DẠY CÂU LỆNH LẶP VỚI SỐ LẦN BIẾT TRƯỚC TRONG NGÔN NGỮ LẬP TRÌNH PASCAL

PGS.TS. LÊ KHẮC THÀNH; Th.S. NGUYỄN CHÍ TRUNG
Đại học Sư Phạm Hà Nội

1. Hình thành câu lệnh

Tình huống dẫn đến nhu cầu sử dụng câu lệnh lặp **for** được chúng tôi cân nhắc lựa chọn là *Bài toán tính tổng* được phát biểu ngắn gọn như sau: *"Hãy lập trình cho máy tính giá trị của biểu thức :*

S = 2 + 3 + 4 + . . . + 21 + 22 + 23."

Dễ thấy tổng S có 11 cặp số hạng mà mỗi cặp số hạng cách đều ở hai đầu dãy luôn có tổng bằng 25. Suy ra tổng S cần tính là 25 × 11 = 275 (nhân nhẩm 25 với 11 bằng cách cộng 2 với 5 được 7 rồi viết vào giữa 2 và 5). Cách tính nhẩm này giúp HS (học sinh) dễ dàng kiểm tra được kết quả chạy chương trình trên máy. Nếu ta cho biểu thức cồng kềnh thì các em sẽ sa vào tính toán kiểm chứng và lãng phí thời gian không cần thiết, vì một tiết học chỉ có 45 phút.

Ngoài lời giải toán học thuần túy trên, có một cách tính khác, đó là cộng lần lượt các số hạng của biểu thức theo thứ tự từ trái qua phải. Để hiểu cách tính này, GV mô tả cho HS quá trình cộng dồn giá trị của biến *K* vào cho biến tổng *S* theo thuật toán sau đây:

Bước 1: Khởi tạo *S := 0; K := 2;*

Bước 2: Gán *S := S + K;*

Bước 3: Nếu K < 23 thì gán K := K + 1 và quay về bước 2.

Bước 3: Đưa ra giá trị của *S*;

Bước 4: Kết thúc thuật toán.

Thuật toán trên được GV diễn tả qua bảng mô phỏng sau đây:

Lần lặp	*K*	*S + K*	*S*
			0
1	2	0 + 2	2
2	3	2 + 3	5
3	4	5 + 4	9
...	...	...	...
21	22	...	252
22	23	252 + 23	275

Từ đây, GV giới thiệu cho HS biết rằng, để thực hiện lặp lại 22 lần công việc tính tổng *S + K* rồi gán cho biến *S*, ta có thể sử dụng câu lệnh lặp **for** trong ngôn ngữ Pascal với cú pháp như sau:

Cú pháp:

for biến := <BT 1> **to** <BT 2> **do**

<câu lệnh>;

Hoạt động : Từ <BT 1> (biểu thức 1) đến <BT 2> (biểu thức 2), biến nhận bao nhiêu giá trị thì câu lệnh được thực hiện bấy nhiêu lần.

Nhìn vào bảng mô phỏng trên, ta thấy: Việc thêm dần các giá trị kiểu byte của *K* vào *S* làm cho *S* lần lượt nhận các giá trị 2, 5, 9, 14, 20... cũng thuộc kiểu byte, đến khi *K* bằng 22 thì giá trị 252 của tổng *S* vẫn thuộc kiểu byte. Nhưng ở lần lặp cuối cùng, *K* bằng 23, giá trị của *S* là 275 vượt ra ngoài phạm vi kiểu byte và đòi hỏi phải được mô tả bằng kiểu word. Các phân tích trên dẫn đến chương trình như sau:

```
program ViDu ; uses crt;
var     K : byte ; S : word ;
begin
clrscr; S := 0 ;
for K := 2 to 23 do S := S + K ;
writeln ( ' Dap so la ' , S ) ;
readln
end.
```

Ta lưu ý với HS việc khởi trị cho *S* là cần thiết, và để không làm ảnh hưởng đến tổng, ta cho *S* khởi trị bằng 0 thông qua lệnh gán S := 0;

2. Luyện tập câu lệnh lặp với số lần định trước

Sau khi hình thành cho HS câu lệnh lặp **for**, để luyện tập, ta có thể cho HS giải quyết *Bài toán tính căn* với tình huống khó khăn hơn một chút so với

tình huống *Bài toán tính tổng* nói trên: *"Hãy lập trình tính giá trị biểu thức sau*:

$$S = \sqrt{23 + \sqrt{22 + ... + \sqrt{3 + \sqrt{2}}}}."$$

Bài toán này có 22 dấu căn bậc hai lồng nhau. Ý tưởng của thuật toán là việc tính căn của các tổng được thực hiện từ trong ra ngoài. Một cách tổng quát, ta dùng một biến *K* lần lượt nhận các giá trị từ 2 đến 23, ứng với mỗi giá trị của *K*, ta lấy $\sqrt{S+K}$ gửi vào *S* (Biến *S* được khởi trị bằng 0).

Ta dán chương trình trên sang cửa sổ thứ hai để thu được một chương trình mới bằng cách sửa một số chỗ. Chương trình thể hiện ý tưởng thuật toán trên như sau.

```
program ViDu ; uses crt;
var  K : byte ; S : real ;
begin
clrscr;  S := 0 ;
for  K := 2 to 23 do S := sqrt(S + K) ;
writeln(' Dap so la ', S : 0 : 4 ) ;
readln;
end.
```

Lời bàn về phương pháp giảng dạy

Về phương diện giao tiếp, khi tạo tình huống dạy câu lệnh lặp **for**, ta cần một biến lần lượt duyệt qua các giá trị *liên tục* và *đếm được* để điều khiển vòng lặp với số lần lặp biết trước. Tại thời điểm này, ta lưu ý cho HS mối quan hệ về giá trị giữa <BT 1> và <BT 2> để biết được <câu lệnh> sau từ khóa **do** thực hiện bao nhiêu lần.

Mục đích chức năng của ta là hình thành vòng **for** cho học sinh, *mục đích phương tiện* là tính tổng các số tự nhiên từ 2 đến 23. Phương châm của ta là thực hiện mục đích chức năng thông qua mục đích phương tiện. Vì vậy mà bài toán minh hoạ (mục đích phương tiện) ta chọn càng đơn giản về mặt tính toán, càng dễ hiểu về mặt cấu trúc thì càng tốt. Việc tính tổng từ 2 đến 23 chỉ là phương tiện trực quan minh hoạ cho HS dễ nắm bắt được cú pháp và hoạt động của câu lệnh lặp **for**. Sau khi HS đã hiểu biết về vòng **for** thông qua ví dụ đơn giản ban đầu, ta cho học sinh vận dụng vào tính biểu thức có 22 dấu căn bậc hai lồng nhau để HS thấy tầm quan trọng của vòng **for**.

Về phương diện dữ liệu, trong *Bài toán tính tổng*, dãy cần tính tổng là một dãy số liên tục, được chủ ý chọn cận dưới là 2 và cận trên là 23 chứ không chọn cận dưới và cận trên nhỏ hơn hoặc lớn hơn vì nhiều lí do, trong đó phải kể đến các lí do sau đây:

- Do tổng các số tự nhiên từ 2 đến 23 là 275 vượt ra ngoài 255 nên ta phải khai báo biến *S* có kiểu word. Về phương diện triết học, điều này minh họa quy luật *"lượng đổi, chất đổi"*. Quan điểm triết học này sẽ không có cơ hội được đề cập đến, nếu ta chọn dãy từ 1 đến 22, vì kết quả tổng *S* vẫn thuộc kiểu byte.

- Nếu ta cho tính tổng từ 1 đến 100 thì cũng dễ dàng nhẩm ra tổng là 5050 bằng công thức $(1 + 100) \times 50$ (cách mà Gau-xơ đã làm khi ông mới 6 tuổi). Nhưng vì tổng này quá dễ nhận ra vượt quá 255 nên sự cảnh báo việc tràn ô nhớ sẽ không mấy ngạc nhiên với các em. Độ giật mình không cao vì các em đã chuẩn bị từ trước, không mất cảnh giác như 275.

Về phương diện thuật toán, ta đã hướng cho HS xây dựng hai thuật toán giải *Bài toán tính tổng*. Thuật toán thứ nhất (tính theo công thức $(2 + 23) \times 11 = 275$), đương nhiên là sáng tạo, thời gian thực hiện nhanh. Nhưng thuật toán thứ hai (cộng lần lượt theo thứ tự tăng dần từ trái sang phải) là phong cách làm việc của máy tính, đặc tả các công việc được lặp lại với số lần biết trước.

Về phương diện quá trình, việc ta dùng bảng mô phỏng quá trình cộng dồn theo *K* như trên chính là giúp HS hình dung ra quá trình xảy ra bên trong bộ nhớ của máy tính khi nó thực hiện việc tính tổng qua câu lệnh **for**. Một số dòng trong bảng mô phỏng, ta để dấu ba chấm để cho HS tư duy tương tự. Việc tách dãy số thành các số độc lập là tư duy phân tích.

Từ chương trình tính tổng sang chương trình tính căn bậc hai là nhằm những ý tưởng sau đây:

Rèn luyện phương diện giải quyết vấn đề. Hai bài toán *Bài toán tính tổng* và *Bài toán tính căn* khác nhau về ý nghĩa toán học nhưng quy trình tiếp cận giải quyết từng công đoạn là tương tự như nhau. Đầu tiên gán cho biến lưu giữ kết quả bằng 0. Cho biến *K* lần lượt duyệt qua các giá trị từ giá trị ban đầu đến giá trị kết thúc. Ở *Bài toán tính tổng* thì lấy $S + K$ rồi gán cho biến *S*, ở *Bài toán tính căn* thì lấy $\sqrt{S+K}$ rồi gán cho biến *S*.

Rèn kĩ thuật lập trình từ dưới lên. HS biết tận dụng những chương trình đã biết để sửa chữa và tạo ra chương trình giải bài toán mới. Với chủ ý này, trong chương trình mới, ta giữ nguyên cận dưới 2 và cận trên 23; giữ nguyên ngữ cảnh của câu lệnh đưa thông tin ra; khai báo biến *S* độc lập để sửa kiểu dữ liệu của biến *S* từ word sang real; còn thân chương trình ta chỉ cần thêm hàm sqrt và thêm chỉ thị :0:4 để in ra số thực dấu phẩy tĩnh ở đáp số.

Nhắc nhở HS *phương diện sử dụng máy tính điện tử*: Rèn luyện cho HS các thao tác cơ bản về quản lí tệp và soạn thảo, bằng các công việc sao chép và dán chương trình sang cửa sổ khác, ghi lại tệp với tên mới, sửa lại bằng cách thêm, bớt, chèn, xoá để được chương trình tính căn.

Bồi dưỡng quan điểm triết học. Như đã đề cập ở trên, ta chủ ý chọn các cận cho dãy số cần tính tổng là 2 và 23 để thấy được giá trị của biến tổng *S* chuyển từ miền byte sang miền word, qua đó thấy được quy luật của cặp phạm trù chất và lượng. Mặt khác, việc chuyển từ bài toán tính tổng sang bài toán tính căn bậc hai đã làm cho biến *S* chuyển từ miền word sang miền real, qua đó thấy được quy luật của cặp phạm trù hình thức và nội dung.

3. DownTo

Sau khi HS đã nắm được hoạt động của câu lệnh lặp **for** dạng tiến, ta hình thành cho các em câu lệnh lặp **for** dạng lùi. Ta xét lại *Bài toán tính tổng* (không nên lấy ví dụ khác).

Theo tính chất giao hoán của phép cộng, tổng trên còn có thể tính theo thứ tự từ phải qua trái, tức là $S = 23 + 22 + 21 + \ldots + 4 + 3 + 2$. Để thực hiện điều này, ngôn ngữ Pascal có câu lệnh **for** dạng lùi với cú pháp và hoạt động như sau:

Cú pháp:

for biến := <BT 1> **downto** <BT 2> **do**
<câu lệnh>;

Hoạt động: Từ <BT 1> đến <BT 2> biến nhận bao nhiêu giá trị thì câu lệnh được thực hiện bấy nhiêu lần.

Ta dán chương trình tính tổng ở cửa sổ thứ nhất sang cửa sổ thứ ba và sửa lại như sau:

```
program ViDu ; uses crt;
var   K : byte ; S : word ;
begin
    clrscr;  S := 23 ;
for K := 22 downto 2 do S := S + K ;
  writeln ( ' Dap so la  ', S ) ;
  readln
    end.
```

Ở chương trình trên, ta cũng lưu ý cho HS cách khởi trị "sớm" cho biến tích lũy tổng *S* bằng câu lệnh gán S := 23. Khi đó trong câu lệnh **for**, biến *K* chỉ cần lùi từ giá trị 22 xuống giá trị 2, do đó số lần lặp giảm xuống còn 21 lần.

Để HS hứng thú và thấy tác dụng của câu lệnh lặp **for** dạng lùi, GV tiếp tục cho các em xem máy giải bài toán tính *S* sau đây:

Tính $S = \sqrt{2 + \sqrt{3 + \ldots + \sqrt{22 + \sqrt{23}}}}$

Ý tưởng tính toán cho bài toán này cũng là lần lượt tính từ trong ra ngoài tương tự như bài toán tính căn đã giải ở trên. Việc dùng **for** dạng lùi ở đây là thích hợp. Ta dán chương trình ở cửa sổ thứ hai sang cửa sổ thứ tư, rồi sửa lại để có chương trình mới:

```
program ViDu ; uses crt;
var   K : byte ; S :  real ;
begin
clrscr;  S := sqrt ( 23 ) ;
for K := 22 downto 2 do
   S := sqrt(S + K) ;
writeln ( ' Dap so la  ', S : 0 : 4 ) ;
readln
end.
```

4. Hai vòng for lồng nhau

Dạy câu lệnh lặp **for** cần phải tiến đến dạy hoạt động của 2 câu lệnh lặp **for** lồng nhau. Vì sau này HS sẽ dùng đến cấu trúc này khi làm việc với mảng 2 chiều và trong nhiều tình huống khác.

Cách tiếp cận cấu trúc **for** lồng nhau một cách tự nhiên nhất là ta đưa ra nhận xét: Trong cú pháp của câu lệnh lặp **for** thì bản thân <câu lệnh> sau từ khóa **do** cũng có thể là câu lệnh có cấu trúc như câu lệnh hợp thành (**begin .. end**), câu lệnh rẽ nhánh (**if-then**) và tất nhiên có thể là câu lệnh **for-do** khác. Hoạt động của **for** lồng nhau có thể hiểu như sau: ứng với từng giá trị của biến điều

khiển vòng **for** bên ngoài, máy sẽ thực hiện một lần vòng **for** bên trong.

Ta minh họa 2 vòng **for** lồng nhau bằng việc cho HS quan sát kết quả khi thực hiện chương trình cho hiện lên màn hình 5 dòng đầu của bảng nhân. Chương trình như sau:

```
program ViDu ; uses crt;
var   d , c : byte ;
begin
clrscr;
for  d := 1 to 5  do
begin
for  c := 1 to 9  do
write (d : 3,' . ',c,' = ', d*c : 2 ) ;
writeln;
end;
        readn
    end.
```

Nhận xét

Ta cho in 5 dòng đầu tiên của bảng nhân mà không cho in cả bảng nhân là để cho HS dễ nhận ra dòng chỉ nhận 5 giá trị, còn cột nhận 9 giá trị. Nếu in cả bảng nhân thì dòng và cột đều nhận 9 giá trị làm cho việc phân biệt giá trị của dòng và giá trị của cột là không trực quan.

Vận dụng: Ta có thể yêu cầu HS lập trình cho máy giải bài toán *"Có 100 con trâu, 100 bó cỏ. Trâu đứng ăn 5, trâu nằm ăn 3, lụ khụ trâu già 3 con một bó. Hỏi mỗi loại có mấy con?"*

Ta gọi số trâu đứng là *d*, số trâu nằm là *n*. Khi đó *d* lần lượt duyệt qua các giá trị từ 0 đến 20 (100 bó cỏ chia cho 5 là số bó một con trâu đứng ăn), ứng với mỗi giá trị của *d*, ta cho trâu nằm *n* lần lượt duyệt qua các giá trị từ 0 đến 33. Biết *d*, *n* ta sẽ tính được số trâu già là 100 - *d* - *n*. Chương trình như sau:

```
program ViDu ; uses crt;
var   d , n : byte ;
begin
clrscr;
for  d := 0 to 20  do
 for  n := 0 to 33  do
if 5*d + 3*n + (100-d-n) =100 then
writeln (d,' trau dung ' ,n,' trau nam ' ,100-d-n,' trau gia') ;
 readln
end.
```

5. Kết luận

Lần đầu tiên dạy cho HS về câu lệnh lặp **for**, theo chúng tôi, tiến trình dẫn dắt HS lĩnh hội kiến thức cùng với việc lựa chọn các tình huống đã nêu là phù hợp, bởi vì nó kết hợp được các ưu điểm quan trọng như:

- Trong thời gian ngắn, GV giúp HS hiểu được cú pháp và hoạt động của câu lệnh lặp **for** dạng tiến và dạng lùi, của cấu trúc lặp **for** lồng nhau.

- Với phương châm thực hiện mục đích chức năng thông qua mục đích phương tiện, ta đồng thời đạt được các ý đồ về phương pháp dạy học như: bồi dưỡng quan điểm triết học, đạt được nhiều phương diện như: giao tiếp, thuật toán, dữ liệu, quá trình, sử dụng máy tính, kĩ năng lập trình từ dưới lên và kĩ năng giải quyết vấn đề.

Các bài toán khó hơn và hay hơn sử dụng câu lệnh lặp **for** sẽ có cơ hội được giới thiệu ở những tiết học sau.

TÀI LIỆU THAM KHẢO

[1] Lê Khắc Thành, *Phương pháp dạy học đại cương môn tin học*, NXB Đại học Sư Phạm Hà Nội, 2006.

[2] Lê Khắc Thành, *Phương pháp dạy học chuyên ngành môn tin học*, NXB Đại học Sư Phạm Hà Nội, 2008.

[3] Hồ Cẩm Hà (Chủ biên), Lê Khắc Thành, Nguyễn Chí Trung, *Dạy học theo chuẩn kiến thức kĩ năng Tin học 10*, NXB Đại học Sư Phạm Hà Nội, 2010.

[4] Hồ Sỹ Đàm (Chủ biên), Hồ Cẩm Hà, Trần Đỗ Hùng, Nguyễn Đức Nghĩa, Nguyễn Thanh Tùng, Ngô Ánh Tuyết, *Tin học 11*, NXB Giáo dục, 2007.

SUMMARY

Pedagogical approaches in teaching the repeating FOR syntax clause in school informatics often start from calculating sum of a series. This article proposes some sample with effective teaching. This teaching has achieved many ideas of teaching methods and simple to implement in short teaching period.