Chương 4. CÁC LỆNH ĐIỀU KHIỂN TRONG MATLAB

4.1. Các lệnh rẽ nhánh: if ... end, switch ... end

4.1.1. Lệnh if … end

Nhiều khi chúng ta cần những câu lệnh được thực hiện theo một điều kiện nào đó. Trong ngôn ngữ lập trình, logic này được cung cấp bởi cấu trúc ***if … end***. Cú pháp của cấu trúc này như sau:

If < biểu thức điều kiện>

khối các lệnh...

end

Khối các lệnh giữa hai trạng thái ***if*** và ***end*** được thực hiện khi tất biểu thức điều kiện là đúng. Trong trường hợp điều kiện bao gồm các điều kiện con, thì tất cả các điều kiện con được tính và trả về một trạng thái logic của điều kiện.

Ví dụ:

```
apple = input('nhap vao so tao:')
cost = apple*25;
if  apple > 5
    cost = (1-20/100)*cost;
end
```

Trong trường hợp có hai điều kiện thay đổi, cấu trúc ***if-else-end*** là:

if Biểu thức điều kiện

Khối các lệnh được thực hiện nếu điều kiện là đúng

else

Khối các lệnh được thực hiện nếu điều kiện là sai

end

Khi có ba hoặc nhiều điều kiện thay đổi, cấu trúc của nó sẽ là:

if Biểu thức điều kiện 1

Khối các lệnh được thực hiện nếu điều kiện 1 là đúng

elseif Biểu thức điều kiện 2

Khối các lệnh được thực hiện nếu điều kiện 2 là đúng

elseif Biểu thức điều kiện 3

Khối các lệnh được thực hiện nếu điều kiện 3 là đúng

elseif Biểu thức điều kiện 4

. .

.

else

Khối các lệnh được thực hiện nếu không có điều kiện nào đúng.

End

Trong mẫu dạng này thì khi biểu thức điều kiện đầu tiên đúng thì các câu lệnh sau không được kiểm tra nữa, các cấu trúc ***if-else-end*** còn lại được bỏ qua. Hơn nữa câu lệnh ***else*** ở cuối có thể không cần cho vào.

Ví dụ 1: Giải phương trình bậc nhất dạng tổng quát sau: ax + b = 0

Nhận xét:

Khi $a \neq 0$ thì phương trình có nghiệm x = -b/a

Khi a = 0 thì xét hệ số b

Nếu $b \neq 0$: phương trình vô nghiệm

$b = 0$: phương trình có vô số nghiệm

Từ đó ta xây dựng chương trình matlab như sau:

```
a=input('nhap a=');
b=input('nhap b=');
if a~=0
        x=-b/a;
        fprintf('nghiem so x=%9.5g\n',x);
else
if b~=0
        disp('ptvn')
else
        disp('ptvsn')
end
end
```

Tương tự ta xét ví dụ 2: Giải phương trình bậc 2 có dạng tổng quát sau:

$$ax^2 + bx + c = 0 \ (a \neq 0).$$

Lập luận và đưa tới chương trình sau:

```
disp('giai phuong trinh bac hai')
a=input('nhap a=');
b=input('nhap b=');
c=input('nhap c=');
delta=b^2-4*a*c;
d=delta;
if d<0
```

```
            disp('phuong trinh co hai nghiem phuc')
            x1=(-b-j*sqrt(d))/(2*a);
            x2=(-b+j*sqrt(d))/(2*a);
            fprintf('nghiem so x1=%9.5g\n',x1);
            fprintf('nghiem so x2=%9.5g\n',x2);
else
if d==0
            disp('phuong trinh co hai kep')
            x1=-b/(2*a);
            x2=x1;
            fprintf('nghiem kep x1=x2=%9.5g\n',x1);
else
            x1=(-b-sqrt(d))/(2*a);
            x2=(-b+sqrt(d))/(2*a);
            fprintf('nghiem so x1=%9.5g\n',x1);
            fprintf('nghiem so x2=%9.5g\n',x2);
end
end
```

4.1.2. Lệnh switch ... end

Khi một chuỗi các lệnh đánh giá dựa trên một biểu thức thử hoặc biểu thức điều kiện với nhiều giá trị thử khác nhau, người ta thường dùng cấu trúc ***switch-end***. Cấu trúc ***switch-end*** có dạng như sau:

switch biểu thức điều kiện

case giá trị thử 1

khối lệnh 1

case { giá trị thử 2, giá trị thử 3, giá trị thử 4}

khối lệnh 2

otherwise

khối lệnh 3

end

Ở đây biểu thức điều kiện phải là dạng số hoặc dạng chuỗi, nếu biểu thức điều kiện là dạng số thì lệnh ***case*** sẽ thử xem giá trị của biểu thức đó có bằng giá trị thử i hay không. Nếu biểu thức điều kiện là một chuỗi thì lệnh ***case*** sẽ so sánh chuỗi đó với giá trị thử i. Trong ví dụ trước, biểu thức điều kiện được đem so sánh với giá trị thử 1, nếu chúng bằng nhau thì khối lệnh đầu tiện được thực hiện, mà các khối lệnh tiếp theo cho đến trước trạng thái ***end*** được bỏ qua, nếu chúng không bằng nhau thì điều kiện tiếp tục được đem so sánh với giá trị thử 2, giá trị thử 3, giá trị thử 4, nếu một trong các giá trị này bằng biểu thức điều kiện thì khối lệnh 2 được thực hiện. Nếu tất cả các lệnh so sánh của ***case*** đều không

đúng thì khối lệnh 3 được thực hiện. Chú ý rằng trong cấu trúc ***switch-end*** có ít nhất một nhóm lệnh phải được thực hiện. Sau đây là một ví dụ về cấu trúc switch-end.

Ví dụ: Bài toán phân loại học sinh: điểm **9-10** xếp loại giỏi, điểm **7-8** xếp loại **khá**, điểm **5-6** xếp loại **trung bình**; điểm **1,2,3,4** xếp loại **yếu**; nếu điểm vào **không phải** số nguyên nằm giữa **1 và 10** thì thông báo điểm **không hợp lệ**.

```
n = input ('nhap diem: ');
disp('phan loai:');
switch n
        case {0,1,2,3,4}
        disp('loai yeu');
case {5,6}
        disp('loai trung binh');
case {7,8}
        disp('loai kha');
case {9,10}
        disp('loai gioi');
otherwise
        disp('diem khong hop le');
end
```

Nhận xét: Cấu trúc **IF…END** chỉ hữu ích trong trường hợp chỉ có một vài lựa chọn. Còn khi có một số lượng lớn các lựa chọn khả dĩ, thông thường ta sử dụng cấu trúc **SWITCH …END** để thay thế.

4.2. Các lệnh lặp: for ... end, while ... end

4.2.1. Lệnh lặp for … end

MATLAB có phiên bản riêng của nó về vòng lặp "**FOR**" tìm thấy trong các ngôn ngữ máy tính. Nó cho phép một câu lệnh, một nhóm lệnh, được lặp lại một số lần cố định xác định trước.

Cú pháp:

for chỉ số = biểu thức

nhóm lệnh A;

end

Diễn đạt : Nếu thỏa điều kiện thì sẽ thực hiện lệnh của vòng lặp. Đến khi không thỏa điều kiện thì sẽ thoát ra khỏi vòng lặp.

Qui tắc thực hiện:

- Chỉ số của vòng lặp phải là biến.
- Nếu ma trận biểu thức là rỗng thì không thực hiện vòng lặp.

- Nếu biểu thức là một đại lượng vô hướng vòng lặp thực hiện 1 lần và chỉ số nhận giá trị của đại lượng vô hướng.

- Nếu biểu thức là một vector hàng, số lần lặp bằng số phần tử trong vector. Sau mỗi lần lặp chỉ số nhận giá trị tiếp theo của vector.

- Nếu biểu thức là một ma trận thì số vòng lặp bằng số cột của ma trận, sau mỗi vòng lặp thì chỉ số sẽ nhận giá trị cột tiếp theo của ma trận.

- Khi kết thúc vòng lặp chỉ số nhận giá trị cuối cùng.

- Có thể dùng toán tử: Trong vòng lặp for i = chỉ số đầu : gia số : chỉ số cuối. Và số vòng lặp được tính = (chỉ số cuối – chỉ số đầu)/ số gia + 1.

Ví dụ xét chương trình sau:

```
n=input('nhap vao n= ')
   for k=n
       s=1;
       s=s+k
   end
```

Chạy chương trình:

```
nhap vao n= 2
n =
    2
s =
    3
nhap vao n= [3 4 5]
n =
    3    4    5
s =
    4
s =
    5
s =
    6
nhap vao n= [1 2 3,4 5 6]
n =
    1    2    3    4    5    6
s =
    2
s =
    3
s =
    4
s =
    5
s =
```

```
    6
s =
    7
```

Một điểm quan trong là: mỗi vòng lặp **for** phải gắn với từ khóa **end**. Nếu đơn giản vào lệnh

for i = 1:n, x(i) = 0

thì hệ thống sẽ kiên nhẫn chờ nhập các lệnh còn lại trong thân vòng lặp. Không có gì xảy ra đánh vào end.

Ví dụ :Thực hiện hàm .m file tính tổng

S = 1 + 2 + 3 + …+ n

```
disp('tinh tong');
n=input('nhap so hang can tinh tong n=');
s=0;
for i=1:n
    s=s+i;
end
fprintf('tong so s=%2.5g\n',s);
```

4.2.2. Lệnh lặp while … end

MATLAB cũng có phiên bản về vòng lặp "**WHILE**", cho phép một lệnh hoặc nhóm lệnh lặp lại với số lần không xác định, dưới điều khiển của một điều kiện logic.

Được sử dụng khi số lần lặp không được biết trước. Quá trình lặp sẽ chấm dứt khi một điều kiện xác định nào đó được thỏa. Cấu trúc của vòng lặp while như sau:

while expression

statements

end

Trong đó:

expression : Điều kiện của vòng lặp

statements : Lệnh hoặc nhóm lệnh thực thi của vòng lặp.

Diễn đạt: Nếu thỏa điều kiện thì sẽ thực hiện lệnh của vòng lặp. Hàm while được sử dụng khi chưa biết số lần lặp, trong khi hàm for được sử dụng khi đã biết rõ số lần lặp.

Ví dụ: tính n!

n! = 1.2.3.4.5…….n

Chương trình:

```
disp('tinh giai thua');
```

```
n=input('nhap so hang can tinh giai thua n=');
GT=1;
i=1;
while i<=n
   GT=GT*i;
   i=i+1;
end
fprintf('tong so s=%2.5g\n',GT);
```

4.3. Các lệnh chuyển trong vòng lặp

Hai lệnh hữu ích rất hay được sử dụng để điều khiển chu trình tính toán là ***continue*** và ***break***. Trong vòng lặp **for** hay **while**, khi gọi **continue** ngay lập tức chu trình tính chuyển sang bước lặp kế tiếp, mọi lệnh chưa thực hiện của vòng lặp (thuộc về bước lặp hiện tại) sẽ bị bỏ qua.

Lệnh ***break*** còn mạnh hơn: Ngừng vòng lặp đang tính. Lệnh **break** có tác dụng trong cả các cấu trúc rẻ nhánh dùng **if**, **switch**.

Ví dụ sau sẽ minh họa các nội dung vừa đề cập đến trong mục này với nội dung: Cần kiểm tra xem trong các số nguyên thuộc khoảng 3-7, số nào là số nguyên tố. Việc kiểm tra này được thực hiện ở mạch vòng bên ngoài.

```
for m = 3 : 1 : 7,
   for n = 2: 1 : m-1,
      if mod(m,n)~= 0, continue, end
      fprintf('   %2d khong phai la so nguyen to!\n', m)
      break
   end
   if n == m-1,
      fprintf('!! %2d la so nguyen to!\n', m)
   end
end
```

Mạch vòng có nhiệm vụ: Lần lượt chia số cần kiểm tra **m** cho tất cả các số trong khoảng từ **2** đến (**m-1**), sau đó kiểm tra xem số dư **mod(m,n)** của phép chia có khác **0** hay không. Nếu số dư bằng **0**, khi ấy **m** chia hết cho **n** và lệnh **continue** không được gọi, lệnh **fprintf** xuất thông báo lên màn hình. Nếu số dư khác **0**, khi ấy **m** không chia hết cho **n** và lệnh **continue** có hiệu lực, lệnh **fprintf** và **break** bị bỏ qua để chuyển sang kiểm tra vòng lặp mới với **n** lớn hơn. Nếu **m** không chia hết cho các số trong khoảng từ **2** đến (**m-1**), mà chỉ chia hết cho **1** và bản thân **m**, khi ấy **m** là số nguyên tố, và vì vậy vòng lặp phía trong đã được rời bỏ bởi lệnh **break** để tiếp tục các lệnh thuộc vòng lặp phía ngoài. Matlab đưa ra kết quả ngoài màn hình như sau:

```
!!  3 la so nguyen to!
   4 khong phai la so nguyen to!
!!  5 la so nguyen to!
```

```
   6 khong phai la so nguyen to!
!!  7 la so nguyen to!
```

Lệnh chuyển đến cuối m-file

Nếu break được sử dụng ngoài vòng for, while trong phạm vi một script hay function của Matlab, khi ấy script file và function sẽ bị ngừng tại vị trí của break.

Ví dụ: Viết chương trình nhập vào một số n (n>=0) với các trường hợp sau:

a) Nếu n < 0 thì in thông báo bạn nhập sai

b) Nếu n > 0 và lẽ thì tính tổng s_1=1+3+5+...+n, n là số lẽ.

c) Nếu n > 0 và chẵn thì s_2=2+4+6+...+n, n chẵn.

d) Nếu n = 0 dừng chương trình lại.

Giải:

```
n=input('nhap n= ');          % nhap so n
du=rem(n,2);                  % kiem tra so n la chan hay le
                              % neu n le =1, n chan du = 0
if n<0
  fprintf('Ban nhap sai')     % xuat ra thong bao
end
if (n>0) & (du==1)            % neu n>0 và le
  i=1;                        % gan i=1;
  s1=1;                       %gan tong s1=1
  while i<n                   % thuc hien vong lap
    i=i+2;                    % tang i len 2 sau moi vong lap
    s1=s1+i;                  % tinh tong s1 voi gia tri i moi
  end
  s1                          % in ra ket qua sau khi ket thuc vong lap
end
if (n>0) & (du==0)
  i=0;
  s2=0;
  while i<n
    i=i+2;
    s2=s2+i;
  end
  s2
end
if n==0                       % neu n=0
   break                      % lenh ket thuc
end
```

Khi chạy chương trình:

```
» vidu
```

```
nhap n= 5
s1 =
    9
» vidu
nhap n= 4
s2 =
    6
» vidu
nhap n= -6
   Ban nhap sai
» vidu
nhap n= 0
```

4.4. Hàm tạm dừng

Lệnh PAUSE

a) Công dụng:

Dừng chương trình theo ý muốn.

b) Cú pháp:

pause on

pause off

pause (n)

c) Giải thích:

pause on: dừng chương trình, và chờ nhấn 1 phím bất kỳ (trừ các phím điều khiển) chương trình thực hiện tiếp.bre

pause off: tắt chức năng pause.

pause (n): dừng chương trình tại n giây.

d) Ví dụ:

```
        for n = 1 : 3;
    disp('Press any key to continue…')
    pause
  end
  Chạy chương trình:
                Press any key to continue…
      Press any key to continue…
      Press any key to continue…
```

Câu hỏi ôn tập chương 4

Câu 1: Trình bày nội dung của lệnh ***if … end*** trong trường hợp bài toán có hai điều kiện thay đổi và có nhiều điều kiện thay đổi?

Câu 2: Trình bày nội dụng của lệnh ***Switch … end***?

Câu 3: Viết và giải thích cấu trúc của lệnh vòng lặp ***for … end***?

Câu 4: Viết và giải thích cấu trúc của lệnh vòng lặp ***while … end***?
