مقدمه ای بر برنامه نویسی با MATLAB

سعید سهیلی
استادیار گروه مکانیک
دانشگاه آزاد اسلامی مشهد


بسم الله الرحمن الرحیم

فصل سوم

عبارات شاخه ای و طراحی برنامه

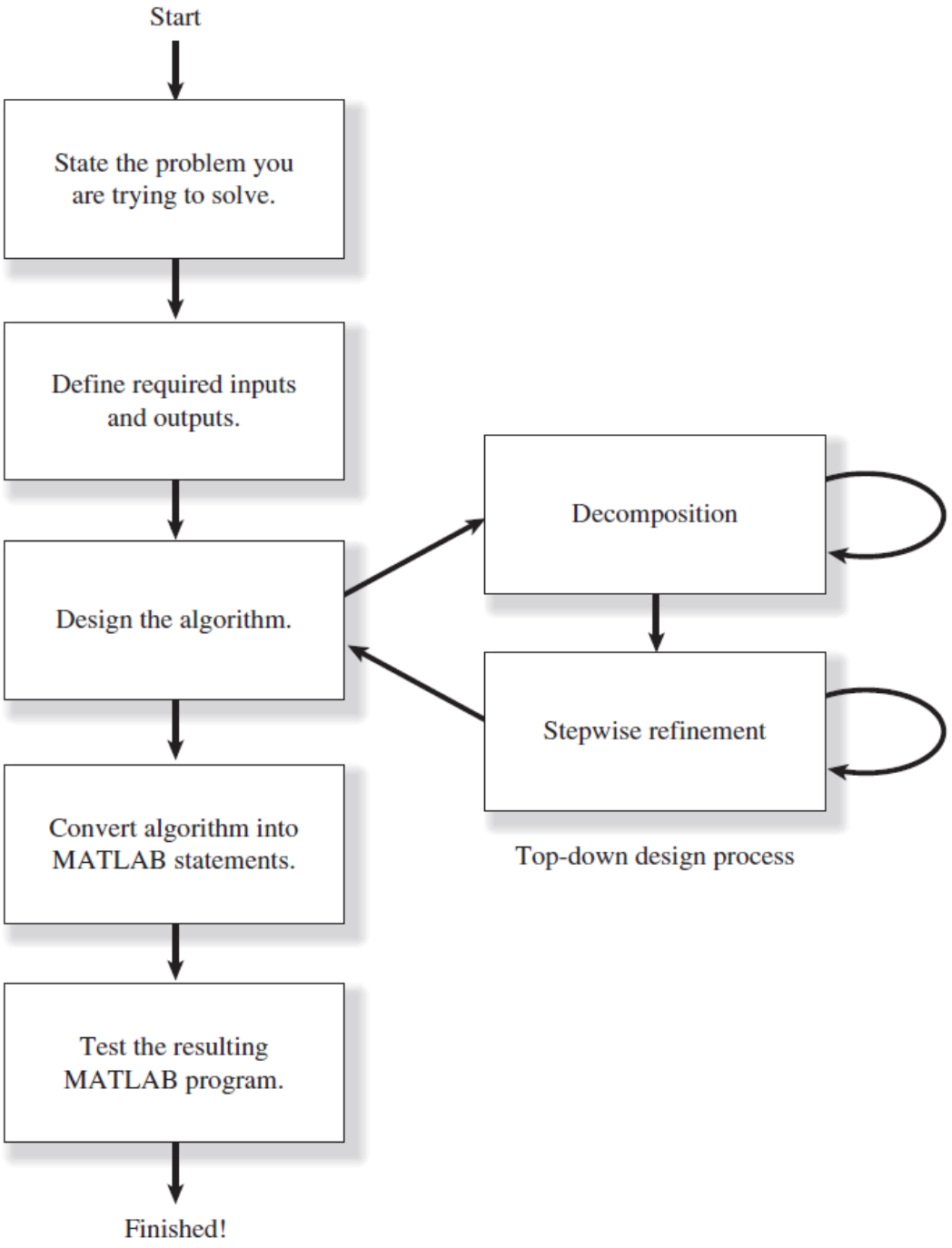
مقدمه

- برنامه های ترتیبی: برنامه هایی که داده های ورودی را خوانده و پس از پردازش داده و محاسبه جواب، پاسخ را چاپ می کند.
- در این برنامه ها نمی توان بخشی از برنامه را تکرار کرد و یا به صورت انتخابی اجرا نمود.
- برنامه ها به صورت پشت سرهم اجرا می شوند. (script files)

- شاخه ها (branches): قسمت های مشخصی از برنامه را برای اجرا انتخاب می کنند.
- حلقه ها(loops): تکرار قسمت خاصی از برنامه

روش طراحی بالا به پایین

- برنامه نویسی آسان است، اینکه بدانیم چه برنامه ای می خواهیم بنویسیم مشکل است
- سخت ترین قسمت کار *درک مسئله* است
- طراحی بالا به پایین روندی است که با یک کار بزرگ آغاز می شود و سپس این کار به زیرمجموعه های کوچکتر و قابل درک تر تقسیم می شود.
- هر قسمت کوچک به طور مستقل کد نویسی می شود
- در نهایت زیرمجموعه ها با هم ترکیب می شوند.


Start
State the problem you are trying to solve.
Define required inputs and outputs.
Design the algorithm.
Convert algorithm into MATLAB statements.
Test the resulting MATLAB program.
Finished!
Decomposition
Stepwise refinement
Top-down design process

1. ابعاد مختلف برنامه را برای خود کاملا روشن کنید.
   - به عنوان مثال، برنامه ای طراحی کنید که یک دستگاه معادلات خطی با ثابت های حقیقی شامل ۲۰ معادله و ۲۰ مجهول را حل کنید.
2. ورودی های لازم برای برنامه و نیز خروجی های برنامه را تعریف نمایید.
   - ضرائب معادلات به عنوان ورودی های مورد نیاز
3. الگوریتمی که قصد دارید در برنامه به کار گیرید را طراحی نمایید.
   - الگوریتم یک روش قدم به قدم برای پیدا کردن راه حل مساله است.
   - طراح مساله را به زیر مجموعه هایی تقسیم می کند
   - برای هر زیر مجموعه توصیفی کلی از کاری که باید انجام شود، آورده می شود.
   - حل دستی مساله در این روند بسیار مفید است.

4. الگوریتم را به دستورات MATLAB تبدیل کنید
- اگر تجزیه و روند برنامه درست انجام شود در این مرحله عبارت دستوری به کد MATLAB به سادگی تبدیل می شوند.

5. برنامه نهایی MATLAB را امتحان کنید.
- این قسمت جان برنامه نویس را به لبش می رساند.
- ابتدا اجزا برنامه به صورت مجزا امتحان می شوند.
- باید مطمئن شویم برنامه برای تمام ورودی های مجاز درست عمل می کند.

- در یک برنامه بزرگ ۱/۳ زمان صرف برنامه ریزی مراحل کارها (مراحل۱تا۳)، ۱/۶ زمان صرف نوشتن برنامه (مرحله۴) و نیمی از وقت صرف آزمایش و رفع اشتباهات برنامه می شود.

استفاده از شبه کد (Pseudocode)

- در طراحی برنامه لازم است الگوریتمی برای برنامه تعریف شود.
- توضیحات الگوریتم باید به صورت استاندارد بیان شود.
- شبه کد ترکیبی از MATLAB و زبان انگلیسی است که در هر خط ایده و کار مورد نیاز را به صورت مختصر و قابل درک بیان می کند.
- شبه کد در ساماندهی افکار قبل از تبدیل به کد MATLAB کمک می کند.

```
Prompt user to enter temperature in degrees Fahrenheit
Read temperature in degrees Fahrenheit (temp_f)
temp_k in kelvins ← (5/9) * (temp_f - 32) + 273.15
Write temperature in kelvins
```

عملگرهای مقایسه ای و منطقی

- عملکرد بسیاری از ساختارهای شاخه ای توسط عباراتی که نتیجه آن درست (۱) یا غلط (.) است کنترل می شود.
- **عملگرهای مقایسه ای**: این عملگرها در جدول زیر خلاصه شده اند.

Operator	Operation
==	Equal to
~=	Not equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to

Operation	Result
`3 < 4`	`true (1)`
`3 <= 4`	`true (1)`
`3 == 4`	`false (0)`
`3 > 4`	`false (0)`
`4 <= 4`	`true (1)`
`'A' < 'B'`	`true (1)`

- نماد == یک عملگر مقایسه ای است
- در حالی که نماد = عبارت موجود در سمت راست نماد را به متغیر سمت چپ نماد نسبت می دهد.
- عملگرهای مقایسه ای بعد از تمامی عملگرهای عددی محاسبه می شوند.

```
7 + 3  <  2 + 11
(7 + 3) < (2 + 11)
```

```
>> 5>8
ans =
     0
```

Checks if 5 is larger than 8.

Since the comparison is false (5 is not larger than 8) the answer is 0.

```
>> a=5<10
a =
     1
```

Checks if 5 is smaller than 10, and assigns the answer to a.

Since the comparison is true (5 is smaller than 10) the number 1 is assigned to a.

```
>> y=(6<10)+(7>8)+(5*3==60/4)
```

Using relational operators in math expression.

Equal to 1 since 6 is smaller than 10.

Equal to 0 since 7 is not larger than 8.

Equal to 1 since 5*3 is equal to 60/4.

```
y =
     2
>> b=[15 6 9 4 11 7 14]; c=[8 20 9 2 19 7 10];
```

Define vectors b and c.

```
>> d=c>=b
```

Checks which c elements are larger than or equal to b elements.

```
d =
     0     1     1     0     1     1     0
```

Assigns 1 where an element of c is larger than or equal to an element of b.

```
>> b == c
ans =
     0     0     1     0     0     1     0
>> b~=c
ans =
     1     1     0     1     1     0     1
>> f=b-c>0
f =
     1     0     0     1     0     0     1
>> A=[2 9 4; -3 5 2; 6 7 -1]
A =
     2     9     4
    -3     5     2
     6     7    -1
>> B=A<=2
```

Checks which b elements are equal to c elements.

Checks which b elements are not equal to c elements.

Subtracts c from b and then checks which elements are larger than zero.

Define a 3×3 matrix A.

Checks which elements in A are smaller than or equal to 2. Assigns the results to matrix B.

احتیاط در استفاده از == و =~

- عملگر تساوی == زمانی که دو مقدار مقایسه شده با هم برابر باشند، مقدار یک را بر می گرداند و اگر متفاوت باشند مقدار صفر را بر می گرداند.
- عملگر نابرابری =~ زمانی که دو مقدار مقایسه شده با هم برابر باشند، مقدار صفر را بر می گرداند و اگر متفاوت باشند مقدار یک را بر می گرداند.
- به علت خطای گردکردن دو عدد که از نظر تئوری مساوی اند می توانند تا حدودی با هم تفاوت داشته باشند.

```
» a = 0;
» b = sin(pi);
» a == b
ans =
     0
```

- بخاطر خطای گرد کردن، sin(pi) بجای صفر عدد 1.2246*1.0e-16 در نظر گرفته شده بنابراین a و b با هم برابر نیستند.
- بجای تساوی از عملگر زیر می توان استفاده نمود.

```
» abs(a - b) < 1.0E-14
ans =
     1
```

.

l_1 op l_2

- **عملگرهای منطقی:**

Operator	Operation
&	Logical AND
&&	Logical AND with shortcut evaluation
|	Logical Inclusive OR
||	Logical Inclusive OR with shortcut evaluation
xor	Logical Exclusive OR
~	Logical NOT

Inputs		and		or		xor	not
l_1	l_2	l_1 & l_2	l_1 && l_2	l_1 | l_2	l_1 || l_2	xor(l_1, l_2)	~l_1
0	0	0	0	0	0	0	1
0	1	0	0	1	1	1	1
1	0	0	0	1	1	1	0
1	1	1	1	1	1	0	0

Logical ANDs

- اپراتور AND درست است (۱) اگر و فقط اگر هر دو ورودی درست باشند.
- با استفاده از && اگر عبارت اول غلط باشد دیگر عبارت دوم ارزیابی نمی شود ولی & هر دو را ارزیابی می کند.
- && فقط بین عبارت اسکالر استفاده می شود در حالی که & برای هم اسکالر و هم آرایه کاربرد دارد.
- مثال) بررسی اینکه تقسیم دو عدد بزرگتر از ۱۰ است و تقسیم بر صفر نداریم

```
>> b=2; a= 52;
>> x = (b ~= 0) && (a/b > 10.0)

x =

     1

>> b=0; a= 52;
>> x = (b ~= 0) && (a/b > 10.0)

x =

     0

>>
```

- **Logical Inclusive Ors**
- حاصل عبارت OR درست است (۱) اگر یکی یا هر دو عبارت ورودی درست باشند.
- با استفاده از || اگر عبارت اول درست باشد دیگر عبارت دوم ارزیابی نمی شود. ولی | هر دو عبارت را بررسی می کند.
- || فقط بین عبارت اسکالر استفاده می شود در حالی که | برای هم اسکالر و هم آرایه کاربرد دارد.

Logical Exclusive OR

- حاصل این عبارت درست است اگر و فقط اگر یکی از عبارات درست و عبارت دیگر غلط باشد.
- درصورتی که هر دو عبارت درست و یا هر دو غلط باشد حاصل غلط می شود.

```
>> xor(1>2,5~=6)

ans =

     1

>> xor(1<2,5~=6)

ans =

     0
```

```
>> a=5|0
a =
     1
>> ~25
ans =
     0
>> t=25*((12&0)+(~0)+(0|5))
t =
    50
>> x=[9 3 0 11 0 15]; y=[2 0 13 -11 0 4];
>> x&y
ans =
     1     0     0     1     0     1
>> z=x|y
z =
     1     1     1     1     0     1
>> ~(x+y)
ans =
     0     0     0     1     1     0
```

5 OR 0 (assign to variable a).

1 is assigned to a since at least one number is true (nonzero).

NOT 25.

The outcome is 0 since 25 is true (nonzero) and the opposite is false.

Using logical operators in a math expression.

Define two vectors x and y.

The outcome is a vector with 1 in every position where both x and y are true (nonzero elements), and 0s otherwise.

The outcome is a vector with 1 in every position where either or both x and y are true (nonzero elements), and 0s otherwise.

The outcome is a vector with 0 in every position where the vector x + y is true (nonzero elements), and 1 in every position where x + y is false (zero elements).

مثال کاربردی

- بیشترین دمای روزانه در طی یک ماه در ادامه آمده است. از اپراتورهای منطقی برای محاسبه موارد زیر استفاده کنید.
  - تعداد روزهایی که دما بالای ۷۵ درجه بوده است.
  - تعداد روزهایی که دما بین ۶۵ و ۸۰ درجه بوده است.
  - تعداد روزهایی که دما بین ۵۰ و ۶۰ درجه بوده است.

```
T=[58 73 73 53 50 48 56 73 73 66 69 63 74 82 84 ...
   91 93 89 91 80 59 69 56 64 63 66 64 74 63 69];
```

```
T=[58 73 73 53 50 48 56 73 73 66 69 63 74 82 84 ...
   91 93 89 91 80 59 69 56 64 63 66 64 74 63 69];
Tabove75=T>=75;
NdaysTabove75=sum(Tabove75)
Tbetween65and80=(T>=65)&(T<=80);
NdaysTbetween65and80=sum(Tbetween65and80)
datesTbetween50and60=find((T>=50)&(T<=60))
```

A vector with 1s at addresses where T >= 75.

Add all the 1s in the vector Tabove75.

A vector with 1s at addresses where T >= 65 and T <= 80.

Add all the 1s in the vector Tbetween65and80.

The function find returns the address of the elements in T that have values between 50 and 60.

سلسله مراتب انجام عملیات

1. تمامی عملگرهای عددی
2. تمامی عملگرهای مقایسه ای از چپ به راست (==, ~=, >, >=, <, <=)
3. تمامی عملگرهای ~
4. تمامی عملگرهای & and &&
5. تمامی عملگرهای |, ||, and xor

```
>> 3+4<16/2
```
+ and / are executed first.

```
ans =
     1
```
The answer is 1 since 7 < 8 is true.

```
>> 3+(4<16)/2
```
4 < 16 is executed first, and is equal to 1, since it is true.

```
ans =
    3.5000
```
3.5 is obtained from 3 + 1/2.

```
>> x=-2; y=5;
```

Define variables x and y.

```
>> -5<x<-1
ans =
     0
```

This inequality is correct mathematically. The answer, however, is false since MATLAB executes from left to right. $-5 < x$ is true (=1) and then $1 < -1$ is false (0).

```
>> -5<x & x<-1
ans =
     1
```

The mathematically correct statement is obtained by using the logical operator &. The inequalities are executed first. Since both are true (1), the answer is 1.

```
>> ~(y<7)
ans =
     0
```

$y < 7$ is executed first, it is true (1), and ~1 is 0.

```
>> ~y<7
ans =
     1
```

~y is executed first, y is true (1) (since y is nonzero), ~1 is 0, and $0 < 7$ is true (1).

```
>> ~((y>=8) | (x<-1))
ans =
     0
```

$y >= 8$ (false), and $x < -1$ (true) are executed first. OR is executed next (true). ~ is executed last, and gives false (0).

```
>> ~(y>=8) | (x<-1)
ans =
     1
```

$y >= 8$ (false), and $x < -1$ (true) are executed first. NOT of ($y >= 8$) is executed next (true). OR is executed last, and gives true (1).

توابع منطقی

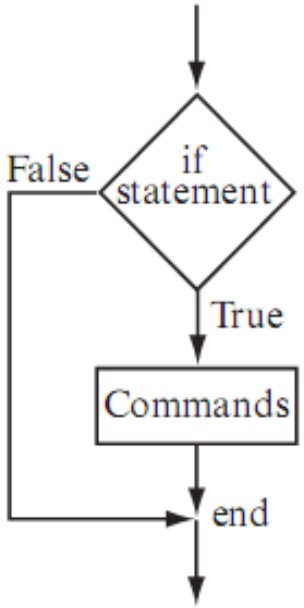
- درصورت درست بودن شرایط توابع مقدار یک حاصل می شود.
- از این توابع به همراه عملگرهای مقایسه ای و منطقی برای کنترل عملیات شاخه ها و حلقه ها استفاده می شود.

Function	Purpose
`false`	Returns a `false` (0) value.
`ischar(a)`	Returns `true` if `a` is a character array and `false` otherwise.
`isempty(a)`	Returns `true` if `a` is an empty array and `false` otherwise.
`isinf(a)`	Returns `true` if the value of `a` is infinite (`Inf`) and `false` otherwise.
`isnan(a)`	Returns `true` if the value of `a` is `NaN` (not a number) and `false` otherwise.
`isnumeric(a)`	Returns `true` if `a` is a numeric array and `false` otherwise.
`logical`	Converts numerical values to logical values; if a value is nonzero, it is converted to `true`. If it is zero, it is converted to `false`.
`true`	Returns a `true` (1) value.

شاخه ها

- به وسیله شاخه ها می توان قسمت مشخصی از برنامه را اجرا نمود.
- **ساختار if بررسی یک شرط**

Flowchart




```
.....
.....  MATLAB program.
.....
if conditional expression
      .......
      .......      A group of
      .......      MATLAB commands.
end
.....
.....  MATLAB program.
.....
```

Sample Problem 6-2: Calculating worker's pay

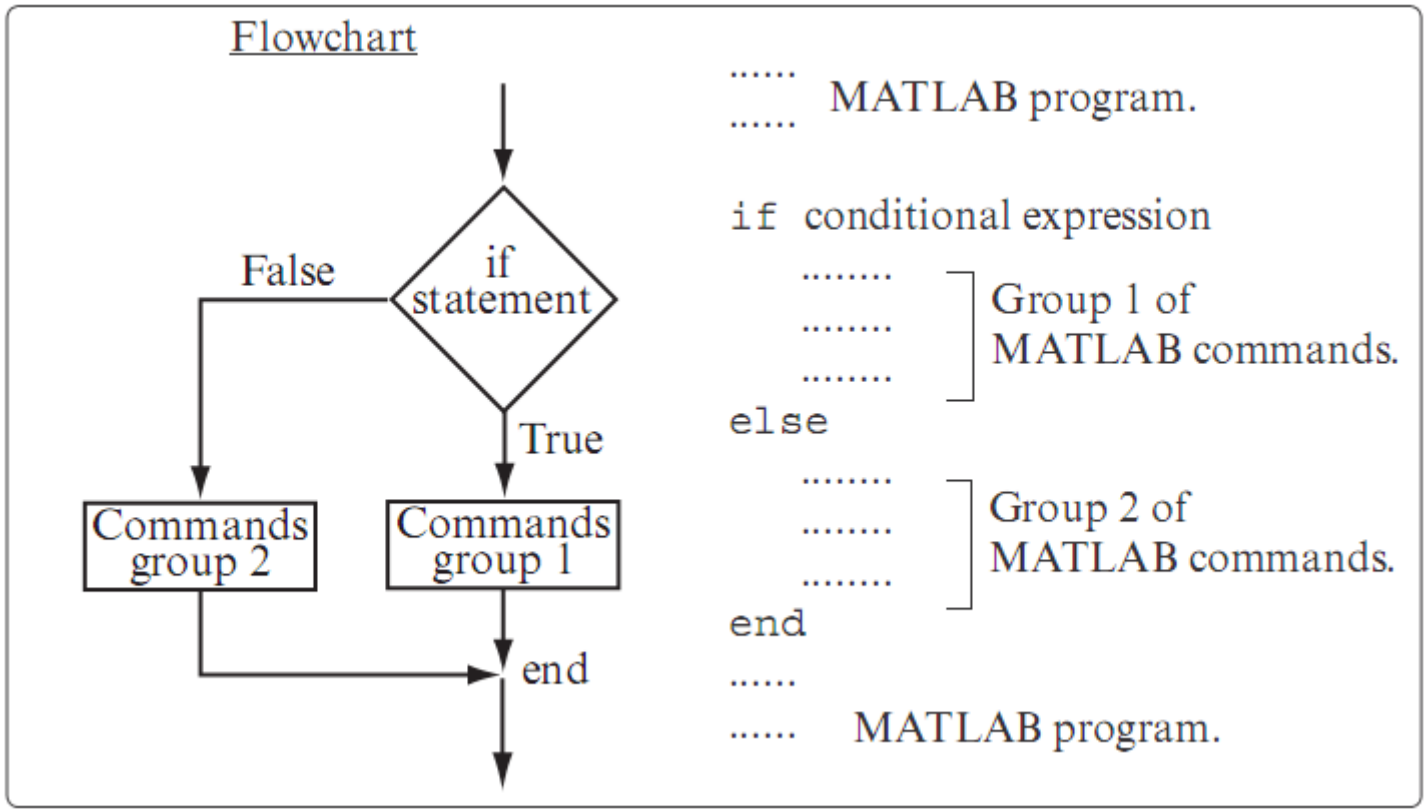
A worker is paid according to his hourly wage up to 40 hours, and 50% more for overtime. Write a program in a script file that calculates the pay to a worker. The program asks the user to enter the number of hours and the hourly wage. The program then displays the pay.

- دستمزد یک کارگر با توجه به ساعت کاریش تا ۴۰ ساعت پرداخت می شود. برای اضافه کاری ۵۰٪ دستمزد ساعتی پرداخت می شود.
- برنامه ساعت کاری و دستمزد یک ساعت را دریافت و پرداخت نهایی را نمایش می دهد.

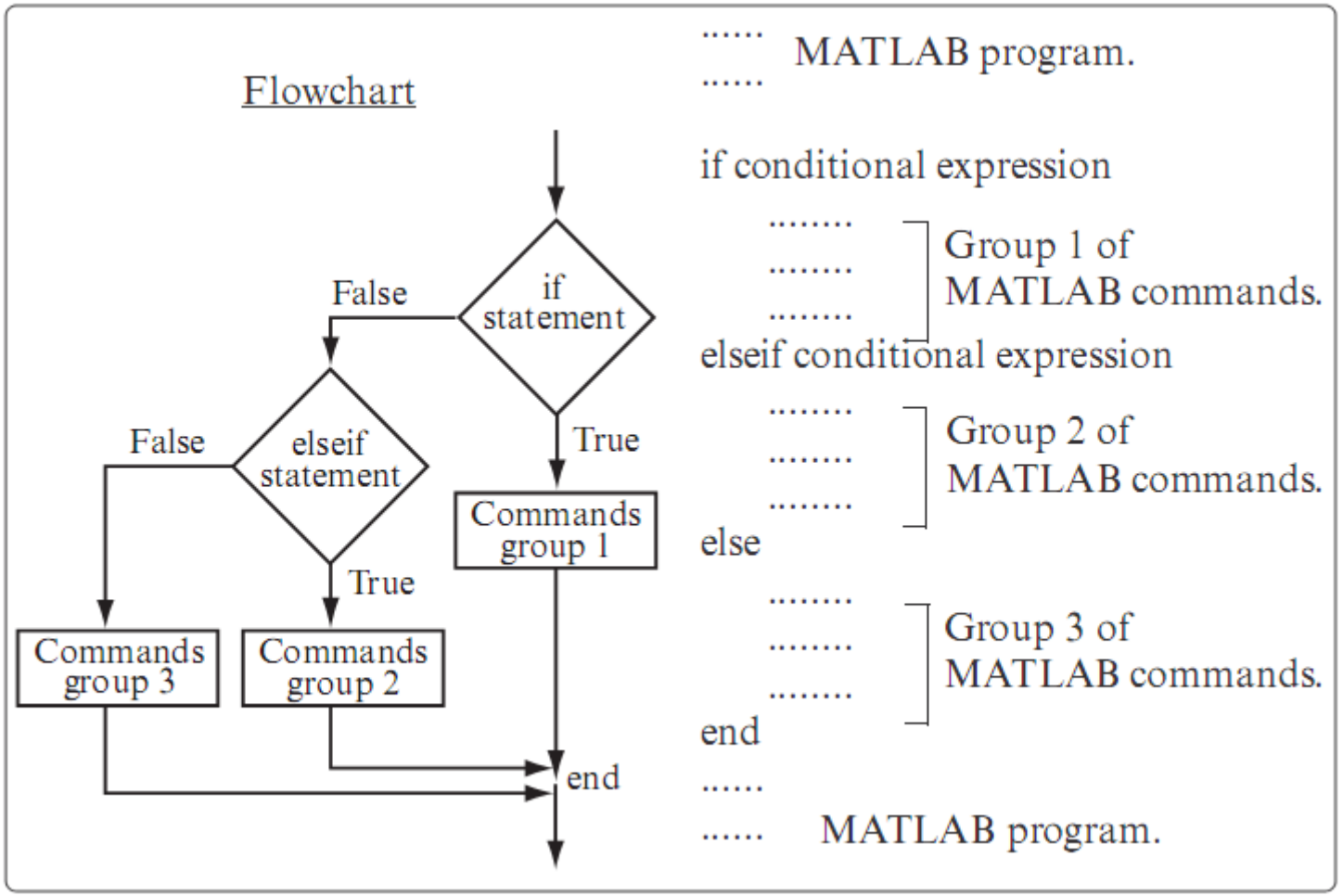
```
t=input('Please enter the number of hours worked  ');
h=input('Please enter the hourly wage in $  ');
Pay=t*h;
if t>40
    Pay=Pay+(t-40)*0.5*h;
end
fprintf('The worker''s pay is  $ %5.2f',Pay)
```

```
Please enter the number of hours worked  35
Please enter the hourly wage in $  8
The worker's pay is  $ 280.00
```

- **ساختار if بررسی یک شرط و عدم درستی آن**



- **ساختار if در حالت کلی**



مثال) برنامه ای طراحی کنید که ریشه های هر نوع معادله درجه دومی را محاسبه کند

1. مساله را کاملاً مشخص نمایید

- برنامه ای مورد نیاز است که ریشه های (حقیقی، تکراری و مختلط) هر معادله درجه دومی را محاسبه کند

2. ورودی ها و خروجی ها را مشخص نمایید

- ورودی ها، ضرایب معادله درجه دو a, b, c هستند

$$ax^2 + bx + c = 0$$

- خروجی ها ریشه های معادله درجه دو هستند.

3. الگوریتم مورد نظر طراحی شود.

- این مساله به سه قسمت اصلی تقسیم می شود.

```
Read the input data
Calculate the roots
Write out the roots
```

- با توجه به علامت دلتا سه روش مختلف برای محاسبه ریشه وجود دارد
- بنابراین از یک ساختار سه شاخه ای if استفاده می شود.
- شبه کد به صورت زیر نوشته شده است.

```
Prompt the user for the coefficients a, b, and c.
Read a, b, and c
discriminant ← b^2 - 4 * a * c
if discriminant > 0
   x1 ← ( -b + sqrt(discriminant) ) / ( 2 * a )
   x2 ← ( -b - sqrt(discriminant) ) / ( 2 * a )
   Write msg that equation has two distinct real roots.
   Write out the two roots.
```

```
elseif discriminant == 0
   x1 ← -b / ( 2 * a )
   Write msg that equation has two identical real roots.
   Write out the repeated root.
else
   real_part ← -b / ( 2 * a )
   imag_part ← sqrt ( abs ( discriminant ) ) / ( 2 * a )
   Write msg that equation has two complex roots.
   Write out the two roots.
end
```

4. تبدیل شبه کد به عبارات Matlab

```
%  Script file: calc_roots.m
%
%  Purpose:
%     This program solves for the roots of a quadratic equation
%     of the form a*x**2 + b*x + c = 0. It calculates the answers
%     regardless of the type of roots that the equation possesses.
%
%  Record of revisions:
%       Date          Programmer          Description of change
%       ====          ==========          =====================
%     01/02/10      S. J. Chapman         Original code
%
```

```
% Script file: calc_roots.m
disp ('This program solves for the roots of a quadratic ');
disp ('equation of the form A*X^2 + B*X + C = 0. ');
a = input ('Enter the coefficient A: ');
b = input ('Enter the coefficient B: ');
c = input ('Enter the coefficient C: ');
% Calculate discriminant
discriminant = b^2  4 * a * c;
% Solve for the roots, depending on the value of the discriminant
if discriminant > 0 % there are two real roots, so...
x1 = ( -b + sqrt(discriminant) ) / ( 2 * a );
x2 = ( -b - sqrt(discriminant) ) / ( 2 * a );
disp ('This equation has two real roots:');
fprintf ('x1 = %f\n', x1);
fprintf ('x2 = %f\n', x2);
elseif discriminant == 0 % there is one repeated root, so...
x1 = ( -b ) / ( 2 * a );
disp ('This equation has two identical real roots:');
fprintf ('x1 = x2 = %f\n', x1);
else % there are complex roots, so ...
real_part = ( -b ) / ( 2 * a );
imag_part = sqrt ( abs ( discriminant ) ) / ( 2 * a );
disp ('This equation has complex roots:');
fprintf('x1 = %f +i %f\n', real_part, imag_part );
fprintf('x1 = %f -i %f\n', real_part, imag_part );
end
```

5. امتحان برنامه

- باید هر سه مسیر برنامه امتحان شود
- ریشه های حقیقی مجزا
- (x-2)*(x+4) = x^2+ 2x -8
- ریشه های حقیقی تکراری
- (x-6)*(x-6) = x^2-12x+36
- ریشه های مختلط
- [x-(7+3i)][x-(7-3i)] = x^2-14x+58

مثال) محاسبه تابعی با دو متغیر

- برنامه ای بنویسید که مقدار تابع f(x,y) را برای دو مقدار مشخص حساب نماید.

$$f(x,y) = \begin{cases} x + y & x \geq 0 \text{ and } y \geq 0 \\ x + y^2 & x \geq 0 \text{ and } y < 0 \\ x^2 + y & x < 0 \text{ and } y \geq 0 \\ x^2 + y^2 & x < 0 \text{ and } y < 0 \end{cases}$$

1. مساله را کاملاً مشخص نمایید
   - مقدار تابع بسته به علامت دو متغیر x,y محاسبه می شود.
2. مشخص کردن ورودی ها و خروجی ها
   - ورودی ها متغیرهای x,y
   - خروجی: مقدار تابع

3. طراحی الگوریتم

```
Read the input values x and y
Calculate f(x,y)
Write out f(x,y)
```

- چهار راه برای محاسبه تابع وجود دارد پس از چهار شاخه if استفاده می شود

```
Prompt the user for the values x and y.
Read x and y
if x ≥ 0 and y ≥ 0
   fun ← x + y
elseif x ≥ 0 and y < 0
   fun ← x + y^2
elseif x < 0 and y ≥ 0
   fun ← x^2 + y
else
   fun ← x^2 + y^2
end
Write out f(x,y)
```

4. تبدیل الگوریتم به عبارات Matlab

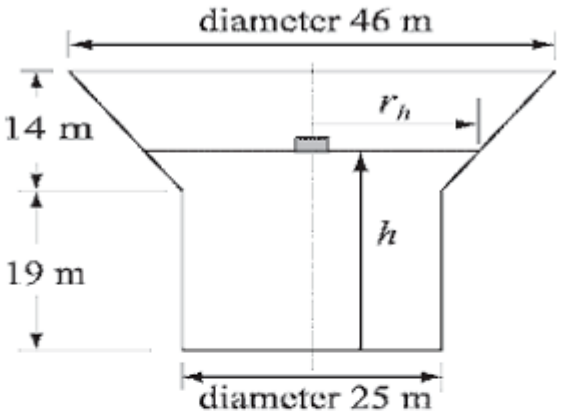
5. امتحان برنامه

$$f(2,3) = 2 + 3 = 5$$
$$f(2,-3) = 2 + (-3)^2 = 11$$
$$f(-2,3) = (-2)^2 + 3 = 7$$
$$f(-2,-3) = (-2)^2 + (-3)^2 = 13$$

مثال) محاسبه حجم آب

Sample Problem 6-3: Water level in water tower

The tank in a water tower has the geometry shown in the figure (the lower part is a cylinder and the upper part is an inverted frustum of a cone). Inside the tank there is a float that indicates the level of the water. Write a MATLAB program that determines the volume of the water in the tank from the position (height h) of the float. The program asks the user to enter a value of h in m, and as output displays the volume of the water in m^3.



Solution

For $0 < h < 19$ m the volume of the water is given by the volume of a cylinder with height h: $V = \pi 12.5^2 h$.

For $19 \le h \le 33$ m the volume of the water is given by adding the volume of a cylinder with $h = 19$ m, and the volume of the water in the cone:

$$V = \pi 12.5^2 \cdot 19 + \frac{1}{3}\pi(h-19)(12.5^2 + 12.5 r_h + r_h^2)$$

where $r_h = 12.5 + \frac{10.5}{14}(h-19)$.

The program is:

```
% The program calculates the volume of the water in the
water tower.
h=input('Please enter the height of the float in meter  ');
if h > 33
   disp('ERROR. The height cannot be larger than 33 m.')
elseif h < 0
   disp('ERROR. The height cannot be a negative number.')
elseif h <= 19
   v = pi*12.5^2*h;
   fprintf('The volume of the water is %7.3f cubic meter.\n',v)
else
   rh=12.5+10.5*(h-19)/14;
   v=pi*12.5^2*19+pi*(h-19)*(12.5^2+12.5*rh+rh^2)/3;
   fprintf('The volume of the water is %7.3f cubic meter.\n',v)
end
```

ساختار switch

- فرم دیگری از ساختارهای شاخه ای
- براساس یک مقدار صحیح، یک کاراکتر و یا یک عبارت منطقی، بلوکی از دستورات را برای اجرای برنامه انتخاب می کند.

```
switch (switch_expr)
case case_expr_1
   Statement 1          }
   Statement 2          }  Block 1
   ...                  }
case case_expr_2
   Statement 1          }
   Statement 2          }  Block 2
   ...                  }
...
otherwise
   Statement 1          }
   Statement 2          }  Block n
   ...                  }
end
```

- اگر مقدار witch_expr برابر با case_expr_2 بلوک دوم اجرا شده و ادامه برنامه به بعد از end منطق می شود.

```
switch (value)
case {1,3,5,7,9}
   disp('The value is odd.');
case {2,4,6,8,10}
   disp('The value is even.');
otherwise
   disp('The value is out of range.');
end
```

مثال) تبدیل واحدهای انرژی

Sample Problem 6-4: Converting units of energy

Write a program in a script file that converts a quantity of energy (work) given in units of either joule, ft-lb, cal, or eV to the equivalent quantity in different units specified by the user. The program asks the user to enter the quantity of energy, its current units, and the desired new units. The output is the quantity of

energy in the new units.

The conversion factors are: $1\,\text{J} = 0.738\,\text{ft-lb} = 0.239\,\text{cal} = 6.24 \times 10^{18}\,\text{eV}$.

Use the program to:

(*a*) Convert 150 J to ft-lb.

(*b*) Convert 2,800 cal to J.

(*c*) Convert 2.7 eV to cal.

```
Ein=input('Enter the value of the energy (work) to be converted: ');
EinUnits=input('Enter the current units (J, ft-lb, cal, or eV):  ','s');
EoutUnits=input('Enter the new units (J, ft-lb, cal, or eV):  ','s');
error=0;
switch EinUnits
case 'J'
    EJ=Ein;
case 'ft-lb'
    EJ=Ein/0.738;
case 'cal'
    EJ=Ein/0.239;
case 'eV'
    EJ=Ein/6.24e18;
otherwise
    error=1;
end
switch EoutUnits
case 'J'
    Eout=EJ;
case 'ft-lb'
    Eout=EJ*0.738;
case 'cal'
    Eout=EJ*0.239;
case 'eV'
    Eout=EJ*6.24e18;
```

Assign 0 to variable `error`.

First `switch` statement. Switch expression is a string with initial units.

Each of the four `case` statements has a value (string) that corresponds to one of the initial units, and a command that converts `Ein` to units of J. (Assign the value to `EJ`.)

Assign 1 to `error` if no match is found. Possible only if initial units were typed incorrectly.

Second `switch` statement. Switch expression is a string with new units.

Each of the four `case` statements has a value (string) that corresponds to one of the new units, and a command that converts `EJ` to the new units. (Assign the value to `Eout`.)

```
otherwise
    error=1;
end
if error
    disp('ERROR current or new units are typed incorrectly.')
else
    fprintf('E = %g %s',Eout,EoutUnits)

end
```

Assign 1 to `error` if no match is found. Possible only if new units were typed incorrectly.

`If-else-end` statement.

If `error` is true (nonzero), display an error message.

If `error` is false (zero), display converted energy.

بردارهای منطقی

- از بردارهای منطقی که حاوی صفر یا یک هستند، می توان برای آدرس دهی استفاده نمود.

```
>> r = [8 12 9 4 23 19 10]
```
Define a vector r.

```
r =
     8    12     9     4    23    19    10
>> s=r<=10
```
Checks which r elements are smaller than or equal to 10.

```
s =
     1     0     1     1     0     0     1
```
A logical vector s with 1s at positions where elements of r are smaller than or equal to 10.

```
>> t=r(s)
```
Use s for addresses in vector r to create vector t.

```
t =
     8     9     4    10
```
Vector t consists of elements of r in positions where s has 1s.

```
>> w=r(r<=10)
```
The same procedure can be done in one step.

```
w =
     8     9     4    10
```

نکاتی بیشتر پیرامون عیب یابی

- در برنامه های حاوی عبارت شاخه ای یا حلقه امکان اشتباه بیشتر است.
- استفاده از breakpoint
- استفاده از F10 برای اجرای خط به خط