CHƯƠNG V. TỆP VÀ THAO TÁC VỚI TỆP

BÀI 14 : KIỂU DỮ LIỆU TỆP

1. **Vai trò của kiểu tệp**

* Kiểu dữ liệu tệp có những đặc điểm sau :
* Dữ liệu kiểu tệp được lưu trữ lâu dài ở bộ nhớ ngoài và không bị mất khi tắt nguồn điện.
* Lượng dữ liệu lưu trữ trên tệp có thể rất lớn và chỉ phụ thuộc vào dung lượng đĩa.

1. **Phân loại tệp và thao tác với tệp**

* Xét theo cách tổ chức dữ liệu, tệp có thể phân làm hai loại :
* **Tệp văn bản** là tệp mà dữ liệu được ghi dưới dạng các kí tự theo mã ASCII.

*Ví dụ* : sách, tài liệu, bài học, giáo án, các chương trình nguồn viết bằng ngôn ngữ bậc cao... thường được lưu trữ dưới dạng văn bản.

* **Tệp có cấu trúc** là tệp mà các thành phần của nó được tổ chức theo một cấu trúc nhất định.

*Ví dụ* : Hình ảnh, âm thanh. thường được lưu trữ dưới d ạng tệp có cấu trúc.

* Xét theo cách thức truy cập, tệp có thể phân làm hai loại :
* **Tệp truy cập tuần tự** là tệp cho phép truy cập đến một dữ liệu nào đó trong tệp chỉ bằng cách b ắt đầu từ đầu tệp và đi qua lần lượt tất cả các dữ liệu trước nó.
* **Tệp truy cập trực tiếp** là tệp cho phép tham chiếu đến dữ liệu cần truy cập bằng cách xác định trực tiếp vị trí (thường là số hiệu) của dữ liệu đó.
* Số lượng phần tử của tệp không cần xác định trước.

xxxxx

BÀI 15 : THAO TÁC VỚI TỆP

1. **Khai báo**

* Khai báo biến tệp có dạng :

**var**< *tên biến tệp>:* **text** ;

1. **Thao tác với tệp**
2. ***Gắn tên tệp***

* Để thao tác với tệp, thao tác trước hết phải gắn tên tệp với đại diện của nó là biến tệp bằng thủ tục :

**assign**(<*bến tệp>, <tên tệp>);*

Trong đó *<tên tệp>* là *biến xâu* hoặc *hằng xâu.*

1. ***Mở tệp***

* Trước khi mở tệp, biến tệp phải được gắn tên tệp bằng thủ tục **assign**.
* Câu lệnh dùng thủ tục mở tệp để ghi dữ liệu có dạng : **rewrite**(< *biến tệp>*);
* Khi thực hiện thủ tục mở tệp, nếu tệp đó chưa có thì tệp sẽ được tạo mới với nội dung rỗng. Nếu đã có, thì nội dung cũ của tệp sẽ bị xóa để chuẩn bị ghi dữ liệu mới.
* Trước khi đọc dữ liệu từ tệp đã gắn với một biến tệp, ta mở tệp bằng thủ tục :

**reset**(< *biến tệp>*);

1. ***Đọc/Ghi tệp văn bản***

* Câu lệnh dùng thủ tục đọc có dạng : **read**(< *biến tệp>, < danh sách biến>*); hoặc

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

* Câu lệnh dùng thủ tục ghi có dạng : **write**(< *biến tệp>*, < *danh sách kết quả>*); hoặc

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

* Một số hàm chuẩn thường dùng khi đọc/ghi tệp văn bản :
* Hàm **eof**(<*biến tệp>)* trả về giá trị **true** nếu con trỏ tệp đang chỉ đến cuối tệp.
* Hàm **eoln**(<*biến tệp>)* trả về giá trị **true** nếu con trỏ tệp đang chỉ đến cuối dòng.

1. ***Đóng tệp***

* Sau khi làm việc xong với tệp, ta cần phải đóng tệp để hệ thống thực sự hoàn tất việc ghi dữ liệu ra tệp.
* Câu lệnh dùng thủ tục đóng tệp có dạng : **close**(<*biến tệp>);*
* Ví dụ để đóng 2 tệp tepA, tepB ta viết như sau : **close**(*tepA*);

**close**(*tepB*);

xxxxx

BÀI 16 : VÍ DỤ LÀM VIỆC VỚI TỆP(tham khảo thêm SGK)

xxxxx

CHƯƠNG VI. CHƯƠNG TRÌNH CON VÀ LẬP TRÌNH CÓ  
CẤU TRÚC

BÀI 17 : CHƯƠNG TRÌNH CON VÀ PHÂN LOẠI

**1. Khái niệm chương trình con**

* Chương trình con cũng có thể được xây dựng từ các chương trình con khác.
* Phương pháp lập trình có cấu trúc là phương pháp viết chương trình dựa trên các chương trình con.
* Chương trình có cấu trúc là chương trình được xây dựng từ các chương trình con.

*Chương trình con là một dãy lệnh mô tả một số thao tác nhất định và có thể được thực hiện (được gọi) từ nhiều vị trí trong chương trình.*

❖ ***Lợi ích của việc sử dụng chương trình con***

o Tránh được việc phải viết lặp đi lặp lại cùng một dãy lệnh.

o Hổ trợ việc thực hiện các chương trình lớn.

o Phục vụ cho quá trình trừu tượng hóa.

o *Mở rộng khả năng ngôn ngữ.*

*o Thuận tiện cho phát triển, nâng cấp chương*

*trình.*

1. **Phân loại và cấu trúc của chương trình con**

***a) Phân loại*** : thường gồm hai loại :

* ***Hàm*** *(Function)* là chương trình con thực hiện một số thao tác nào đó và **trả về một giá trị qua tên của nó**.
* ***Thủ tục*** *(Procedure)* : là chương trình con thực hiện các thao tác nhất định nhưng **không trả về giá trị nào qua tên của nó.**

*Ví dụ* : các thủ tục vào/ra chuẩn hay thủ tục xử lí xâu :

***Write/n, readln, delete, insert, ...***

***b) Cấu trúc chương trình con***

*<phần đầu>*

[< *phần kha í báo>]*

*<phần thân>*

* ***Phần đầu*** : dùng để khai báo tên của chương trình con, các biến cho dữ liệu vào và ra, nếu là hàm phải khai báo kiểu dữ liệu cho giá trị trả về của hàm.
* ***Phần khai báo*** : khai báo các hằng và biến dùng trong chương trình con. Phần khai báo này có thể bỏ qua.
* ***Phần thân*** : là dãy câu lệnh thực hiện để từ những dữ liệu vào ta nhận được dữ liệu ra hay kết quả mong muốn.

***Ví dụ.*** ta có chương trình con (Hàm) tính lũy thừa **xk** sau :

**Function**LuyThua (x : real, k : integer) : real;

**Var** j : integer;

Tich : real;

**Begin**

Tich := 1.0;

**For** j := 1 **to** k **do** Tích := Tích \* x;

LuyThua := Tich;

**End**;

• *Phần đầu* :

• *Phần khai báo* :

**Var** j : integer;

khai báo các biến dùng trong hàm

**(2)**

Tich : real;

• *Phần thân* :

**Begin**

Tich := 1.0;

dãy câu lệnh thực hiện việc  
tính toán lũy thừa xk

**For** j := 1 **to** k **do** Tích := Tích \* X?

LuyThua := Tich;

**End**;

* ***Tham số hình thức*** : Các biến khai báo cho dữ liệu vào/ra được gọi là tham số hình thức của chương trình con. VD : **(1)** ở trên
* Một chương trình con có thể có hoặc không có tham số hình thức.
* ***Biến cục bộ*** : là các biến được khai báo để dùng riêng trong chương trình con. VD : **(2)** ở trên
* ***Biến toàn cục*** : là các biến của chương trình chính
* Chương trình chính và các chương trình con khác không

thể sử dụng được các *biến cục bộ* của một

chương trình con.

* Mọi chương trình con đều có thể sử dụng được các *bến toàn cục*
* Một chương trình con có thể có hoặc không có biến cục bộ

1. ***Thực hiện chương trình con***

*•* Để thực hiện (gọi) một chương trình con, ta cần phải có lệnh gọi nó, bao gồm tên chương trình con với các tham số là các hằng và biến tương ứng với các tham số hình thức (nếu có). Các hằng và biến này được gọi là các *tham số thực sự*

❖ ***Tham số thực sự*** : là các hằng và biến tương ứng  
với các tham số hình thức khi gọi chương trình con.

xxxxx

BÀI 18 : VÍ DỤ VỀ CÁCH VIẾT VÀ SỬ DỤNG CHƯƠNG TRÌNH CON

**1. Cách viết và sử dụng thủ tục**

***a) Cấu trúc của thủ tục***

**Procedure**< *tên thủ tục>[<danh sách tham số>];*

[<*phần khai báo]*

**Begin**

*[<dãy các lệnh>]*

**End**;

* ***Phần đầu*** thủ tục gồm tên dành riêng ***procedure****,* tiếp theo là tên của thủ tục. Danh sách các tham số có thể có hoặc không có.
* ***Phần khai báo*** dùng để xác định các hằng, kiểu, biến và cũng có thể xác định các chương trình con khác được sử dụng trong thủ tục.
* ***Phần thân*** là dãy câu lệnh được viết giữa cặp tên dành riêng ***begin*** và ***end***.

***b) Ví dụ về thủ tục***

***Ví dụ 1*** :

Chương trình sau đây chứa thủ tục Ve\_Hcn với các tham số chdai, chrong và sử dụng thủ tục này để vẽ các hình chữ nhật có kích thước khác nhau.

1. **Cách viết và sử dụng hàm**

• Hàm có cấu trúc tương tự thủ tục, tuy nhiên có khác nhau ở phần đầu. Khai báo phần đầu của hàm như sau :

**function***<tên hàm>* [(< *danh sách tham sơ*>)] : *<kiểu  
dữ Hệu>;*

mở đầu là tên dành riêng **function**

kết thúc là kiểu dữ liệu của giá trị mà hàm trả về và chỉ có thể là các kiểu ***integer, real, char, boolean, string.***

*•* Điểm khác nhau cơ bản giữa thủ tục và hàm là việc thực hiện hàm luôn trả về giá trị kết quả thuộc kiểu xác định và giá trị đó được gán cho tên hàm. Trong thân hàm cần có lệnh gán giá trị cho tên hàm :

< *tên hàm>* := < *biểu  
thức>;*

***Ví dụ 1:***

Chương trình sau thực hiện việc rút gọn một phân số, trong đó có sử dụng hàm tính ước chung lớn nhất (UCLN) của hai số nguyên.

❖ ***Sử dụng hàm***

* Lệnh gọi hàm gồm tên hàm và các tham số thực sự tương ứng với các tham số hình thức.
* Lệnh gọi hàm có thể tham gia vào biểu thức như một toán hạng và thậm chí là tham số của lời gọi hàm, thủ tục khác.

Ví dụ : A := 6 \* UCLN(Tuso, Mauso) +1;

lời gọi hàm UCLN

BÀI 19 : THƯ VIỆN CHƯƠNG TRÌNH CON CHUẨN