Matlab Programming

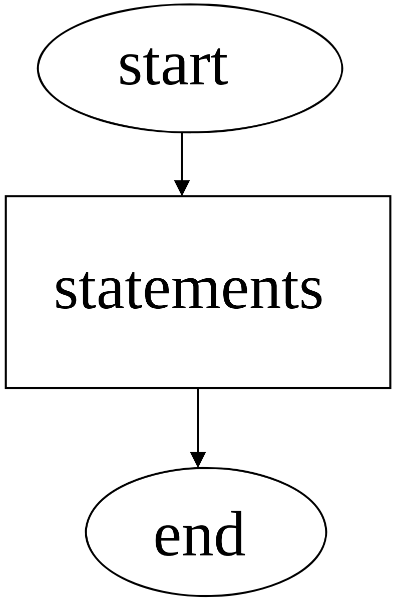
Recursive programming for determinant calculation

- MatlabProgramming: Problem Set B.pdf
- MatlabProgramming: Problem Set D.pdf

MATLAB Programming

- Flow control
  - if, for, while
- Function call
- Recursive programming

sequential execution



One-by-one stepwise sequential execution

Statements:

Assignment: A = B*C-D

x = sort(x)

I/O function: plot(x,y)

: imread(X)

Control statements

-- if statement

-- if else statement

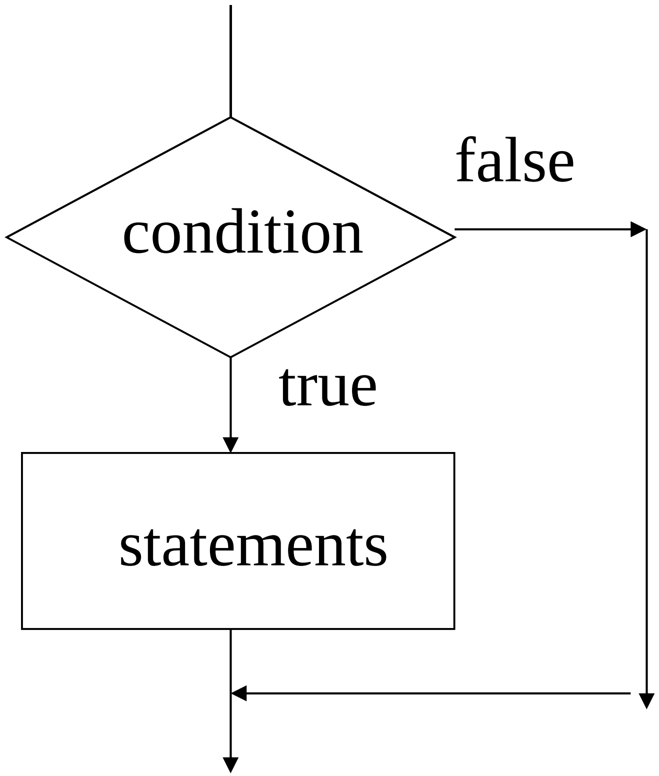
-- for statement

-- while statement

If

If statement

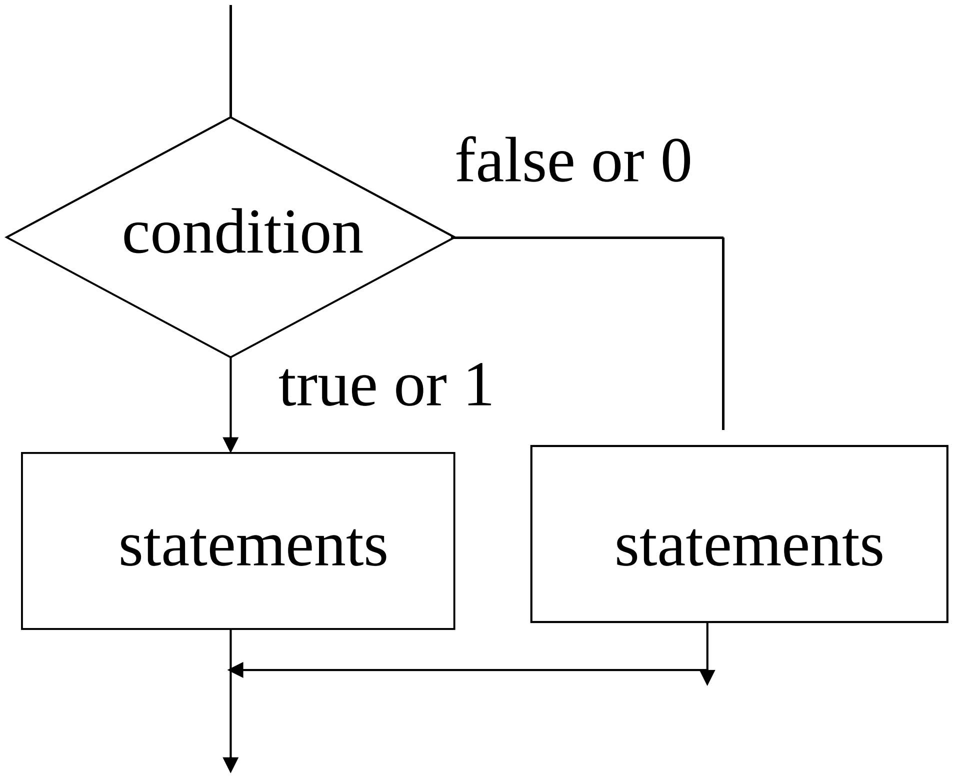
```
tag = 0;
if ~ tag
   tag = tag+1;
end
```



If else

If else statement

```
if ~ tag
   tag = tag +1;
else
   tag = tag –1;
end
```

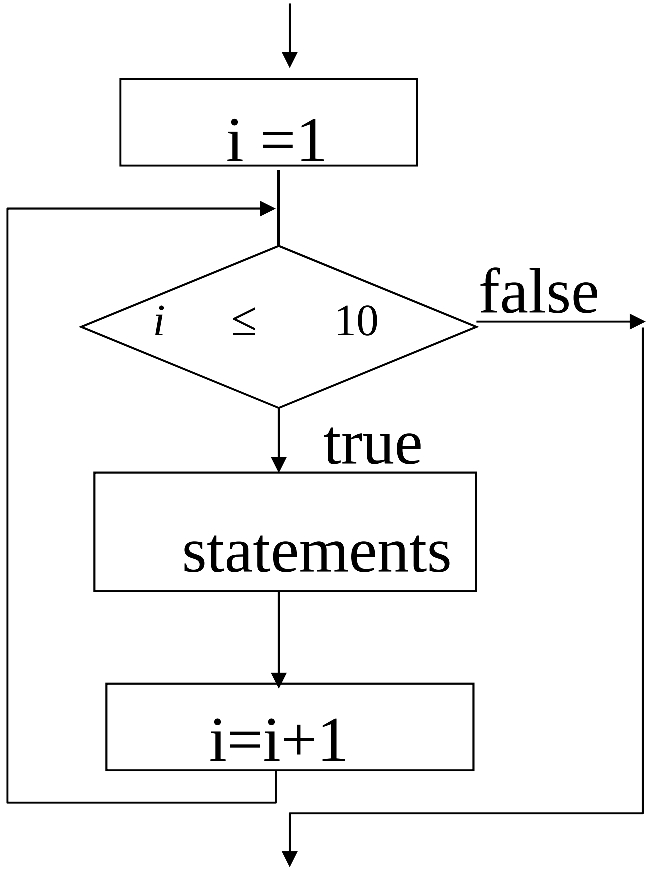


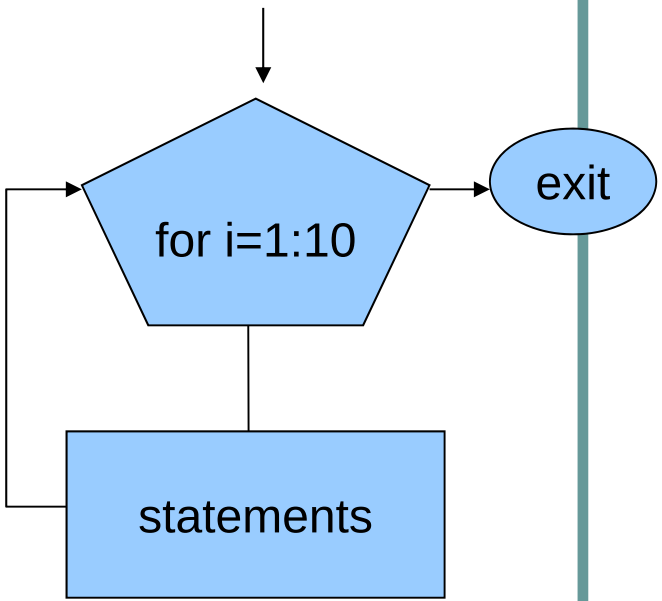
For loop

for statement

Example

```
x=2;
for  i  = 1:10
   x = x*x;
end
```



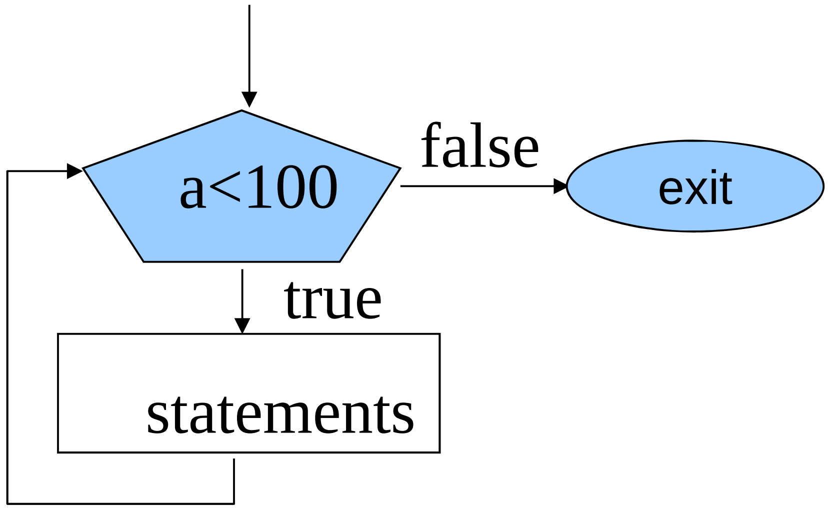


While loop

while statement

Example

```
a=1;
b=2;
while  a < 100
   a = a*b;
end
```



Break and return

break is used to escape from an enclosing `while` or `for` loop. Execution continues at the end of the enclosing loop construct.

return is used to force an exit from a **function**.

Example

```
x = rand(1,n);
k = 1;
while k<=n
   if x(k)>0.8
     break
   end
   k = k + 1;
end
fprintf('x(k)=%f   for k = %d   n = %d\n',x(k),k,n);
```

Example

```
x = rand(1,n);
k = 1;

while k<=n
   if x(k)>0.8
     return
   end
   k = k + 1;
end
```

Comparison of return and break

break is used to escape the current `while` or `for` loop.

return is used to escape the current function.

```
function k = demoBreak(n)

...

while k<=n
   if x(k)>0.8
     break;
   end
   k = k + 1;
end
```

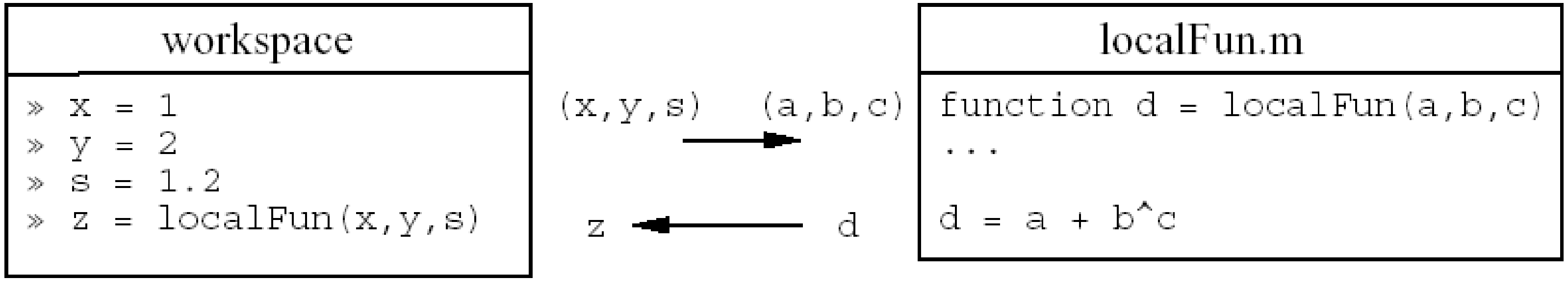
jump to end of enclosing "`while ... end`" block

```
function k = demoReturn(n)

...

while k<=n
   if x(k)>0.8
     return;
   end
   k = k + 1;
end
```

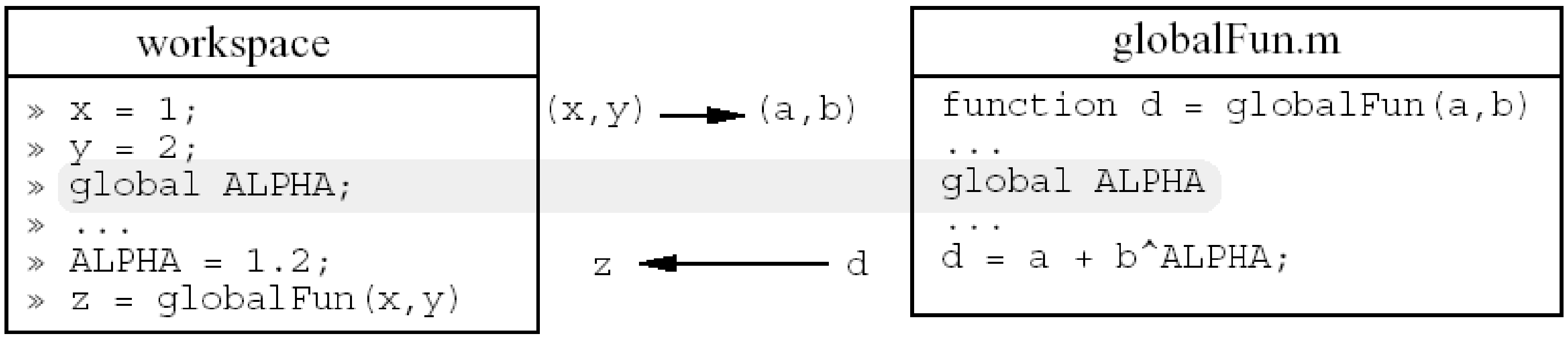
return to calling function

Function call

Communication of values via input and output variables



Communication of values via input and output variables
and global variables shared by the workspace and function



Append

```
a=[1 2 3]'
A=[]
A=[A a];
```

Determinant

```
>> A=[1 2;3 4]

A =

     1     2
     3     4

>> det(A)

ans =

    -2
```

Problem statement

- A denotes a square matrix
- Find the determinant of matrix A by recursive programming

Strategy

- A is an NxN matrix
- N demotes the problem size
- problem reduction
  - Decompose the determinant of matrix A to subtasks of determining determinants of N (N-1)x(N-1) sub-matrices if N > 2
  - Directly calculate the determinant of matrix A if N <= 2

Rule I

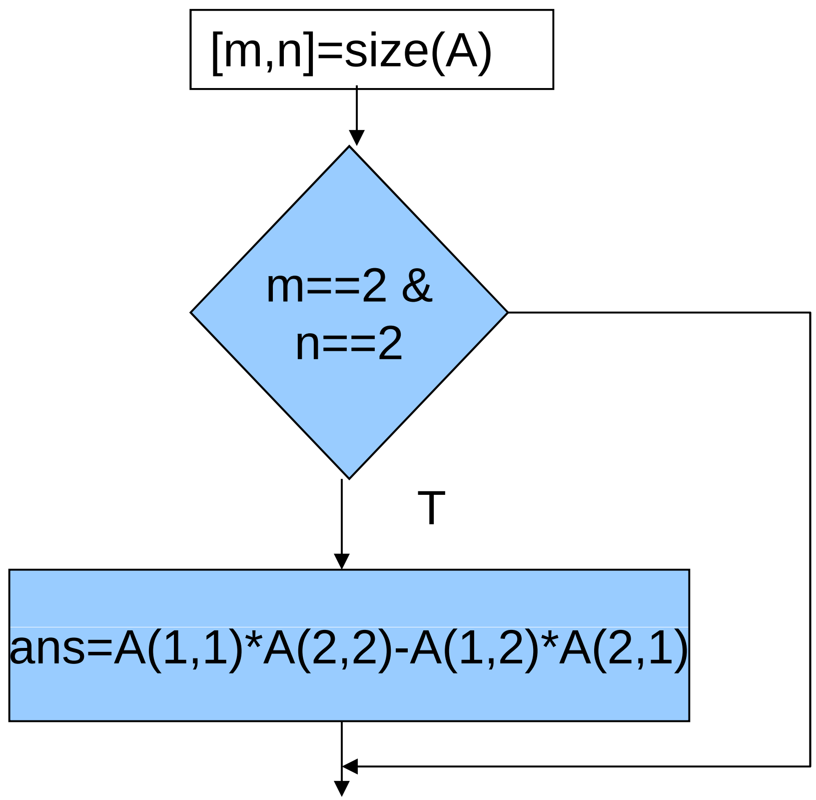
- A is 2x2

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$$

$$Det(A) = a_{11}a_{22} - a_{12}a_{21}$$

Flow chart

- Let A be a 2-by-2 matrix



$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

$$\tilde{A}_{11} = \begin{pmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{pmatrix}, \tilde{A}_{12} = \begin{pmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{pmatrix}, \tilde{A}_{13} = \begin{pmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{pmatrix}$$

$\tilde{A}_{ij}$ denote a submatrix of A, which is obtained by removing the ith row and the jth column of matrix A

n=3

- A is 3x3
- Ex n=3

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

$$Det(A) = a_{11} \det(\tilde{A}_{11}) - a_{12} \det(\tilde{A}_{12}) + a_{13} \det(\tilde{A}_{12})$$

- Calculating the determinant of a 3-by-3 matrix is decomposed to subtasks of calculating determinants of 2-by-2 matrices

Rule II

- A is nxn and n > 2

$$Det(A) = \sum_{i=1}^{n} (-1)^{i+1} a_{1i} \det(\tilde{A}_{1i})$$

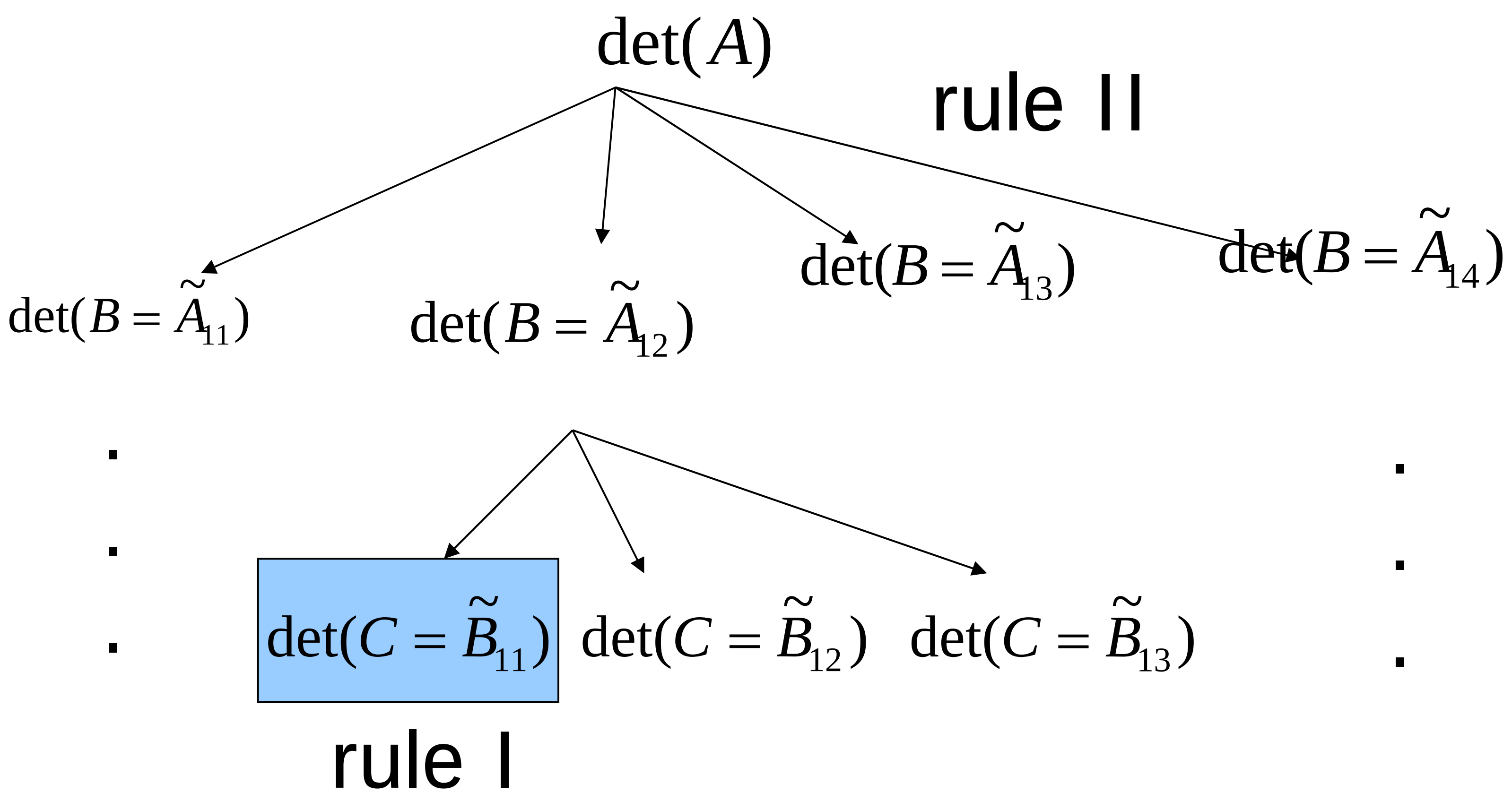
- $\tilde{A}_{1i}$ denotes a (n-1)x(n-1) sub-matrix of A. It is obtained by removing elements in the first row and the jth column of matrix A

Recurrent relation for reduction of problem size

- det(A) is decomposed to n sub-tasks
- Each calculates determinant of an (n-1)-by-(n-1) matrix $\tilde{A}_{1i}$
- The problem size is reduced from n to n-1

$$Det(A) = \sum_{i=1}^{n} (-1)^{i+1} a_{1i} \det(\tilde{A}_{1i})$$

Tree of recursive calls



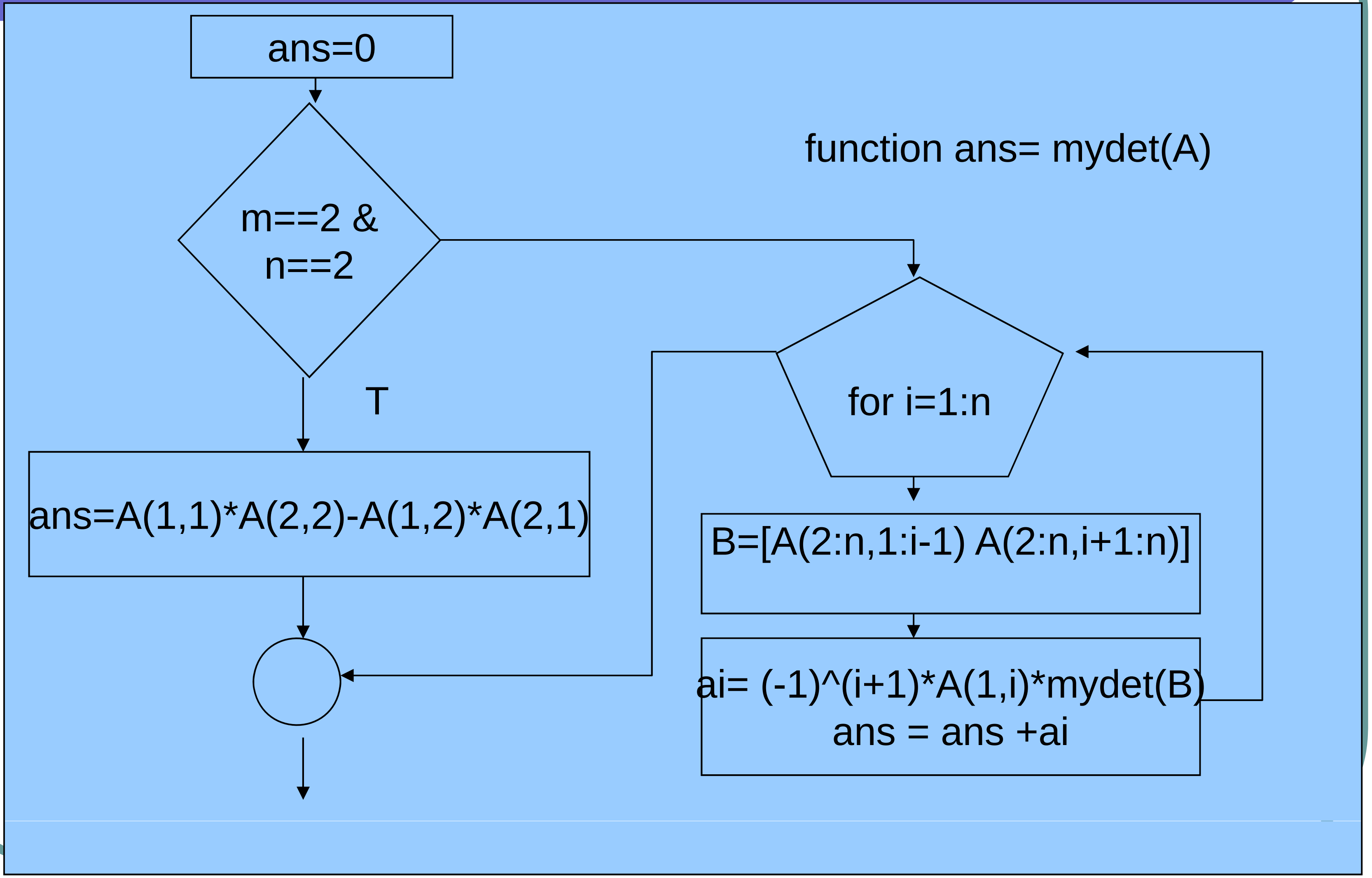
Append

```
a=[1 2 3]'
A=[]
A=[A a a];
```

matrix $\tilde{A}_{1i}$

B1=A(2:n,1:i-1)
B2=A(2:n,i+1:n)
B=[B1 B2]

Flow chart



ans=0
m==2 &
n==2
function ans= mydet(A)
T
for i=1:n
ans=A(1,1)*A(2,2)-A(1,2)*A(2,1)
B=[A(2:n,1:i-1) A(2:n,i+1:n)]
ai= (-1)^(i+1)*A(1,i)*mydet(B)
ans = ans +ai

MATLAB codes

```
function ans=mydet(A)
    ans=0;[m,n]=size(A);
    if m==1
        ans=A; return;
    end
    if m==2
        ans=A(1,1)*A(2,2)-A(1,2)*A(2,1); return;
    else
        for i=1:n
         …
        end
    end
```

```
function v=mydet(A)
% Calculate determinant of A
n=size(A,1);v=0;
if n==2
    v=A(1,1)*A(2,2)-A(1,2)*A(2,1);
    return
end
v=0;
for i=1:n
    B=get_submatrix(A,i);
    det_B=mydet(B);
    s=(-1)^(i+1);
    v=v+s*A(1,i)*det_B;
end
return
```

while-loop

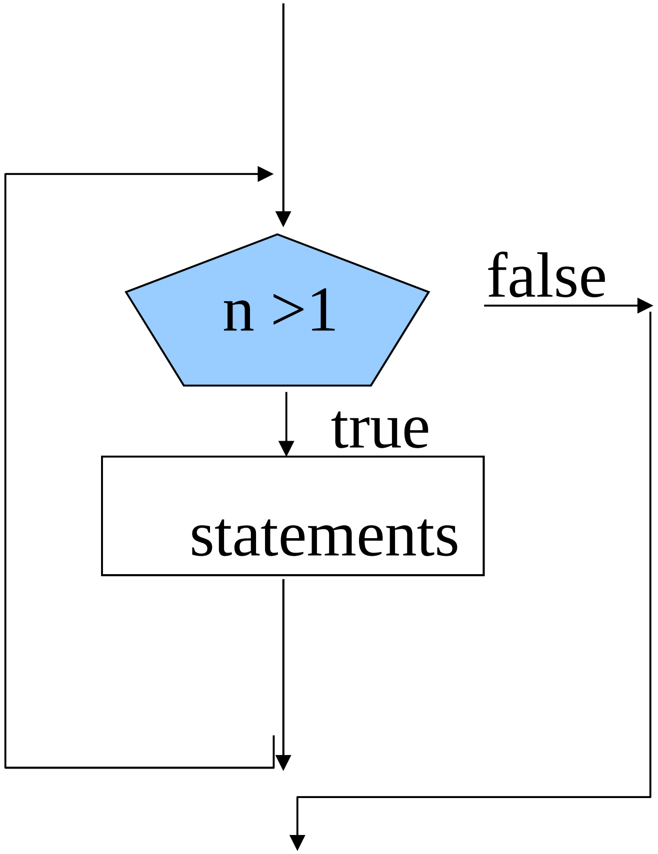
- Halting condition: a logical expression
- Loop body

```
while n>1
      s=[mod(n,2) s];
      n=(n-mod(n,2))/2;
end
 if n==1
      s=[n s];
 end
```

while-loop

```
while n>1
    s=[mod(n,2) s];
    n=(n-mod(n,2))/2;
end
```

Body statements are executed repeatedly until the halting condition holds



dec2bin

```
function s=dec2bin(n)
% n: a positive integer
% s: a vector for binary representation of n
    s=[];
    while n>1
        s=[mod(n,2) s];
        n=(n-mod(n,2))/2;
    end
    if n==1
        s=[n s];
    end
return
```