển (mã số 13 và mã số 10).

Với học sinh khá, giáo viên có thể giới thiệu về hướng dẫn biên dịch {$I-} đặt trước lời gọi *Close*, *eof* hoặc *eoln*. Hàm *IoResult* sẽ trả về giá trị 0 nếu thực hiện thành công, trái lại nó trả về giá trị khác 0.