§10. CẤU TRÚC LẶP

A. Mục đích, yêu cầu

- Hiểu nhu cầu của cấu trúc lặp trong biểu diễn thuật toán;
- Hiểu cấu trúc lặp với số lần biết trước, cấu trúc lặp kiểm tra điều kiện trước;
- Biết cách vận dụng đúng đắn từng loại cấu trúc lặp vào tình huống cụ thể;
- Mô tả được thuật toán của một số bài toán đơn giản có sử dụng lệnh lặp;
- Viết đúng các lệnh lặp với số lần biết trước, lệnh lặp kiểm tra điều kiện trước;
- Viết được thuật toán của một số bài toán đơn giản.

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

§10 được dạy trong ba tiết (tiết 2, 3 và 4) và có thể phân phối như sau:

+ Tiết 2: gồm mục 1 và mục 2 (phần lí thuyết).

+ Tiết 3: mục 2 (phần ví dụ, luyện tập).

+ Tiết 4: mục 3.

1. Một số chú ý về nội dung

- *Về cấu trúc lặp*

Ý nghĩa của cấu trúc lặp: Cấu trúc lặp là điều khiển thực hiện công việc lặp đi lặp lại khi chưa đủ số lần lặp hoặc khi một điều kiện nào đó còn đúng. Giáo viên cần nêu rõ ý nghĩa của cấu trúc điều khiển lặp thông qua ví dụ nêu trong sách giáo khoa. Ngoài hai bài toán 1 và 2 trong mục 1, có thể chọn ví dụ khác dẫn từ thực tế.

Ví dụ, chương trình tính điểm cho học sinh một lớp (giả sử lớp có 50 học sinh) sẽ phải lặp lại một số thao tác như sau với mỗi học sinh:

+ Nhập họ tên học sinh (hoặc số báo danh);
+ Nhập điểm (các môn);
+ Tính tổng điểm hoặc tính điểm trung bình.

Nếu chương trình viết đi viết lại các câu lệnh thực hiện những thao tác trên thì rõ ràng là không thuận tiện dễ dẫn tới nhàm chán. Vì thế ngôn ngữ lập trình đưa ra cấu trúc điều khiển lặp, chỉ cần viết các câu lệnh thực hiện các thao tác cho một học sinh sau đó cho phép chương trình tự thực hiện lặp lại các thao tác đó với các học sinh khác. Quá trình lặp sẽ dừng khi đã lặp đủ một số lần nào đó, ví dụ như đã lặp đủ 50 lần (nhập đủ điểm cho 50 học sinh) hoặc khi gặp một điều kiện nào đó, ví dụ như điều kiện người nhập điểm nhấn phím **Enter** khi cần nhập số báo danh chẳng hạn (khi đó có thể chưa nhập đủ điểm cho 50 học sinh nhưng người nhập điểm bận việc khác không thể tiếp tục công việc nhập điểm được nữa). Qua ví dụ trên có thể gợi ý cho học sinh thấy rõ có hai loại lặp: lặp với số lần biết trước (nhập đủ 50 lần) và lặp với số lần chưa biết trước (nhập cho đến khi nhấn phím **Enter** thay cho nhập số báo danh).

Quá trình lặp không thể dừng được gọi là quá trình lặp vô hạn. Điều này xảy ra khi điều kiện để dừng lặp không còn bị biến đổi giá trị sau mỗi lần lặp. Khi đó để thoát lặp vô hạn, cần có các câu lệnh cho phép thoát ngay khỏi lặp.

- *Lặp với số lần biết trước*

Dạng lặp với số lần biết trước dùng để thực hiện *câu lệnh* một số lần xác định. Dạng này dùng một *biến điều khiển* để điều khiển vòng lặp. Trong Pascal mỗi lần thực hiện *câu lệnh* thì *biến điều khiển* (giả sử là *i*) được tự động tăng (nhận giá trị tiếp theo là *succ*(*i*)) hoặc giảm (nhận giá trị nhỏ hơn ngay trước là *pred*(*i*)). Đến khi *biến điều khiển* đạt giá trị xác định thì vòng lặp kết thúc.

Trong Pascal, lặp với số lần biết trước thể hiện qua câu lệnh *for-do* và có hai dạng tiến và lùi với cú pháp như trình bày trong sách giáo khoa.

Lưu ý: Trong vòng lặp không được chứa lệnh làm thay đổi giá trị của *biến điều khiển* vì sẽ gây ra tình trạng khó theo dõi và quản lí vòng *for-do*. Mặc dù, với Turbo Pascal, trong vòng lặp ta có thể gán giá trị của biến điều khiển bằng giá trị cuối của vòng *for* tiến (hoặc giá trị đầu của vòng *for* lùi) để thoát ngay vòng lặp, song giáo viên cần giải thích việc làm này là không hợp lệ. Trong Free Pascal, việc gán giá trị cho biến điều khiển vòng *for* bên trong vòng lặp sẽ bị thông báo mắc lỗi khi biên dịch. Việc thay đổi giá trị của biến điều khiển ngay trong vòng lặp là không hợp lí vì can thiệp vào cấu trúc vòng *for*, có thể so với hình ảnh người đi đường đi sai phần đường, đôi khi vẫn đi được nhưng là phạm luật và có thể tai nạn nếu không thận trọng.

Trong thực hành, với học sinh khá giáo viên có thể giới thiệu thêm về hai vòng *for* lồng nhau: nếu *câu lệnh* sau *do* của vòng *for-do* thứ nhất là một câu lệnh *for-do* thứ hai thì ta có hai vòng *for-do* lồng nhau. Vòng *for-do* thứ nhất được gọi là vòng *for* ngoài, vòng *for-do* thứ hai được gọi là vòng *for* trong. Các biến đếm của hai vòng *for* lồng nhau không được trùng tên.

Các *giá trị đầu*, *giá trị cuối* có thể là những biểu thức. Trong trường hợp này, giá trị của các biểu thức được tính trước khi vào vòng lặp và làm nhiệm vụ *giá trị đầu* và *giá trị cuối*. Do đó, trong vòng lặp nếu có biến nào nằm trong hai biểu thức này bị thay đổi giá trị thì *giá trị đầu* và *giá trị cuối* vẫn không thay đổi.

Giá trị của *biến điều khiển* sau khi thực hiện xong lệnh *for-do* là không xác định (chưa chắc là giá trị tiếp theo của *giá trị cuối* với dạng tiến và cũng không chắc là giá trị trước của *giá trị đầu* với dạng lùi). Vì vậy, không được sử dụng giá trị này vào các câu lệnh tiếp theo.

- *Lặp với số lần chưa biết trước*

Lặp với số lần chưa biết trước có hai dạng:

Dạng 1: Trong khi <*điều kiện*> còn đúng thì còn thực hiện <*công việc*>;

Dạng 2: Thực hiện <*công việc*> trong khi <*điều kiện*> đúng.

Trong dạng 1, đầu tiên kiểm tra và tính giá trị của *điều kiện*, nếu *điều kiện* nhận giá trị *true* thì thực hiện *công việc (một lần)*. Mỗi lần thực hiện *công việc* có thể sẽ làm thay đổi giá trị của *điều kiện* nên đến một lúc nào đó điều kiện lặp không còn đúng nữa và cấu trúc lặp sẽ được kết thúc. Ngược lại, nếu khi thực hiện *công việc* không làm thay đổi giá trị của *điều kiện* thì cấu trúc lặp kéo dài mãi (gọi là *vòng lặp vô hạn*). Để thoát khỏi vòng lặp vô hạn, trong *công việc* cần có câu lệnh rẽ nhánh thoát khỏi vòng lặp vô hạn khi thoả mãn điều kiện rẽ nhánh.

Trong dạng 2, đầu tiên *công việc* được thực hiện, sau đó *điều kiện* được kiểm tra, nếu *điều kiện* còn đúng thì tiếp tục thực hiện *công việc*, ngược lại việc thực hiện lặp kết thúc. Trong *công việc* cần có câu lệnh làm thay đổi *điều kiện* hoặc câu lệnh thoát khỏi vòng lặp. Để giảm tải, trong sách giáo khoa chỉ giới thiệu dạng 1.

Trong Pascal, một dạng chủ yếu trong hai dạng lặp với số lần chưa biết trước là dạng *while-do*.

Câu lệnh *while-do* chứa một biểu thức điều kiện để điều khiển thực hiện lặp một câu lệnh đơn hoặc ghép.

Cú pháp:

while <*điều kiện*> **do** <*câu lệnh*>;

Câu lệnh viết sau từ khoá *do* được thực hiện khi biểu thức *điều kiện* còn nhận giá trị *true*. Biểu thức *điều kiện* được tính giá trị trước khi *câu lệnh* được thực hiện, nhưng nếu biểu thức *điều kiện* đã nhận giá trị *false* ngay từ đầu thì *câu lệnh* không được thực hiện lần nào. Nếu biểu thức *điều kiện* luôn nhận giá trị *true* thì *câu lệnh* được thực hiện mãi, ta gọi là vòng lặp vô hạn.

2. Một số gợi ý về tổ chức dạy học

- Từ các ví dụ trong sách giáo khoa, bằng ngôn ngữ tự nhiên giáo viên nên trình bày cách giải những bài toán này. Từ đó, dẫn dắt học sinh tới các khái niệm về vòng lặp với số lần biết trước và không biết trước. Giáo viên trình bày cú pháp các dạng lặp và sơ đồ thuật toán (nên chuẩn bị minh hoạ trên giấy khổ lớn) của các bài toán minh hoạ.
- Dành nhiều thời gian cho học sinh đọc và tìm hiểu các lệnh và cấu trúc điều khiển trong các chương trình giải các bài toán đã trình bày trong sách giáo khoa, đối chiếu câu lệnh trong chương trình với các bước trong dạng biểu diễn thuật toán bằng liệt kê.
- Giáo viên cần dẫn giải chú thích một số câu lệnh chủ yếu trong chương trình bằng ngôn ngữ tự nhiên.
- Về cấu trúc *for-do* nên có minh hoạ hai thuật toán *Tong_1a* và *Tong_1b* trên giấy khổ lớn (diễn đạt kiểu liệt kê các bước, không dùng sơ đồ khối).

Giải thích thuật toán *Tong_1a* để học sinh hiểu giá trị biến điều khiển N cho biết số lần lặp. Khi $1 \leq N \leq 100$ thì thực hiện lặp. Sau khi lặp lần thứ 100 thì kết thúc lặp.

Trong chương trình Pascal cài đặt thuật toán *Tong_1a*, câu lệnh lặp

```
for N:= 1 to 100 do
    S:= S+1.0/(a+N);
```

đã bao hàm cả ba bước 2, 3, 4 của thuật toán.

Giáo viên không nên sử dụng sơ đồ khối biểu diễn thuật toán này vì lệnh gán $N := N+1$ rất khó thể hiện đúng trong sơ đồ (nó không là lệnh nằm trong chương trình, mà là lệnh thuộc cấu trúc của vòng *for*).

Sau khi học sinh đã nắm được thuật toán *Tong_1a*, hãy giới thiệu thuật toán *Tong_1b* bằng gợi ý so sánh với thuật toán *Tong_1a*.

Cuối cùng có thể đưa ra cú pháp của hai dạng *for* (tiến và lùi).

Nên liên hệ cấu trúc *while-do* với *for-do* tương ứng. Mọi vòng lặp *for-do* đều có thể chuyển về *while-do*, giáo viên có thể hướng dẫn học sinh giải quyết vấn đề này trong giờ bài tập. Giáo viên cũng cần nhắc học sinh lưu ý trong vòng lặp với số lần chưa biết trước phải có câu lệnh làm thay đổi giá trị của biểu thức kiểm tra điều kiện lặp.

Về thiết bị dạy học: Một số minh hoạ chuẩn bị sẵn theo nội dung nêu trên. Nếu nhà trường có máy chiếu, có thể chuẩn bị các slide (bằng phần mềm Microsoft PowerPoint) để trình chiếu hoặc chạy thử từng đoạn chương trình để cả lớp cùng theo dõi.

C. Kiến thức bổ sung

- Trong ứng dụng thực tế có thể sử dụng câu lệnh lặp vô hạn như các ví dụ dưới đây:

– Kiểm tra dữ liệu nhập từ bàn phím:

Bài toán: Nhập giá trị *N* nguyên dương nhỏ hơn 100.

```
write('nhap gia tri N nguyen duong nho hon 100: ');
while TRUE do
 begin
   readln(N);
   if (N>0) and (N<100) then break;
 end;
```

– Chứng minh khẳng định sau là sai: "Số Mec-xen $2^p - 1$ (với số *p* nguyên tố) là nguyên tố".

```
p:= 2;
while TRUE do
   begin
      if <p nguyên tố> then
         if < 2^p - 1 không nguyên tố> then break;
      Inc(p);
   end;
```

- Tạo bảng chọn chức năng công việc mà người dùng có thể tuỳ ý quyết định kết thúc vòng lặp chọn công việc vào bất cứ lúc nào.

```
while true do
begin
   write('chon cong viec Nhap (A)-Sua (B)-Thoat (Q) ');
   while true do
      begin
         readln(ch);
         if ch in ['A', 'B', 'Q'] then break;
      end;
if ch='A' then <gọi chương trình Nhập dữ liệu>
else
   if ch='B' then <gọi chương trình Sửa dữ liệu>
   else break;
end;
```

- Trong Pascal, nếu vòng lặp *while-do* bị lặp vô hạn (do không có câu lệnh thoát vòng lặp: **if** <*điều kiện* 2> **then break;** hoặc **if** <*điều kiện* 2> **then exit;**...) thì phải dùng tổ hợp phím **Ctrl+Break** sau đó nhấn phím **F9**, hoặc ngắt chương trình Pascal đang chạy bằng cách dùng tổ hợp phím **Ctrl+Alt+Del**,...

  Ví dụ về sử dụng vòng lặp vô hạn:

  Cho số thực *a*. Tìm số nguyên dương *N* nhỏ nhất thoả mãn:

$$1+\frac{1}{2}+\frac{1}{3}+...+\frac{1}{N}>a\,.$$

Chương trình:

```
uses crt;
var a,s: real;
n: longint;
begin
   clrscr;
   write('Nhap gia tri a= '); readln(a);
   s:= 1; n:= 1;
   while true do {lặp vô hạn vì điều kiện là hằng true}
      begin
         if s>a then break; {rẽ nhánh để thoát lặp vô hạn}
         inc(n);
         s:= s + 1/n;
      end;
```

```
    write(n);
    readln
end.
```

Bài toán trên có thể giải bằng chương trình không dùng vòng lặp vô hạn:

```
s:= 1; n:= 1;
while s<=a do
    begin
        inc(n);
        s:= s + 1/n;
    end;
```

Nếu nhập *a* = 10 sẽ tìm được *n* = 12367.

- Cú pháp câu lệnh lặp trong các ngôn ngữ lập trình khác nhau có thể khác nhau. Thậm chí nội dung cũng có thể được mở rộng khác nhau.

Ví dụ vòng lặp với số lần biết trước *for* trong C/C++ có cú pháp là:

for (*<câu lệnh khởi đầu>*; *<biểu thức 1>*; *<biểu thức 2>*) *<câu lệnh>*;

Có thể mô hình hoá vòng lặp *for* trên đây trong C/C++ bằng vòng lặp *while* như sau:

<câu lệnh khởi đầu>;
while (*<biểu thức 1>*)
{
<câu lệnh>;
<biểu thức 2>;
}

Câu lệnh khởi đầu có thể là câu lệnh ghép, gồm nhiều câu lệnh đơn khởi trị các giá trị đầu của các biến điều khiển vòng *for*. *Biểu thức 1* là biểu thức có chứa các biến điều khiển. Khi *biểu thức 1* còn nhận giá trị đúng (số khác 0) thì còn lặp: thực hiện *câu lệnh* và sau đó thay đổi giá trị các biến điều khiển trong *biểu thức* 2.

- Ví dụ về các vòng *for* lồng nhau.

Lập trình giải bài toán cổ sau:

Trăm trâu trăm cỏ
Trâu đứng ăn năm
Trâu nằm ăn ba
Lụ khụ trâu già
Ba con một bó.
Hỏi số trâu mỗi loại ?

Chương trình

```
uses crt;
var dung, nam, gia: byte;
begin
 clrscr;
 for dung:=1 to 20 do  {so trau dung tu 1 den 20}
   for nam:=1 to 33 do {so trau nam tu 1 den 33}
    begin
      gia:= 100-dung-nam; {so trau gia }
      if gia mod 3 = 0 then {dieu kien voi so trau gia}
         if dung*5+nam*3+(gia div 3) = 100 then {100 bo co}
      writeln('dung: ',dung,' nam: ',nam,' gia: ',gia);
   end;
 readln
end.
```