Fondamenti di Informatica

Execution control

Prof. Emiliano Casalicchio
emiliano.casalicchio@uniroma2.it
Notice

Please send me an email with your name surname and your ID number (numero di maticola)

emiliano.casalicchio@uniroma2.it
Objectives

This chapter discusses techniques for changing the flow of control of a program, which may be necessary for two reasons:

- You may want to execute some parts of the code under certain circumstances only
  - If and switch statements
- You may want to repeat a section of code a certain number of times
  - for and while statements
A code block is a collection of zero or more MATLAB instructions identified for one of two reasons:

1. you wish to execute them only under certain circumstances, or
2. You wish to repeat them a certain number of times

Some languages identify code blocks by enclosing them in braces ({. . .}); others identify them by the level of indentation of the text.

MATLAB uses the occurrence of key command words in the text to define the extent of code blocks:

- if, switch, while, for, case, otherwise, else, elseif, end

Code blocks are identified with blue coloring by the MATLAB text editor. They are not part of the code block, but they serve both

- as instructions on what to do with the code block, and
- as delimiters that define the extent of the code block.
A code block is a collection of zero or more MATLAB instructions identified for one of two reasons:

1. you wish to execute them only under certain circumstances, or
2. You wish to repeat them a certain number of times

\[
\begin{align*}
1. & \text{ if } N=1 \text{ then} \\
2. & \quad \text{max} = v(1); \\
3. & \quad \text{exit;} \\
4. & \quad \text{end} \\
5. & \quad \text{max} = v(1); \\
6. & \quad i = 1; \\
7. & \quad i = i + 1; \\
8. & \quad \text{if } v(i) > \text{max} \text{ then} \\
9. & \quad \quad \quad \text{max} = v(i); \\
10. & \quad \quad \text{end} \\
11. & \quad \text{if } i < N \text{ then} \\
12. & \quad \quad \text{go to 7} \\
13. & \quad \text{end} \\
14. & \text{exit;}
\end{align*}
\]
Concept: Code Blocks

- Some languages identify code blocks by enclosing them in braces ( {. . . }); others identify them by the level of indentation of the text.

C

```c
for(i=1;i<=o;i++)
{
for(j=1;j<=d;j++)
x[i][j]=0;
}
```

Fortran77

```fortran
DO 1, I = 0, 7
   X = EXP(XJ * I * PI / 4)
   IF (AIMAG(X).LT.0) THEN
      PRINT...
   ELSE
      PRINT...
   END IF
   FORMAT (A, I1, A, F10.7, A, F9.7)
CONTINUE
STOP
```
MATLAB uses the occurrence of key command words in the text to define the extent of code blocks:

- if, switch, while, for, case, otherwise, else, elseif, end

MATLAB

```matlab
for i=1:length(log3s_300);
    for iterK=1:length(K)
        start=1+(i-1)*(tau1)*W;
        lambda=log3s_300(start:start+W*tau1-1);
        Xmax=Tmax*max(lambda)/(Tmax*mu-1)*2;
        filename=sprintf('output/%0.0fhK%sWl%0.0f',tau1, strK{iterK},start);
        [x,TC,neval,XFlg, cost, avgViol] = optimalAllocation();
    end
end
```
Conditional Execution in General

- Basic conditional execution requires two things:
  - A logical expression, and
  - A code block
- If the expression is true, the code block is executed.
- Otherwise, execution is resumed at the instruction following the code block
max(v) pseudocode

1. if N=1 then
2.   max=v(1);
3.   exit;
4. end
5. max=v(1);
6. i=1;
7. i=i+1;
8. if v(i)>max then
9.   max=v(i);
10. end
11. if i<N then
12.   go to 7
13. end
14. exit;

MATLAB implementation

N=7;
v=[4 5 2 19 5 7 8];
if N==1
   max=v(1);
   return;
end
... if v(i)>max
   max=v(i);
end
...
Let us consider the problem of the top 10 “qualified” baseball player.

Let us define:

- `yearsInLeague`: vector containing the years in league
- `errorPerYears`: vector containing the errors per year
- `plateApparance`: vector containing the number of apparence per years

A more complex “condition”:

If `yearsInLeague(i) >= 5` && `errorPerYears(i) <= 10` && `plateApparance(i) > 100`

selectThePlayer(i)

Logical End operator

A && B = true if A = true and B = true
A && B = false otherwise
By introducing `elseif` and `else`, we allow for the possibility of either conditional or unconditional execution when a test returns false as illustrated.
if Statements

- The general template for if statements is:

```plaintext
if <logical expression 1>
    <code block 1>
elseif <logical expression 2>
    <code block 2>
    ...
    ...
    ...
elseif <logical expression n>
    <code block n>
else
    <default code block>
end
```
if temperature > 100
    disp('Too hot - equipment malfunctioning.')
elseif temperature > 90
    disp('Normal operating range.')
elseif temperature > 50
    disp('Below desired operating range.')
else
    disp('Too cold - turn off equipment.')
end
%% Listing 4-1  if statement example

day = input(‘enter a day(1-7): ’);
if day == 7  % Saturday
    state = ’weekend’
elseif day == 1  % Sunday
    state = ’weekend’
else
    state = ’weekday’
end

%% Listing 4-2  Script with if statements

grade = input(’what grade? ’);
if grade >= 90
    letter = ’A’
elseif grade >= 80
    letter = ’B’
elseif grade >= 70
    letter = ’C’
elseif grade >= 60
    letter = ’D’
else
    letter = ’F’
end
function testvar(x)
% Display text indicating whether x is a scalar, vector, or matrix
    [m,n] = size(x);
    if m==n && m==1
        disp(' Argument is a scalar')
    elseif m==1 || n==1
        disp(' Argument is a vector')
    else
        disp(' Argument is a matrix')
    end

Logical Or operator
A || B=true if at least A=true or B=true
A || B = false otherwise
General Observations

- A logical expression is any statement that returns a logical result.
- If that result is a logical vector, \( v \), the if statement behaves as:

\[
\text{if all}(v)
\]

- While indentation has no effect on the logical flow, it helps to clarify the logical flow. The MATLAB editor automatically creates suitable indentation as you type.

```matlab
%% Listing 4-3  the if statement with a logical vector
A = [true true false]
if A
    % will not execute
end
A(3) = true;
if A
    % will execute
end
```
4.4 switch Statements

- The template for a switch statement is:

```
switch <parameter>
    case <case specification 1>
        <code block 1>
    case <case specification 2>
        <code block 2>
        .
        .
    case <case specification n>
        <code block n>
otherwise
    <default code block>
end
```
General Observations

- The switch statement is looking for the parameter to have an exact match to one of the cases.
- One case specification may have multiple values enclosed in braces( {...}).
- The default case catches any values of the parameter other than the specified cases.
- The default case should trap bad parameter values.

%% Listing 4-4  Example of a switch statement
leapYear = false;
month = input('enter a month (1-12): ');
switch month
    case {9, 4, 6, 11}% Sept, Apr, June, Nov
        days = 30;
    case 2% Feb
        if leapYear
            days = 29;
        else
            days = 28;
        end
    case {1, 3, 5, 7, 8, 10, 12}% other months
        days = 31;
    otherwise
        error('bad month index')
end
days
if...else versus switch

Problem statement

if variable interval is less than one, set the value of xinc to interval/10; otherwise, set the value of xinc to 0.1.

IF...ELSE

if interval < 1
  xinc = interval/10;
else
  xinc = 0.1;
end

SWITCH

switch interval < 1
  case 1
    xinc = interval/10;
  case 0
    xinc = 0.1;
end
Iteration in General

Iteration allows controlled repetition of a code block. Control statements at the beginning of the code block specify the manner and extent of the repetition:

- The for loop is designed to repeat its code block a fixed number of times and largely automates the process of managing the iteration.

- The while loop is more flexible in character. Its code block can be repeated a variable number of times. It is much more of a “do-it-yourself” iteration kit.
for Loops

The template for a for loop is:

```
for <variable> = <vector>
  <code block>
end
```

more in general

```
for variable = expr
  <code block>
end
```

The for loop automatically sets the value of the variable to each element of the vector in turn and executes the code block with that value.
find the max value in a vector

%%Listing 4–6   for statement using
indexing
A = floor(rand(1,10)*100)
theMax = A(1);
theIndex = 1;
for index = 1:length(A)
    if A(index) > theMax
        theMax = A(index);
        theIndex = index;
    end
end
fprintf('the max value in A ... is %d at %d\n', ... 
    theMax, theIndex);

1. if N=1 then
2.    max=v(1);
3.    exit;
4. end
5. max=v(1);
6. i=1;
7. i=i+1;
8. if v(i)>max then
9.    max=v(i);
10. end
11. if i<N then
12.   go to 7
13. end
14. exit;
find the max value in a vector

%%Listing 4-5   Example of a for statement

A = [6 12 6 91 13 6] % initial vector

theMax = A(1); % set initial max value

for x = A % iterate through A
  if x > theMax % test each element
    theMax = x;
  end
end

fprintf('max(A) is %d\n', theMax);
while Loops

The code block will be repeated as long as the logical expression returns true.

The while loop template is:

```plaintext
<initialization>
while <logical expression>
    <code block>
    % must make some changes
    % to enable the loop to terminate
end
```
% Script to compute ax^2 + bx + c
disp('Quadratic ax^2+bx+c evaluated')
disp('for user input a, b, c; and x')
a=1; b=1; c=1; x=0;
while a~=0 || b~=0 || c~=0 || x~=0
    disp('Enter a=b=c=x=0 to terminate')
a = input('Enter value of a: ?');
b = input('Enter value of b: ?');
c = input('Enter value of c: ?');
x = input('Enter value of x: ?');
if a==0 && b==0 && c==0 && x==0
    break
end
quadratic = a*x^2 + b*x + c;
disp('Quadratic result:')
disp(quadratic)
end
Give a tank as shown, how do you calculate the volume of liquid? The answer of course is “it depends on h.”
If $h \leq r$, do one calculation; otherwise if $h < (H-r)$ do a second; Otherwise if $h \leq H$, do a third; Otherwise there is an error!
The Solution

```matlab
if h < r
    v = (1/3)*pi*h.^2.*(3*r-h);
elseif h < H-r
    v = (2/3)*pi*r^3 + pi*r^2*(h-r);
elseif h <= H
    v = (4/3)*pi*r^3 + pi*r^2*(H-2*r) ...
        - (1/3)*pi*(H-h)^2*(3*r-H+h);
else
    disp('liquid level too high')
    continue
end
fprintf( ...
    'rad %0.2f ht %0.2f level %0.2f vol %0.2f\n', ...
    r, H, h, v);
```
Summary

This chapter presented techniques for changing the flow of control of a program for condition execution and repetitive execution:

■ The most general conditional form is the if statement, with or without the accompanying elseif and else statements
■ The switch statement considers different cases of the values of a countable variable
■ A for loop in its most basic form executes a code block for each of the elements of a vector
■ A while loop repeats a code block a variable number of times, as long as the conditions specified for continuing the repetition remain true
Homeworks

- Implement
  - a function that return the min value of a vector and related index
  - a function that find a value in a vector and return the related index
  - implement `linspace (Xmin, Xmax, N)`
  - implement the sum of the element of a vector
  - implement the mean of the element of a vector