[Bài đọc] Pseudo-code và flowchart

Chúng ta tuân theo những cấu trúc tuần tự, mà trong đó luồng thực thi chương trình đi qua tất cả các chỉ thị bắt đầu từ chỉ thị đầu tiên. Chúng ta có thể bắt gặp các điều kiện trong chương trình, dựa trên các điều kiện này hướng thực thi của chương trình có thể rẽ nhánh. Những cấu trúc cho việc rẽ nhánh như là cấu trúc chọn lựa, cấu trúc điều kiện hay rẽ nhánh. Những cấu trúc này được đề cập chi tiết sau đây:

**Cấu trúc IF (Nếu)**

Cấu trúc chọn lựa cơ bản là cấu trúc ‘IF`. Ðể hiểu cấu trúc này chúng ta hãy xem xét ví dụ trong đó khách hàng được giảm giá nếu mua trên 100 đồng. Mỗi lần khách hàng trả tiền, một đoạn mã chương trình sẽ kiểm tra xem lượng tiền trả có quá 100 đồng không?. Nếu đúng thế thì sẽ giảm giá 10% của tổng số tiền trả, ngược lại thì không giảm giá.

Ðiều này được minh họa sơ lược qua mã giả như sau:

IF khách hàng mua trên 100 thì giảm giá 10%

Cấu trúc dùng ở đây là câu lệnh IF.

Hình thức chung cho câu lệnh IF (cấu trúc IF) như sau:

IF Điều kiện  
Các câu lệnh *//Phần thân của cấu trúc IF*  
*END IF*

Một cấu trúc "IF" bắt đầu là IF theo sau là điều kiện. Nếu điều kiện là đúng (thỏa điều kiện) thì quyền điều khiển sẽ được chuyển đến các câu lệnh trong phần thân để thực thi. Nếu điều kiện sai (không thỏa điều kiện), những câu lệnh ở phần thân không được thực thi và chương trình nhảy đến câu lệnh sau END IF (chấm dứt cấu trúc IF). Cấu trúc IF phải được kết thúc bằng END IF.

Chúng ta xem ví dụ 4 cho cấu trúc IF.

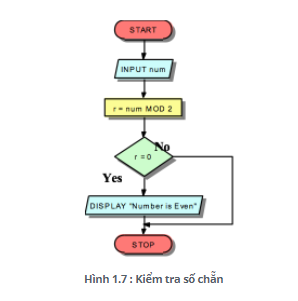
Ví dụ 4:

Yêu cầu: Kiểm xem một số là chẵn hay không và hiển thị thông điệp báo nếu đúng là số chẵn,  ta xử lý như sau :

BEGIN  
   INPUT num  
   r = num MOD 2  
   IF r=0  
     Display "Number is even"  
   END IF  
END

Ðoạn mã trên nhập một số từ người dùng, thực hiện toán tử MOD (lấy phần dư) và kiểm tra xem phần dư có bằng 0 hay không. Nếu bằng 0 hiển thị thông điệp, ngược lại thoát ra.

Lưu đồ cho đoạn mã giả trên thể hiện qua hình 1.7.



## ấu trúc IF…ELSE

Trong ví dụ 4, sẽ hay hơn nếu ta cho ra thông điệp báo rằng số đó không là số chẵn tức là số lẻ thay vì chỉ thoát ra. Ðể làm điều này ta có thể thêm câu lệnh IF khác để kiểm tra xem trường hợp số đó không chia hết cho 2. Ta xem ví dụ 5.

### Ví dụ 5:

BEGIN  
   INPUT num  
   r = num MOD 2  
   IF r=0  
      DISPLAY "Even number"  
   END IF  
   IF r<>0  
      DISPLAY "Odd number"  
   END IF  
END

Ngôn ngữ lập trình cung cấp cho chúng ta cấu trúc **IF…ELSE**. Dùng cấu trúc này sẽ hiệu quả và tốt hơn để giải quyết vấn đề. Cấu trúc **IF …ELSE** giúp lập trình viên chỉ làm một phép so sánh và sau đó thực thi các bước tùy theo kết quả của phép so sánh là True (đúng) hay False (sai).

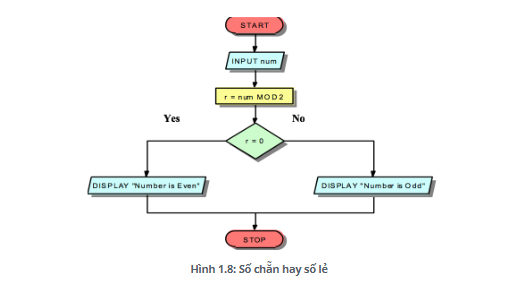
Cấu trúc chung của câu lệnh IF…ELSE như sau:

IF Điều kiện  
   Câu lệnh 1  
ELSE  
   Câu lệnh 2  
END IF

### Ví dụ 6:

BEGIN  
  INPUT num  
  r = num MOD 2  
  IF r = 0  
    DISPLAY "Even Number"  
  ELSE  
    DISPLAY "Odd Number"  
  END IF  
END

Lưu đồ cho đoạn mã giả trên thể hiện qua Hình 1.8.



**Hình 1.8: Số chẵn hay số lẻ**

## Ða điều kiện sử dụng AND/OR

Cấu trúc IF…ELSE làm giảm độ phức tạp, gia tăng tính hữu hiệu. Ở một mức độ nào đó, nó cũng nâng cao tính dễ đọc của mã. Các thí dụ IF chúng ta đã đề cập đến thời điểm này thì khá đơn giản. Chúng chỉ có một điều kiện trong IF để đánh giá. Thỉnh thoảng chúng ta phải kiểm tra cho hơn một điều kiện, thí dụ: Ðể xem xét nhà cung cấp có đạt MVS (nhà cung cấp quan trọng nhất) không?, một công ty sẽ kiểm tra xem nhà cung cấp đó đã có làm việc với công ty ít nhất 10 năm không? và đã có tổng doanh thu ít nhất 5,000,000 không?. Hai điều kiện thỏa mãn thì nhà cung cấp được xem như là một MVS. Do đó toán tử AND có thể được dùng trong câu lệnh ‘IF’ như trong ví dụ 7 sau:

### Ví dụ 7:

BEGIN  
  INPUT yearsWithUs  
  INPUT bizDone  
  IF yearsWithUs >= 10 AND bizDone >=5000000  
       DISPLAY "Classified as an MVS"  
  ELSE  
       DISPLAY "A little more effort required!"  
  END IF  
END

Ví dụ 7 cũng khá đơn giản, vì nó chỉ có 2 điều kiện. Ở tình huống thực tế, chúng ta có thể có nhiều điều kiện cần được kiểm tra. Nhưng chúng ta có thể dễ dàng dùng toán tử AND để nối những điều kiện lại giống như ta đã làm ở trên.

Bây giờ, giả sử công ty trong ví dụ trên đổi quy định, họ quyết định đưa ra điều kiện dễ dàng hơn. Như là : Hoặc làm việc với công ty  trên 10 năm hoặc có doanh số (giá trị thương mại,giao dịch) từ 5,000,000 trở lên. Vì vâỵ, ta thay thế toán tử AND bằng toán tử OR. Nhớ rằng toán tử OR cho ra giá trị True (đúng) nếu chỉ cần một điều kiện là True.

## Cấu trúc IF lồng nhau

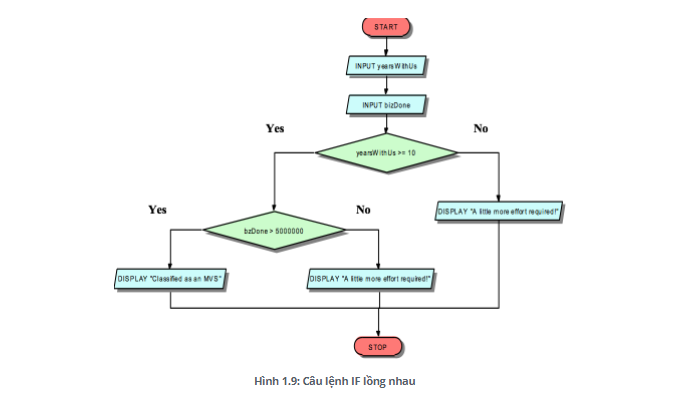
Một cách khác để thực hiện ví dụ 7 là sử dụng cấu trúc IF lồng nhau. Cấu trúc IF lồng nhau là câu lệnh IF này nằm trong trong câu lệnh IF khác. Chúng ta viết lại ví dụ 7 sử dụng cấu trúc IF lồng nhau ở ví dụ 8 như sau:

### Ví dụ 8:

BEGIN  
  INPUT yearsWithUs  
  INPUT bizDone  
  IF yearsWithUs >= 10  
     IF bizDone >=5000000  
        DISPLAY "Classified as an MVS"  
     ELSE  
        DISPLAY "A little more effort required!"  
     END IF  
  ELSE  
     DISPLAY "A little more effort required!"  
  END IF  
END

Ðoạn mã trên thực hiện cùng nhiệm vụ nhưng không có ‘AND`. Tuy nhiên, chúng ta có một  lệnh IF (kiểm tra xem bizDone lớn hơn hoặc bằng 5,000,000 hay không?) bên trong lệnh IF khác (kiểm tra xem yearsWithUs lớn hơn hoặc bằng 10 hay không?). Câu lệnh IF đầu tiên kiểm tra điều kiện thời gian nhà cung cấp làm việc với công ty có lớn hơn 10 năm hay không. Nếu dưới 10 năm (kết quả trả về là False), nó sẽ không công nhận nhà cung cấp là một MVS; Nếu thỏa điều kiện nó xét câu lệnh IF thứ hai, nó sẽ kiểm tra tới điều kiện bizDone lớn hơn hoặc bằng 5,000,000 hay không. Nếu thỏa điều kiện (kết quả trả về là True) lúc đó nhà cung cấp được xem là một MVS, nếu không thì một thông điệp báo rằng đó không là một MVS.

Lưu đồ cho mã giả của ví dụ 8 được trình bày qua hình 1.9.



Mã giả trong trường hợp này của cấu trúc IF lồng nhau tại ví dụ 8 chưa  hiệu quả. Câu lệnh thông báo không thỏa điều kiện MVS phải viết hai lần. Hơn nữa lập trình viên phải viết thêm mã nên trình biên dịch phải xét hai điều kiện của lệnh IF, do đó lãng phí thời gian. Ngược lại, nếu dùng toán tử AND chỉ xét tới điều kiện của câu lệnh IF một lần. Ðiều này không có nghĩa là cấu trúc IF lồng nhau nói chung là không hiệu quả. Nó tùy theo tình huống cụ thể mà ta dùng nó. Có khi dùng toán tử AND hiệu quả hơn, có khi dùng cấu trúc IF lồng nhau hiệu quả hơn. Chúng ta sẽ xét một ví dụ mà dùng cấu trúc IF lồng nhau hiệu quả hơn dùng toán tử AND.

Một công ty định phần lương cơ bản cho công nhân dựa trên tiêu chuẩn như trong bảng 1.1.

|  |  |  |
| --- | --- | --- |
| **Grade** | **Experience** | **Salary** |
| E | 2 | 2000 |
| E | 3 | 3000 |
| M | 2 | 3000 |
| M | 3 | 4000 |

**Bảng 1.1: Lương cơ bản**

Vì vậy, nếu một công nhân được xếp loại là E và có hai năm kinh nghiệm thì lương là 2000, nếu ba năm kinh nghiệm thì lương là 3000.

Mã giả dùng toán tử AND cho vấn đề trên như ví dụ 9:

**Ví dụ 9:**

BEGIN  
  INPUT grade  
  INPUT exp  
  IF grade ="E" AND exp=2  
    salary=2000  
  ELSE  
    IF grade = "E" AND exp=3  
       salary=3000  
    END IF  
  END IF  
  
  IF grade ="M" AND exp=2  
    salary=3000  
  ELSE  
    IF grade = "M" AND exp=3  
      salary=4000  
    END IF  
  END IF  
END

Câu lệnh IF đầu tiên kiểm tra xếp loại và kinh nghiệm của công nhân. Nếu xếp loại là E và kinh nghiệm là 2 năm thì lương là 2000, ngoài ra nếu xếp loại E, nhưng có 3 năm kinh nghiệm thì lương là 3000.

Nếu cả 2 điều kiện không thỏa thì câu lệnh IF thứ hai cũng tương tự sẽ kiểm điều kiện xếp loại và kinh nghiệm cho công nhân để phân định lương.

Giả sử xếp loại của một công nhân là E và có hai năm kinh nghiệm. Lương người đó sẽ được tính theo mệnh đề IF đầu tiên. Phần còn lại của câu lệnh IF thứ nhất được bỏ qua. Tuy nhiên, điều kiện tại mệnh đề IF thứ hai sẽ được xét và tất nhiên là không thỏa, do đó nó kiểm tra mệnh đề ELSE của câu lệnh IF thứ 2 và kết quả cũng là False. Ðây quả là những bước thừa mà chương trình đã xét qua. Trong ví dụ, ta chỉ có hai câu lệnh IF bởi vì ta chỉ xét có hai loại là E và M. Nếu có khoảng 15 loại thì sẽ tốn thời gian và tài nguyên máy tính cho việc tính toán thừa mặc dù lương đã xác định tại câu lệnh IF đầu tiên. Ðây dứt khoát không phải là mã nguồn hiệu quả.

Bây giờ chúng ta xét mã giả  dùng cấu trúc IF lồng nhau đã được sửa đổi trong ví dụ 10.

**Ví dụ 10:**

BEGIN  
  INPUT grade  
  INPUT exp  
  IF grade="E"  
    IF exp=2  
      salary = 2000  
    ELSE  
      IF exp=3  
        salary=3000  
      END IF  
    END IF  
  ELSE  
    IF grade="M"  
      IF exp=2  
        Salary=3000  
      ELSE  
        IF exp=3  
          Salary=4000  
        END IF  
      END IF  
    END IF  
  END IF  
END

Ðoạn mã trên nhìn khó đọc. Tuy nhiên, nó đem lại hiệu suất cao hơn. Chúng ta xét  cùng ví dụ như trên. Nếu công nhân được xếp loại là E và kinh nghiệm là 2 năm thì lương được tính là 2000 ngay trong bước đầu của câu lệnh IF. Sau đó, chương trình sẽ thoát ra vì không cần thực thi thêm bất cứ lệnh ELSE nào. Do đó, không có sự lãng phí và đoạn mã này mang lại hiệu suất cho chương trình và chương trình chạy nhanh hơn

**Vòng lặp**

Một chương trình máy tính là một tập các câu lệnh sẽ được thực hiện tuần tự. Nó có thể lặp lại một số bước với số lần lặp xác định theo yêu cầu của bài toán hoặc đến khi một số điều kiện nhất định được thỏa.

Chẳng hạn, ta muốn viết chương trình hiển thị tên của ta 5 lần. Ta xét mã giả dưới đây.

**Ví dụ 11:**

BEGIN  
DISPLAY "Scooby"  
DISPLAY "Scooby"  
DISPLAY "Scooby"  
DISPLAY "Scooby"  
DISPLAY "Scooby"  
END

Nếu để hiển thị tên ta 1000 lần, nếu ta viết DISPLAY “Scooby” 1000 lần thì rất tốn công sức. Ta có thể tinh giản vấn đề bằng cách viết câu lệnh DISPLAY chỉ một lần, sau đó đặt nó trong cấu trúc vòng lặp, và chỉ thị máy tính thực hiện lặp 1000 lần cho câu lệnh trên.

Ta xem mã giả của cấu trúc vòng  lặp trong  ví dụ 12 như sau:

**Ví dụ 12:**

Do loop 1000 times  
   DISPLAY "Scooby"  
End loop

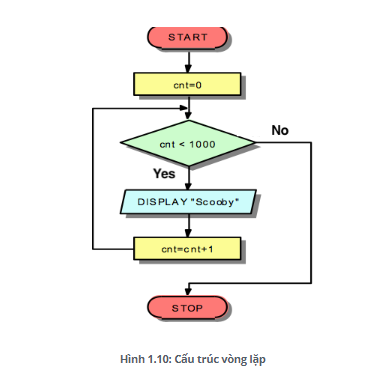
Những câu lệnh nằm giữa Do loop và End loop (trong ví dụ trên là lệnh DISPLAY) được thực thi 1000 lần. Những câu lệnh này cùng với các lệnh **do loop** và **end loop** được gọi là cấu trúc vòng lặp. Cấu trúc vòng lặp giúp lập trình viên phát triển thành những chương trình lớn trong đó có thể yêu cầu thực thi hàng ngàn câu lệnh. **Do loop…end loop** là một dạng thức tổng quát của vòng lặp.

Ví dụ sau là cách viết khác nhưng cũng dùng cấu trúc vòng lặp.

**Ví dụ 13:**

BEGIN  
  cnt=0  
  WHILE (cnt < 1000)  
  DO  
    DISPLAY "Scooby"  
    cnt=cnt+1  
  END DO  
END

Lưu đồ cho mã giả trong ví dụ 13  được vẽ trong Hình 1.10.



Chú ý rằng Hình 1.10 không có ký hiệu đặc biệt nào để biểu diễn cho vòng lặp. Chúng ta dùng ký hiệu phân nhánh để kiểm tra điều kiện và quản lý hướng đi của của chương trình bằng các dòng chảy (flow\_lines).

Last modified: Wednesday, 3 June 2020, 10:52 AM