Nguyên lý và phương pháp lập trình

Kiểm chứng tính đúng đắn của chương trình

Nội dung

- Lập trình có cấu trúc
- Các phương pháp hình thức
- Lập trình có cấu trúc theo tiếp cận top-down
- Cách tiếp cận kết hợp
- Các cấu trúc trình tự (sequences)
  - Các cấu trúc trình tự – hình thức hóa
  - Các cấu trúc trình tự – kiểm chứng
  - Các cấu trúc trình tự – sơ đồ kiểm chứng
- Các cấu trúc điều kiện
  - Các cấu trúc điều kiện – hình thức hóa
  - Các cấu trúc điều kiện – kiểm chứng
  - Các cấu trúc điều kiện – sơ đồ kiểm chứng

Nội dung (tt)

- Các cấu trúc vòng lặp
  - Các cấu trúc vòng lặp – ví dụ
  - Các cấu trúc vòng lặp – chương trình dẫn xuất
  - Các cấu trúc vòng lặp – kết quả
  - Các cấu trúc vòng lặp – sơ đồ kiểm chứng
  - Các cấu trúc vòng lặp – các lỗi trong kiểm chứng
- Tóm lược

Lập trình có cấu trúc

- Chương trình sử dụng các cấu trúc điều khiển căn bản theo nguyên tắc One-in, One-out :
  - Cấu trúc trình tự : **begin** S1 S2 **end**
  - Cấu trúc rẽ nhánh : **if** E **then** S1 **else** S2 **end**
  - Cấu trúc vòng lặp : **while** E **loop** S1 **end**

- Boehm và Jacopini, 1966
  - Chứng minh rằng cấu trúc điều khiển của bất kỳ một lược đồ chương trình nào cũng có thể được biểu đạt mà không cần dùng các phát biểu goto, chỉ cần dùng các cấu trúc: trình tự, chọn lựa và vòng lặp.

Lập trình có cấu trúc

- Edsger Dijkstra, 1970
  - Lý luận rằng các phát biểu goto là có hại trong chương trình, đồng thời đưa ra ý tưởng về việc kiểm chứng tính đúng đắn của chương trình bằng các phương pháp hình thức.

Các phương pháp hình thức

- Dựa trên ý tưởng rằng các chương trình là những đối tượng toán học
  - Một chương trình "hữu hạn" (có tính dừng) là biểu diễn của một hàm, hàm này ánh xạ các trạng thái nào đó của bộ nhớ (các trạng thái bắt đầu) đến các trạng thái khác của bộ nhớ (các trạng thái kết thúc).
  - Một chương trình "vô hạn" (không dừng) có thể được diễn đạt như một module "hữu hạn" (có tính dừng) trong một vòng lặp vô hạn.

Các phương pháp hình thức (tt)

- Tony Hoare, 1969
  - Viết một bài báo tựa đề: “An Axiomatic Basis for Computer Programming”. Bài báo này mô tả sự dẫn xuất của mã chương trình từ các tiên đề, theo cách đó thì tính đúng đắn của chương trình có thể chứng minh một cách hình thức.
- Harlan Mills, 1987
  - Phát triển kỹ thuật “Cleanroom Software Engineering” dựa trên các phương pháp hình thức để kiểm tra tính đúng đắn của chương trình.

Các phương pháp hình thức (tt)

- Ưu điểm:
  - Cung cấp một phương pháp có thể kiểm chứng được để khai triển các đặc tả thành chương trình
  - Dựa trên các kỹ thuật toán học hình thức
  - Cách hiệu quả để kiểm chứng tính đúng đắn của chương trình
  - Được hỗ trợ bằng một số ngôn ngữ hình thức (Z, VDM-SL, REFINE), các hệ thống kiểm tra-kiểm chứng.
- Nhược điểm:
  - Một số các khái niệm của phương pháp này đôi khi còn trừu tượng

Lập trình có cấu trúc theo tiếp cận top-down

- Phương pháp
  - Mịn hóa từng bước
  - Dùng các cấu trúc điều khiển dạng One-in / One-out
  - Các kỹ thuật kiểm chứng
- Bắt đầu với
  - Một phát biểu chưa được mịn hóa
  - Một đặc tả hình thức của phát biểu

Lập trình có cấu trúc theo tiếp cận top-down (tt)

- Mịn hóa từng bước các phát biểu để đạt được kết quả là thay thế chúng (nếu có) bằng:
  - Các cấu trúc trình tự
  - Các cấu trúc điều kiện
  - Các cấu trúc lặp
  - Các phát biểu thực thi (gán, gọi hàm)
- Kiểm tra tính đúng đắn ở mỗi bước

Các cấu trúc trình tự

1. Vấn đề: làm mịn phát biểu sau

   [F: Đặt x = y * | z |]

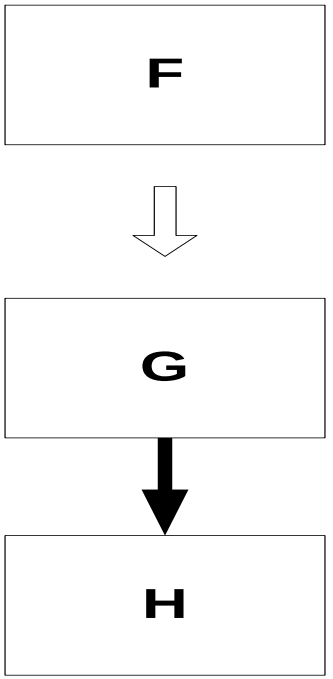
2. Bước 1: Thay F bằng các phát biểu sau

   [G: Đặt k = | z |]

   [H: Đặt x = y * k]

3. Phương pháp thay F bằng cấu trúc trình tự:

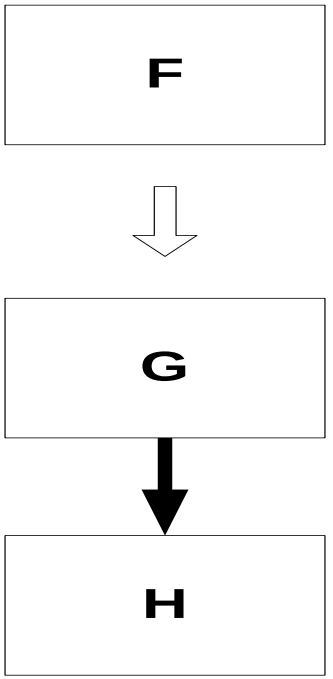
   – Chọn G



Các cấu trúc trình tự (tt)

- Chọn H sao cho

  (a) H được xây dựng trên cơ sở G và H phải tương đương với F
- Kiểm tra (a)



Các cấu trúc trình tự – Hình thức hóa

4. Hình thức hóa: thay thế mỗi phát biểu bằng một *đặc tả hình thức* với các tiên đề P, Q, và R:

formal spec	PDL
{P} F {R} {P} G {Q} {Q} H {R}	[F: let x = y * | z |] [G: let k = | z |] [H: let x = y * k]

where P = (x, y, z are integers)
Q = (P and K is integer and k = |z|)
R = (P and x = y * |z|)

Các cấu trúc trình tự – Hình thức hóa (tt)

- {P}F{R} có nghĩa:
  "Nếu P đúng *trước khi* F được thực hiện, thì R sẽ đúng *sau khi* F được thực hiện."
- P được gọi là *tiền điều kiện* (precondition) của F
- R được gọi là *hậu điều kiện* (postcondition) của F

Các cấu trúc trình tự – Kiểm chứng

5. Giả sử G và H đã được kiểm chứng đúng, nghĩa là:

$$\{P\}\ G\ \{Q\} \text{ and } \{Q\}\ H\ \{R\}$$

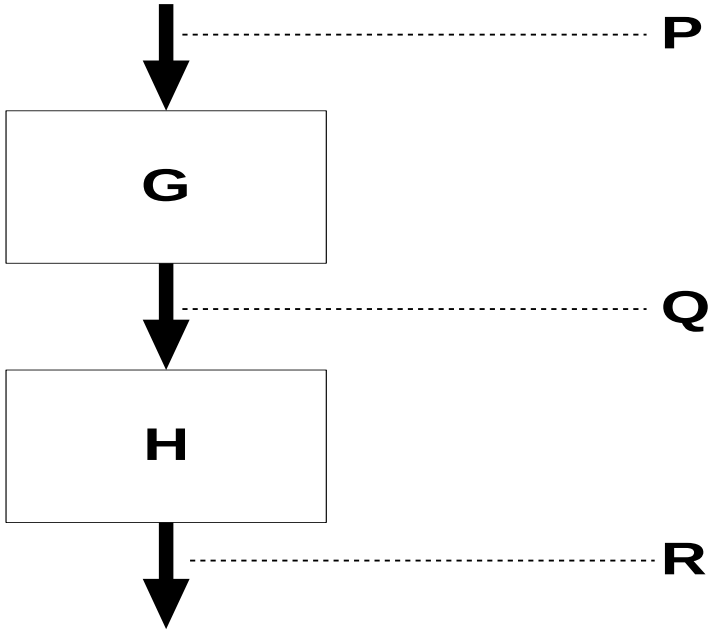
6. Thì F đúng, vì

$$\{P\}\ F\ \{R\}$$

7. Bước (6) được suy ra từ bước (5) bằng qui tắc kiểm chứng:

$$\frac{\{P\}\ G\ \{Q\},\ \{Q\}\ H\ \{R\}}{\{P\}\ G\ H\ \{R\}}$$

Các cấu trúc trình tự – Sơ đồ kiểm chứng



Proof rule for sequences

$$\frac{\{P\}\ G\ \{Q\},\ \{Q\}\ H\ \{R\}}{\{P\}\ G\ H\ \{R\}}$$

Các cấu trúc điều kiện

1. Vấn đề: mịn hóa phát biểu sau

   [G: Đặt k = | z |]

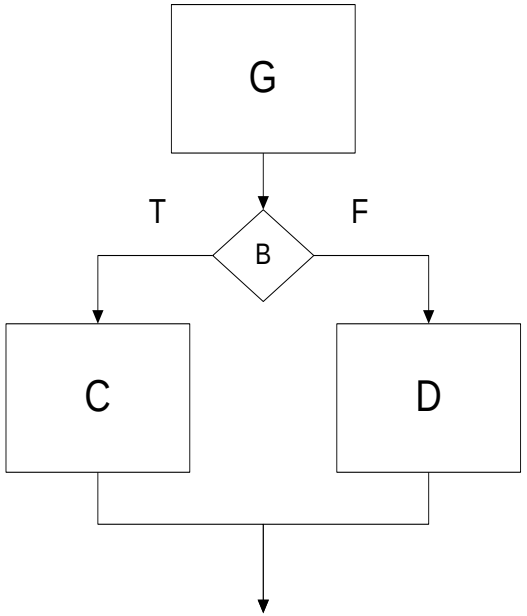
2. Bước 2: Thay thế G với cấu trúc điều kiện

   G: **if** z < 0 [test B]

   then [C: Đặt k = -z]

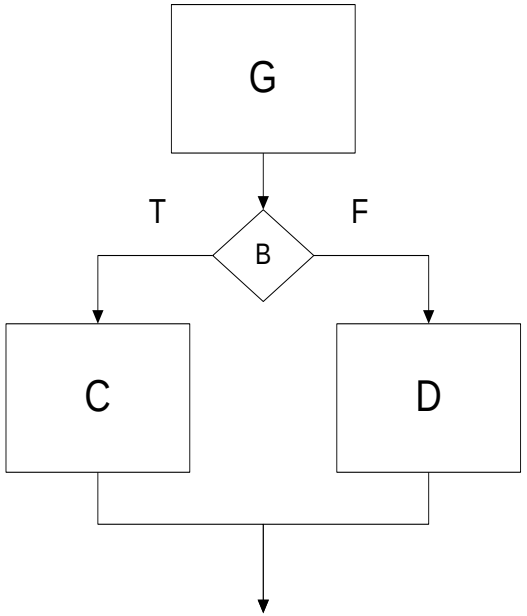
   else [D: Đặt k = z]

   end



Các cấu trúc điều kiện

3. Phương pháp thay thế G bằng một cấu trúc điều kiện:
   - Chọn điều kiện B
   - Chọn C và D sao cho
     (a) C thực hiện G khi B đúng
     (b) D thực hiện G khi B sai
   - Kiểm tra (a) và (b)



Các cấu trúc điều kiện – Hình thức hóa

4. Các đặc tả hình thức cho G, C, D:

formal spec	PDL
{P} G {Q} {P and B} C {Q} {P and not B} D {Q}	[G: let k = | z |] G: **if** z < 0 [test b] **then** [C: let k = -z] **else** [D: let k = z] **end**

where P = (x, y, z are integers)
B = (z < 0)
Q = (P and k is integer and k = |z|)

Các cấu trúc điều kiện – Hình thức hóa (tt)

5. Nếu C và D đúng, G được kiểm chứng bằng qui tắc:

$$\frac{\{P \text{ and } B\}\ C\ \{Q\},\ \{P \text{ and not } B\}\ D\ \{Q\}}{\{P\} \text{ if } B \text{ then } C \text{ else } D \text{ end } \{Q\}}$$

Các cấu trúc điều kiện – Kiểm chứng

6. Bước mịn hóa cuối cùng: thay C, D, và H bằng các phát biểu (của ngôn ngữ lập trình) thỏa mãn đặc tả:

```
G: if      z < 0                    [kiểm tra B]
   then    C: k := -z
   else    D: k := z
   end
H: x := y * k
```

Các cấu trúc điều kiện – Kiểm chứng (tt)

7. Kiểm chứng tính đúng đắn:

   {P and B} C {Q} [định nghĩa phát biểu gán]

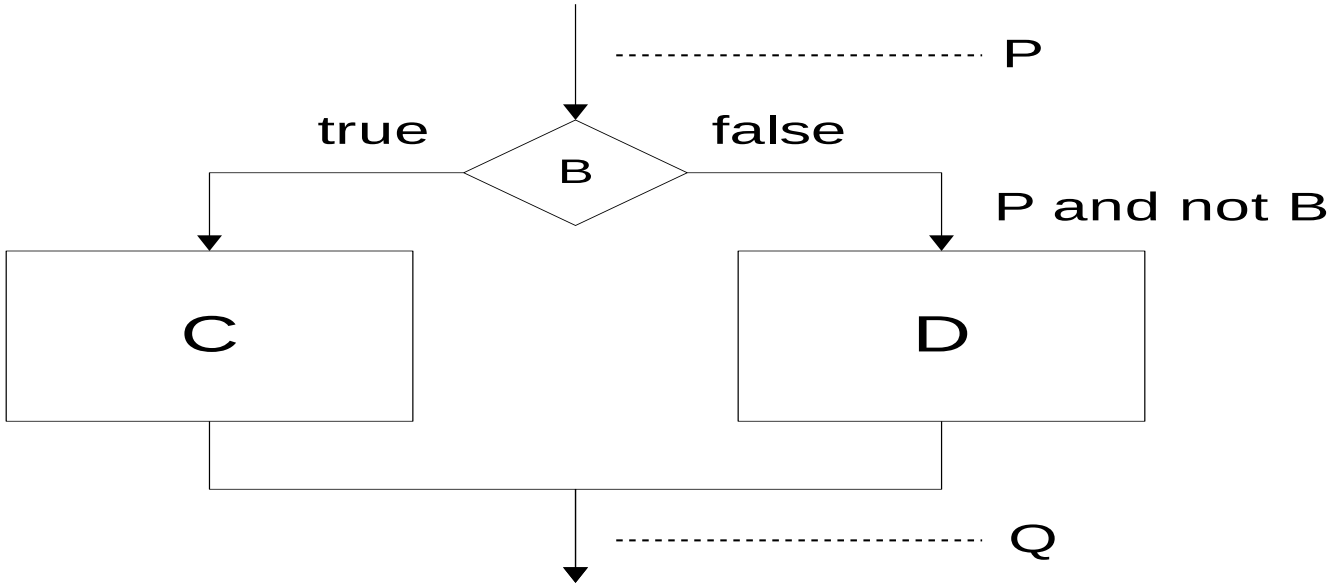
   {P and not B}D{Q} [định nghĩa phát biểu gán]

   {P} G {Q} [Qui tắc kiểm chứng cho các cấu trúc điều kiện]

   {Q} H {R} [định nghĩa phát biểu gán]

   {P} F {R} [Qui tắc kiểm chứng cho các cấu trúc trình tự]

Các cấu trúc điều kiện – Sơ đồ kiểm chứng



Qui tắc kiểm chứng cho các điều kiện:

$$\frac{\{P \text{ and } B\}\ C\ \{Q\},\ \{P \text{ and not } B\}\ D\ \{Q\}}{\{P\}\ \text{if B then C else D end}\ \{Q\}}$$

Các cấu trúc vòng lặp

1. Dạng tổng quát cho vòng lặp while :

formal spec	PDL
{P} F {R} {P} G {Q} {Q and B} H {Q}	[F: ... function] [G: ... initialization] **while** B **loop** [H: ... body] **end**

Các cấu trúc vòng lặp (tiếp theo)

2. Qui tắc kiểm chứng cho vòng lặp while:

$$\frac{\{P\}\ G\ \{Q\},\ \{Q \text{ and } B\}\ H\ \{Q\}}{\{P\}\ G;\ \text{while } B \text{ loop } H \text{ end } \{Q \text{ and not } B\}}$$

3. Phương pháp thay thế F bằng cấu trúc vòng lặp while:
   - Chọn Q (bất biến vòng lặp) và B (điều kiện vòng lặp) sao cho (Q and not B) bao hàm R
   - Chọn G và H sao cho

     (a) G làm cho Q đúng

     (b) B có thể sai (để vòng lặp kết thúc)

     (c) thực hiện H khi B và Q đúng
   - Kiểm tra (a), (b), và (c)

Các cấu trúc vòng lặp – Ví dụ

4. Vấn đề: mịn hóa phát biểu

   [F: Tìm phần tử nhỏ nhất a(j) trong dãy các phần tử a(i..n) của mảng a]

5. Đặc tả hình thức: {P} F {R} trong đó

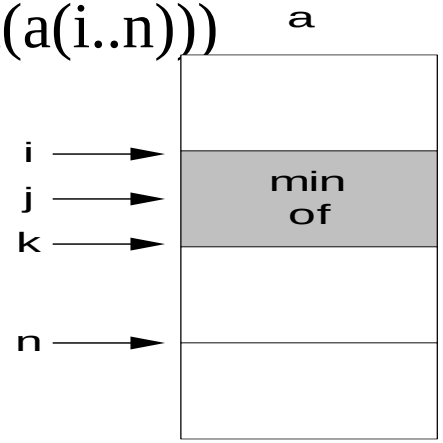
   P=(a ∈ integer array and i, n ∈ integer and $1 \le i \le n \le$ size(a))

   R=(P and j ∈ integer and $i \le j \le n$ and a(j)=min(a(i..n)))

6. Giải pháp:

   Q=(P and j, k ∈ integer and $i \le j \le k \le n$ and ↓ a(j)=min(a(i..k)))

   B=(k<n)

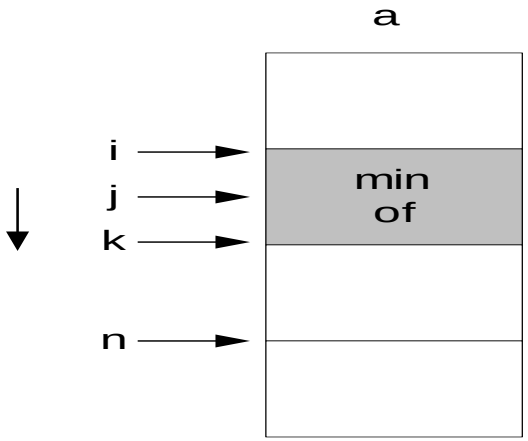


Các cấu trúc vòng lặp – Ví dụ

7. Kiểm tra:

(Q and not B) bao hàm k=n

∴ (Q and not B) bao hàm R



Các cấu trúc vòng lặp

```
proc   f (a [array], i, n [integers] )
descr  [f(a, i, n) = j where a(j) = min(a(i..n))]
local  j, k [integers]
pre                [1 ≤ i ≤ n ≤ size(a) ]
post   [1 ≤ i ≤ j ≤ n ≤ size(a) and
       a(j)=min(a(i..n)) ]
begin  if not (1<=i and i<=n and n<= size(a))
       then write("input error") return nil
       end
       j := i
       k := i
       while k < n
```

Các cấu trúc vòng lặp

```
        loop    [loop invariant: a(j)=min(a(i..k))
                  and 1 ≤ i ≤ j ≤ k ≤ n ≤ size(a) ]
                k := k + 1
                if a(j) > a(k)
                then j := k
                end
        end
        return j
end
```

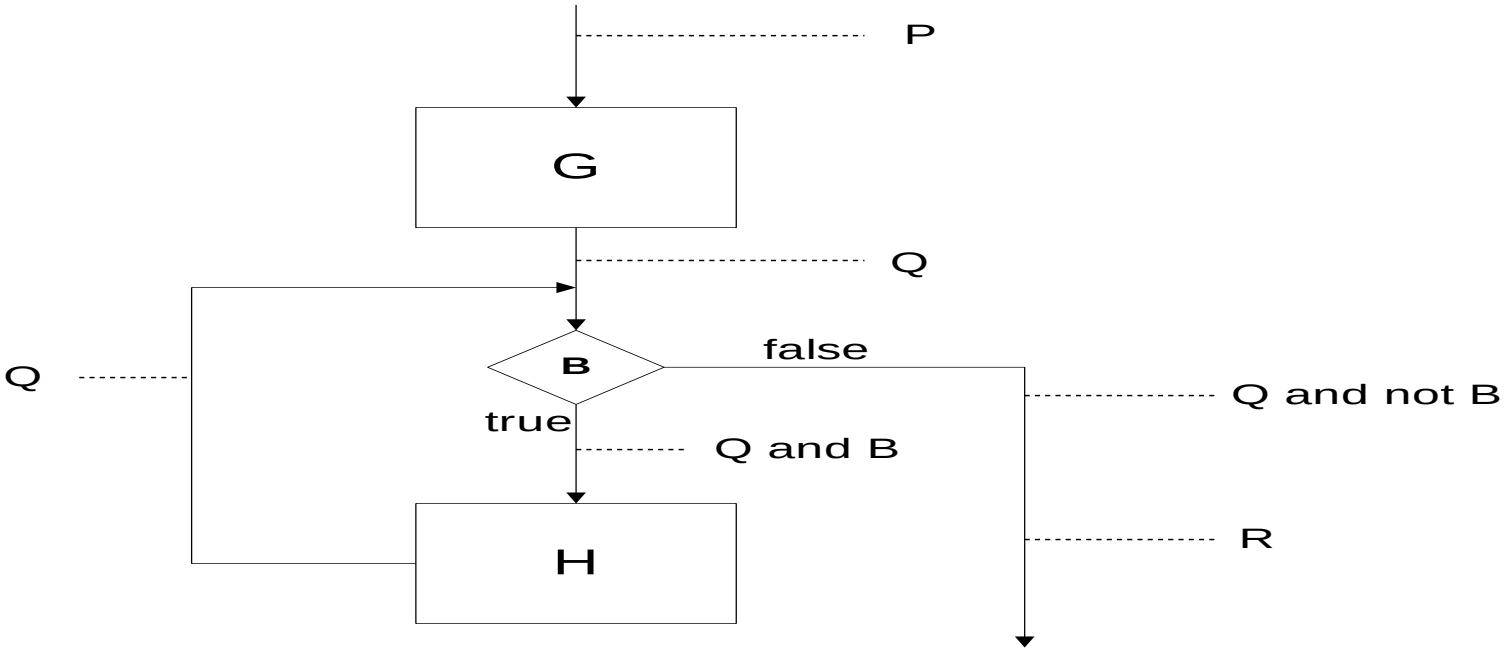
Các cấu trúc vòng lặp – Chương trình dẫn xuất

```
    [ P: … 1 ≤ i ≤ n ≤ size(a) ]
G:  j := i
    k := i
    [ Q: P and … i ≤ j ≤ k ≤ n
         and a(j)=min(a(i..k)) ]
    while k < n [test B]
    loop [ L1: Q and B ]
    H:   k := k + 1                                  [để B có thể sai]
         [ L2: P and … a(j)=min(a(i..k-1)) ]         [để Q đúng]
         if a(j) > a(k)
         then    [ T1: L2 and a(k) < a(j) ]
```

Các cấu trúc vòng lặp – Chương trình dẫn xuất (tt)

```
                j := k
                [ T2: … a(j)=min(a(i..k)) ]
        else    [ F1: … a(j) ≤ a(k) ]
        end
end
[ R: Q and not B ]
```

Các cấu trúc vòng lặp – Sơ đồ kiểm chứng



Qui tắc kiểm tra cho vòng lặp while là :

{P} G {Q}, {Q and B} H {Q}

{P} G; while B loop H end {Q and not B}

Các cấu trúc vòng lặp – Các lỗi trong kiểm chứng

- Các lỗi phổ biến trong việc kiểm chứng tính đúng đắn của chương trình:
  - Không xác định đúng bất biến vòng lặp Q. Một bất biến vòng lặp là một khẳng định không thay đổi, chẳng hạn như Q đúng:
    - Trước khi thực hiện vòng lặp
    - Sau mỗi bước của vòng lặp
    - Sau khi kết thúc vòng lặp
  - Không kiểm tra xem vòng lặp có kết thúc hay không – trường hợp các vòng lặp vô hạn

Tóm lược

- Có thể áp dụng kiểm chứng hình thức đối với:
  - Các cấu trúc trình tự
  - Các cấu trúc điều kiện (rẽ nhánh)
  - Các cấu trúc vòng lặp
- Mỗi phát biểu có:
  - Một tiền điều kiện
  - Một hậu điều kiện
- Mỗi thủ tục có:
  - Một tiền điều kiện
  - Một hậu điều kiện

Tóm lược (tt)

- Mỗi cấu trúc vòng lặp có:
  - Một tiền điều kiện
  - Một hậu điều kiện
  - Một bất biến vòng lặp