Lecture 2 Iteration

- Looping
  - for loop
  - while loop
- Break and return
- Recursion

for-loop

Source codes

$$\sum_{i=1}^{n}(4i^3+3i^2+2i+1)$$

Evaluate

```
n=input('input an integer:');
v=0;
for i=1:n
    v=v+4*i^3+3*i^2+2*i+1;
end
fprintf('The answer is %d\n',v);
```

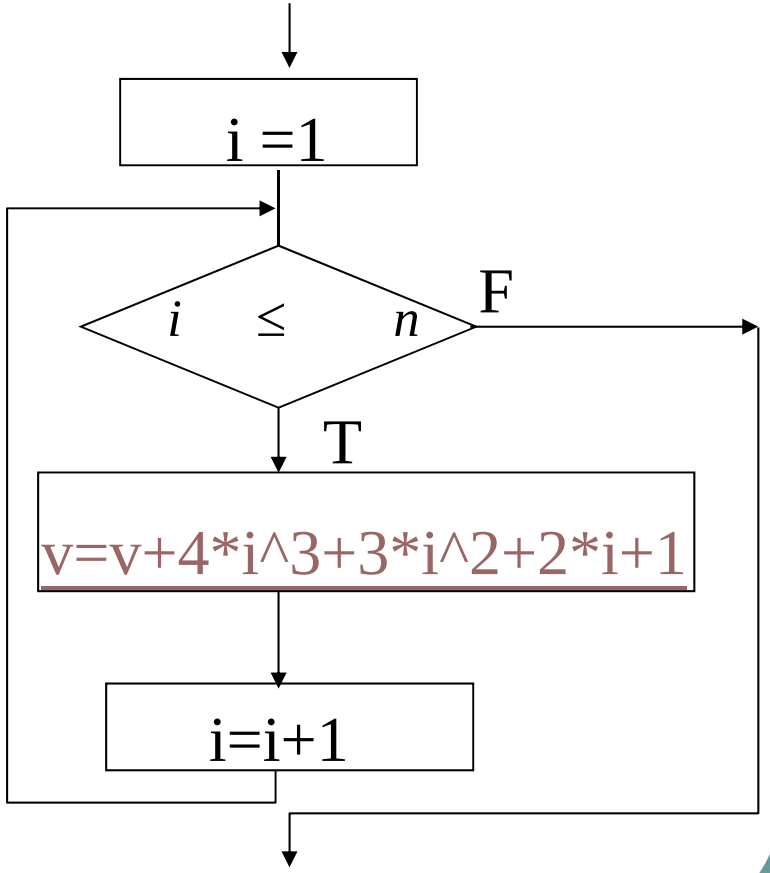
For-Looping

- Iteratively executing same body instructions
- Increasing or decreasing control index

```
for i=1:n
   v=v+4*i^3+3*i^2+2*i+1;
end
```

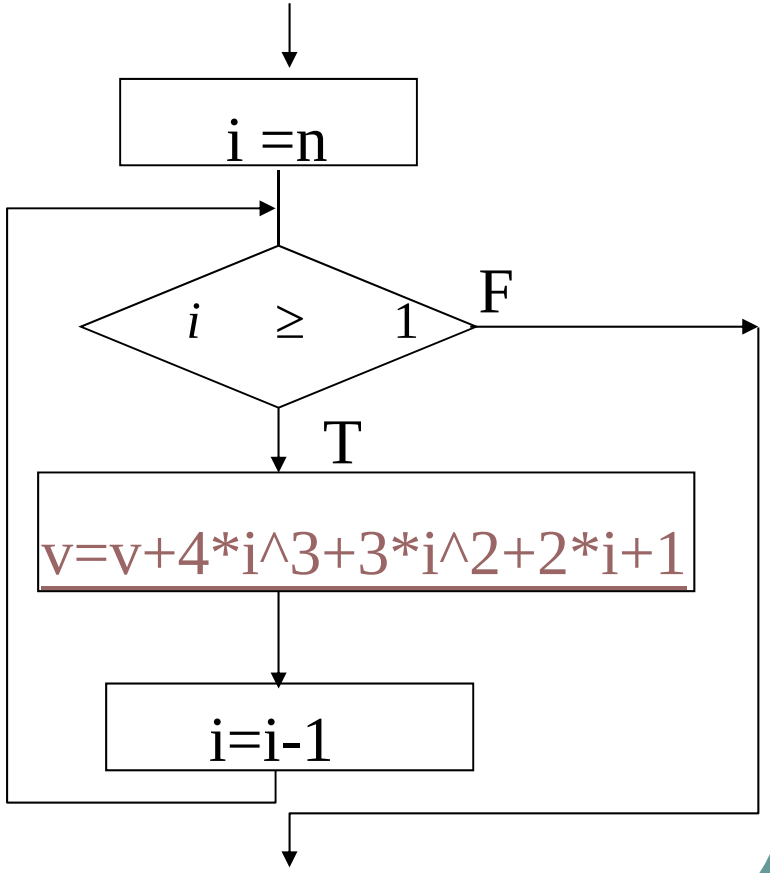
Increasing index

```
for i=1:n
    v=v+4*i^3+3*i^2+2*i+1;
end
```



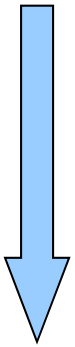
Decreasing index

```
for i=n:-1:1
    v=v+4*i^3+3*i^2+2*i+1;
end
```



Representation of polynomials

$$f(x) = a_m x^m + a_{m-1} x^{m-1} + \cdots + a_1 x + a_0$$



representation

$$\mathbf{a} = [a_m, a_{m-1}, \cdots, a_1, a_0]$$

roots

- $f(x) = x - 2$

```
roots([1 -2])
ans =

     2
```

Addition of two polynomials

$$f(x) = a_m x^m + a_{m-1} x^{m-1} + \cdots + a_1 x + a_0$$

$$+ \quad g(x) = b_n x^n + b_{n-1} x^{n-1} + \cdots + b_1 x + b_0$$

$$h(x) = c_k x^k + c_{k-1} x^{k-1} + \cdots + c_1 x + c_0$$

$$c_i = a_i + b_i, i = 0,...,k$$

$$c_i = 0, if\ i > k$$

add_poly

Source codes

```
m=length(a);n=length(b);
c=zeros(1,max(m,n));
size_c=length(c);
for i=1:m
    c(size_c-i+1)=c(size_c-i+1)+a(m-i+1);
end
for i=1:n
    c(size_c-i+1)=c(size_c-i+1)+b(n-i+1);
end
ii=min(find(c~=0));
c=c(ii:size_c);
```

add_poly

```
>> a=[-1 2 3]; b=[1 2];
add_poly(a,b)
ans =
    -1     3     5
```

```
>> a=[-1 2 3]; b=[1 -2 3];
>> add_poly(a,b)
ans =
     6
```

Nested loop

- Loops within loops

Ex.

source codes : multiplication of two matrices

Nested loop

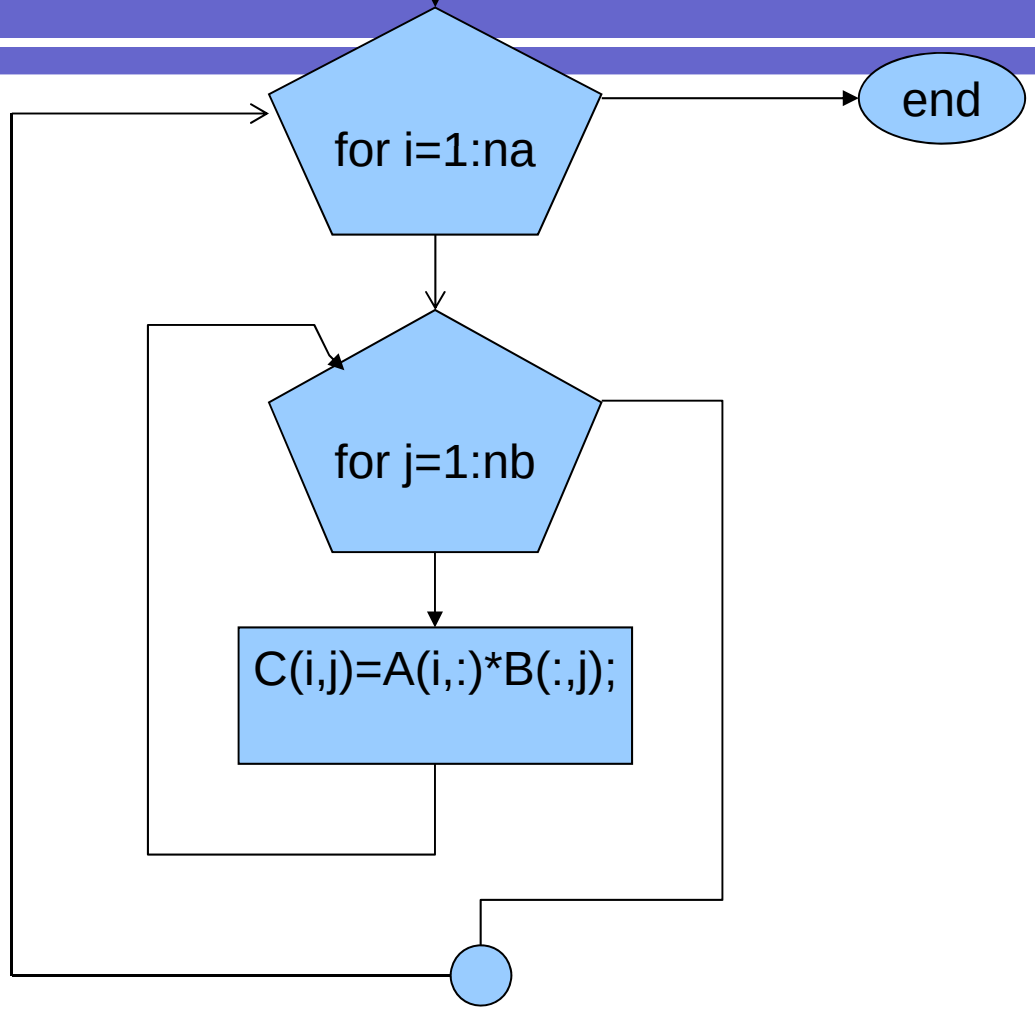
outer

inner

```
[na ma]=size(A);
[nb mb]=size(B);
for i=1:na
    for j=1:mb
        C(i,j)=A(i,:)*B(:,j);
    end
end
```

- The inner loop body is executed na x mb times
- Nested loops are usually time consuming

Nested loop



Speeding-up

- Eliminate nested loops
- Rewrite loops or nested loops to equivalent loop-free codes

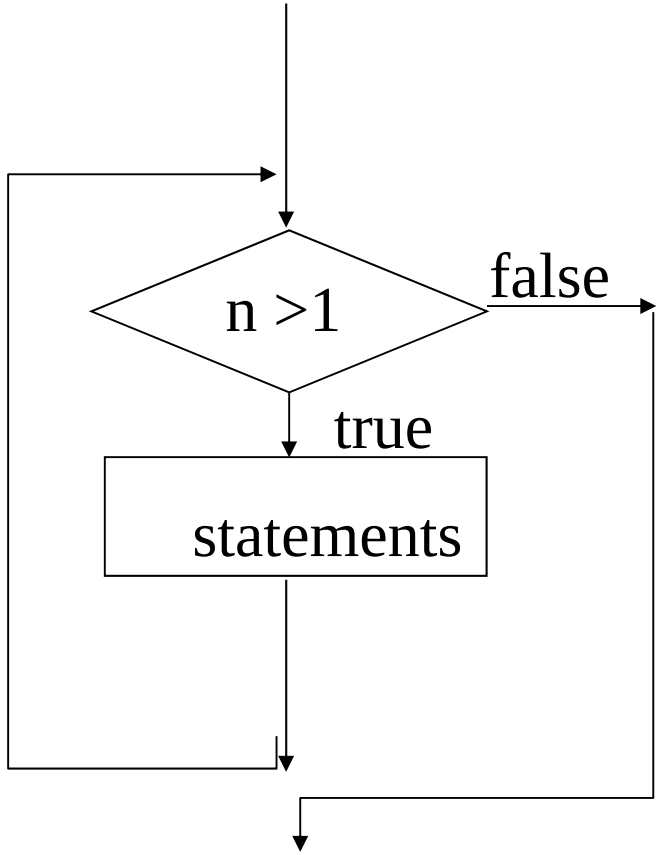
Ex.

C=A*B

while-loop

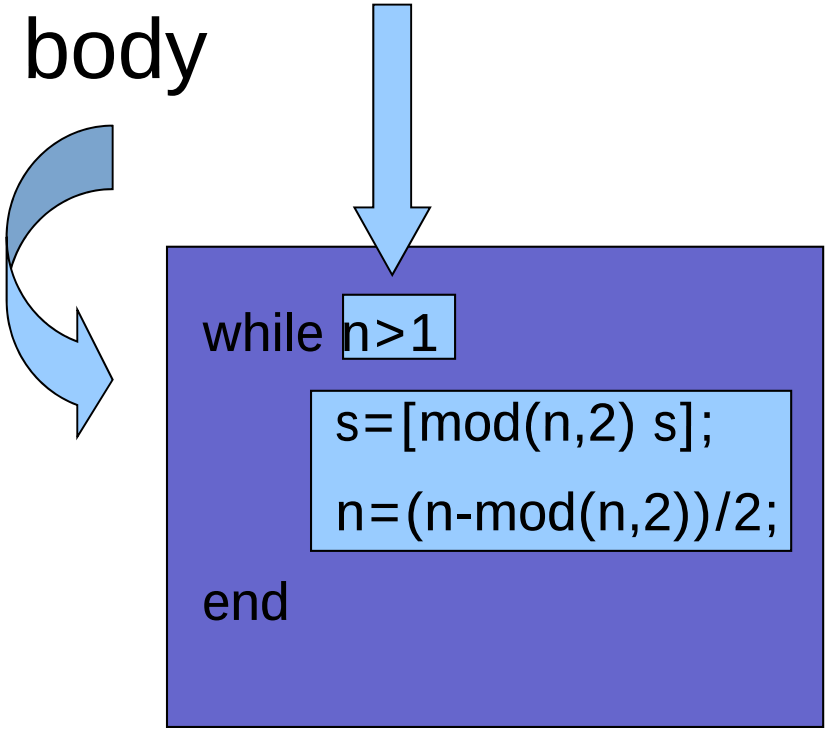
```
while n>1
        s=[mod(n,2) s];
        n=(n-mod(n,2))/2;
end
```

Body statements are executed repeatedly until the halting condition holds



while-loop

- Halting condition: a logical expression
- Loop body



dec2bin

Source codes

```
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
```

Decimal to binary

- Find binary representation of a decimal number
- Input: a decimal number
- Output: binary representation

dec2bin

```
>> dec2bin(12)

ans =

     1     1     0     0
```

```
>> ceil(log2(12))
ans =
     4
```

dec2bin

```
>> dec2bin(88)
ans =
     1     0     1     1     0     0     0
```

```
>> ceil(log2(88))
ans =
     7
```

Recursive function

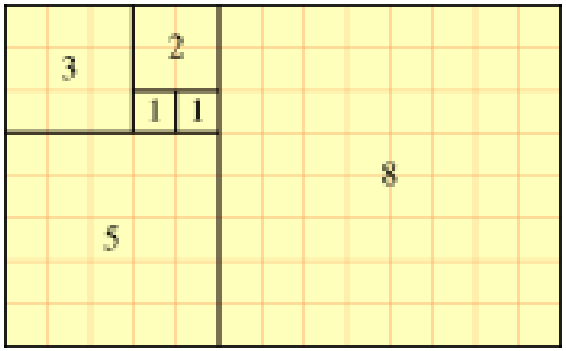
$$a_n = a_{n-1} + 2a_{n-2} \quad \text{if } n > 1$$

$$a_1 = a_0 = 1$$

Fibonacci number

$$F_n = \begin{cases} 0 & \text{if } n = 0; \\ 1 & \text{if } n = 1; \\ F_{n-1} + F_{n-2} & \text{if } n > 1. \end{cases}$$

Fibonacci number



A tiling with squares whose sides are successive Fibonacci numbers in length

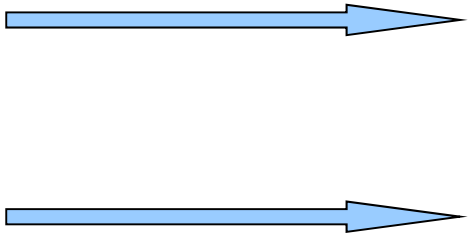
fib

source codes

```
function s=fib(n)
%
% n: a nonnegative integer
% fib(1) and fib(0) returns 1 and 0
% fib(n) equals fib(n-1)+fib(n-2) for n>1
    if n<=1
        s=n;
        return
    end
    s=fib(n-1)+fib(n-2);
return
```

return

- Execution of a matlab function always exits whenever ‘return’ is executed
- There are two exit points in fib function



```
if n<=1
    s=n;
    return
end
s=fib(n-1)+fib(n-2);
return
```

Recursive function call

- Recursive function call
  - A function calls itself during its execution
- Call fib(n) with n > 1
  - fib calls itself at line 10

Calling Tree of fib(4)

- Fib(4)
  - Fib(3)
    - Fib(2)
      - Fib(1)
      - Fib(0)
    - Fib(1)
  - Fib(2)
    - Fib(1)
    - Fib(0)

Exercise

- exercise 2