§10. CẤU TRÚC LẶP

1. Lặp

Với a là số nguyên và $a > 2$, xét các bài toán sau đây:

Bài toán 1. Tính và đưa kết quả ra màn hình tổng

$$S = \frac{1}{a} + \frac{1}{a+1} + \frac{1}{a+2} + \ldots + \frac{1}{a+100}.$$

Bài toán 2. Tính và đưa kết quả ra màn hình tổng

$$S = \frac{1}{a} + \frac{1}{a+1} + \frac{1}{a+2} + \ldots + \frac{1}{a+N} + \ldots$$

cho đến khi $\frac{1}{a+N} < 0{,}0001$.

Với cả hai bài toán, dễ thấy cách để tính tổng S có nhiều điểm tương tự:

- Xuất phát, S được gán giá trị $\frac{1}{a}$;
- Tiếp theo, cộng vào tổng S một giá trị $\frac{1}{a+N}$ với $N = 1, 2, 3, 4, 5,\ldots$

  Việc cộng này được lặp lại một số lần.

Đối với bài toán 1, số lần lặp là 100 và việc cộng vào tổng S sẽ kết thúc khi đã thực hiện việc cộng 100 lần.

Đối với bài toán 2, số lần lặp chưa biết trước nhưng việc cộng vào tổng S sẽ kết thúc khi điều kiện $\frac{1}{a+N} < 0{,}0001$ được thoả mãn.

Nói chung, trong một số thuật toán có những thao tác phải thực hiện lặp đi lặp lại một số lần. Một trong các đặc trưng của máy tính là có khả năng thực hiện hiệu quả các thao tác lặp. Cấu trúc lặp mô tả thao tác lặp và có hai dạng là *lặp với số lần biết trước* và *lặp với số lần chưa biết trước*.

Các ngôn ngữ lập trình đều có các câu lệnh để mô tả cấu trúc lặp.

2. Lặp với số lần biết trước và câu lệnh for-do

Có hai thuật toán *Tong_1a* và *Tong_1b* để giải bài toán 1 như sau:

Thuật toán Tong_1a

Bước 1. $S \leftarrow 1/a$; $N \leftarrow 0$; {Khởi tạo S và N}

Bước 2. $N \leftarrow N + 1$;

Bước 3. Nếu $N > 100$ thì chuyển đến bước 5;

Bước 4. $S \leftarrow S + 1/(a + N)$ rồi quay lại bước 2;

Bước 5. Đưa S ra màn hình, rồi kết thúc.

Thuật toán Tong_1b

Bước 1. $S \leftarrow 1/a$; $N \leftarrow 101$; {Khởi tạo S và N}

Bước 2. $N \leftarrow N - 1$;

Bước 3. Nếu $N < 1$ thì chuyển đến bước 5;

Bước 4. $S \leftarrow S + 1/(a + N)$ rồi quay lại bước 2;

Bước 5. Đưa S ra màn hình rồi kết thúc.

Lưu ý, số lần lặp của cả hai thuật toán trên là biết trước và như nhau (100 lần).

Trong thuật toán *Tong_1a*, giá trị N khi bắt đầu tham gia vòng lặp là 1 và sau mỗi lần lặp N tăng lên 1 cho đến khi $N > 100$ ($N = 101$) thì kết thúc lặp (thực hiện đủ 100 lần). Trong thuật toán *Tong_1b*, giá trị N bắt đầu tham gia vòng lặp là 100 và sau mỗi lần lặp N giảm đi 1 cho đến khi $N < 1$ ($N = 0$) thì kết thúc lặp (thực hiện đủ 100 lần). Ta nói cách lặp trong thuật toán *Tong_1a* là dạng tiến và trong thuật toán *Tong_1b* là dạng lùi.

Để mô tả cấu trúc lặp với số lần biết trước, Pascal dùng câu lệnh lặp *for-do* với hai dạng tiến và lùi như sau:

- *Dạng lặp tiến:*

  `for` <*biến đếm*>`:=` <*giá trị đầu*> `to` <*giá trị cuối*> `do` <*câu lệnh* >`;`

- *Dạng lặp lùi:*

  `for` <*biến đếm*>`:=` <*giá trị cuối*> `downto` <*giá trị đầu* > `do` <*câu lệnh*>`;`

  trong đó:

- *Biến đếm* là biến đơn, thường có kiểu nguyên.
- *Giá trị đầu, giá trị cuối* là các biểu thức cùng kiểu với biến đếm và *giá trị đầu* phải nhỏ hơn hoặc bằng *giá trị cuối*. Nếu *giá trị đầu* lớn hơn *giá trị cuối* thì vòng lặp không được thực hiện.

Hoạt động của lệnh *for-do*:

- Ở dạng *lặp tiến*, câu lệnh viết sau từ khoá *do* được thực hiện tuần tự, với *biến đếm* lần lượt nhận các giá trị liên tiếp tăng từ *giá trị đầu* đến *giá trị cuối*.
- Ở dạng *lặp lùi*, câu lệnh viết sau từ khoá *do* được thực hiện tuần tự, với *biến đếm* lần lượt nhận các giá trị liên tiếp giảm từ *giá trị cuối* đến *giá trị đầu*.

Chú ý: Giá trị của *biến đếm* được điều chỉnh tự động, vì vậy câu lệnh viết sau ***do*** không được thay đổi giá trị *biến đếm*.

Ví dụ 1. Sau đây là hai chương trình cài đặt các thuật toán *Tong_1a* và *Tong_1b*.

```
program Tong_1a;
uses crt;
var S: real;
   a, N: integer;
begin
   clrscr;
   write('Hay nhap gia tri a vao!');
   readln(a);
   S:=1.0/a;                          {Buoc 1}
   for N:= 1 to 100 do                {Buoc 2, Buoc 3}
       S:= S+1.0/(a+N);               {Buoc 4}
   writeln('Tong S la: ', S:8:4);     {Buoc 5}
   readln
end.
program Tong_1b;
uses crt;
var S: real;
   a, N: integer;
begin
   clrscr;
   write ('Hay nhap gia tri a vao!');
   readln(a);
```

```
    S:=1.0/a;                          {Buoc 1}
    for N:= 100 downto 1 do            {Buoc 2 va Buoc 3}
        S:= S+1.0/(a+N);               {Buoc 4}
    writeln('Tong S la: ', S:8:4);     {Buoc 5}
    readln
end.
```

Ví dụ 2. Chương trình sau thực hiện việc nhập từ bàn phím hai số nguyên dương M và N ($M < N$), tính và đưa ra màn hình tổng các số chia hết cho 3 hoặc 5 trong phạm vi từ M đến N.

```
program Vi_du_2;
uses crt;
var M, N, I: integer;
    T: longint;
begin
   clrscr;
   writeln('Nhap so M nho hon N');
   write('M = ');readln(M);
   write('N = ');readln(N);
   T:= 0;
   for I:= M to N do
      if (I mod 3 = 0) or (I mod 5 = 0) then
         T:=T+I;
   writeln('KET QUA: ', T);
   readln
end.
```

3. Lặp với số lần chưa biết trước và câu lệnh while-do

Có thể xây dựng thuật toán *Tong_2* như sau để giải bài toán 2.

Thuật toán Tong_2

Bước 1. $S \leftarrow 1/a$; $N \leftarrow 0$; {Khởi tạo S và N}

Bước 2. Nếu $1/(a + N) < 0{,}0001$ thì chuyển đến bước 5;

Bước 3. $N \leftarrow N + 1$;

Bước 4. $S \leftarrow S + 1/(a + N)$ rồi quay lại bước 2;

Bước 5. Đưa S ra màn hình, rồi kết thúc.

Như vậy, việc lặp với số lần chưa biết trước sẽ chỉ kết thúc khi một điều kiện cho trước được thoả mãn.

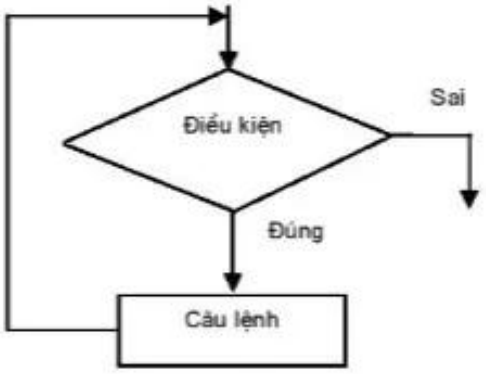
Để mô tả cấu trúc lặp như vậy, Pascal dùng câu lệnh *while-do* có dạng:

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

trong đó:

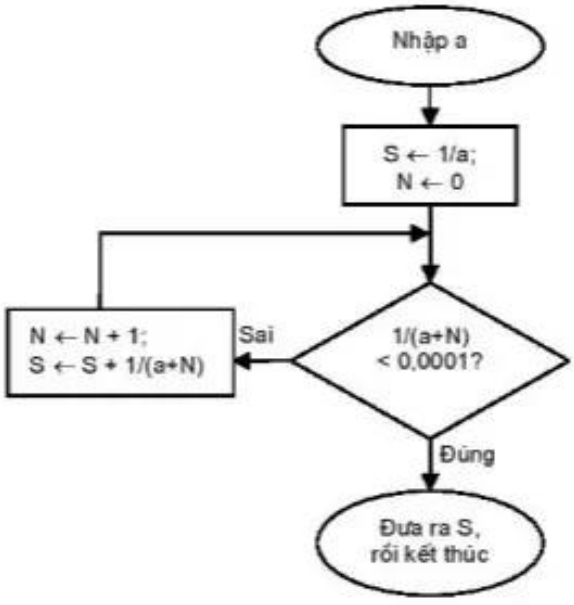
- *Điều kiện* là biểu thức lôgic;
- *Câu lệnh* là một câu lệnh đơn hoặc ghép.

Việc thực hiện lệnh *while-do* được thể hiện trên sơ đồ ở hình 7.



Hình 7. *Sơ đồ lặp với số lần lặp chưa biết trước*

Ví dụ 1. Sau đây là sơ đồ khối và chương trình cài đặt thuật toán *Tong_2*.



Hình 8. *Sơ đồ khối của thuật toán Tong_2*

```
program Tong_2;
uses crt;
var    S: real;
       a, N: integer;
begin
   write ('Hay nhap gia tri a vao!');
   readln(a);
   S:= 1.0/a; N:= 0;                          {Buoc 1}
    while not (1/(a+N)<0.0001) do             {Buoc 2}
       begin
           N:= N+1;                           {Buoc 3}
           S:= S+1.0/(a+N);                   {Buoc 4}
       end;
   writeln('Tong S la: ', S:8:4);             {Buoc 5}
   readln
end.
```

Ví dụ 2. Tìm ước chung lớn nhất (ƯCLN) của hai số nguyên dương M và N.

Có nhiều thuật toán khác nhau tìm ƯCLN của M và N. Sau đây là một thuật toán tìm ƯCLN.

Thuật toán

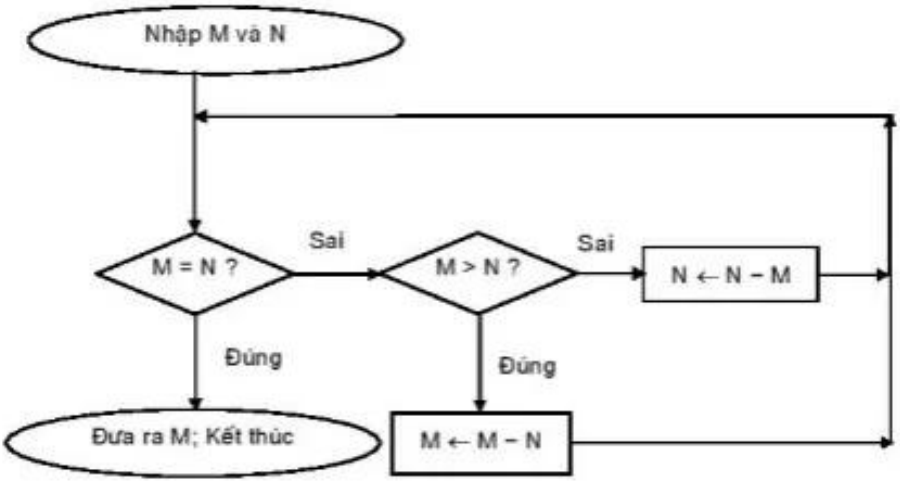
Bước 1. Nhập M, N;

Bước 2. Nếu $M = N$ thì lấy giá trị chung này làm ƯCLN rồi chuyển đến bước 5;

Bước 3. Nếu $M > N$ thì $M \leftarrow M - N$ ngược lại $N \leftarrow N - M$;

Bước 4. Quay lại bước 2;

Bước 5. Đưa ra kết quả ƯCLN rồi kết thúc.



Hình 9. *Sơ đồ khối của thuật toán tìm ước chung lớn nhất*

Chương trình sau thể hiện thuật toán tìm ước chung lớn nhất.

```
program UCLN;
uses crt;
var M,N:integer;
begin
   clrscr;
   write('M, N = ');
   readln(M,N);
   while M <> N do
      if M > N then M:= M-N else N:= N-M;
   writeln('UCLN = ', M);
   readln
end.
```

Chú ý: Các câu lệnh trong vòng lặp thường được lặp lại nhiều lần, vì vậy để tăng hiệu quả của chương trình thì những thao tác không cần lặp lại nên đưa ra ngoài vòng lặp.

TÓM TẮT

- Các ngôn ngữ lập trình đều có câu lệnh thể hiện cấu trúc rẽ nhánh và cấu trúc lặp.
- Câu lệnh rẽ nhánh có hai dạng:
  - Dạng thiếu;
  - Dạng đủ.
- Có thể gộp dãy câu lệnh thành câu lệnh ghép.
- Các câu lệnh mô tả cấu trúc lặp:
  - Lặp với số lần biết trước;
  - Lặp với số lần chưa biết trước.

Định lí Bohn Jacopini (Bon Ja-co-pi-ni): Mọi quá trình tính toán đều có thể mô tả và thực hiện dựa trên ba cấu trúc cơ bản là cấu trúc tuần tự, cấu trúc rẽ nhánh và cấu trúc lặp.