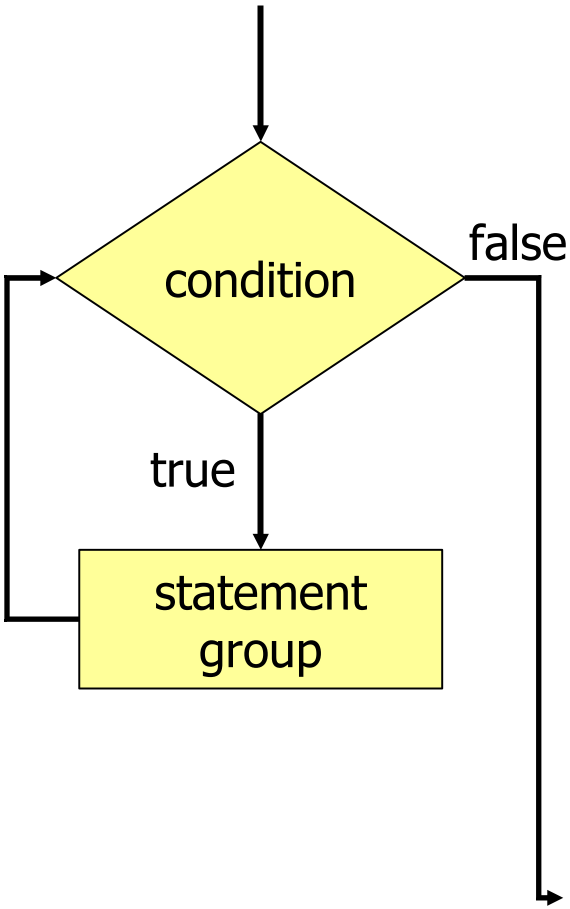
Loops

- Loops permit us to execute a sequence of statements more than once
- We will learn:
  - while loop
  - for loop
- They differ in how the repetition is controlled

“while” Loop



- Statements are executed indefinitely as long as the condition is satisfied

```
while ( condition ),
    statement 1
    statement 2
    …
end
```

statement group

Example: “Average computation”

- Compute the average of *n* values entered by the user. The number of values (*n*) will be specified by the user.

  -
    ```
    n = input('Enter the number of values: ');
    counter = 0;
    total = 0;
    while (counter < n )
        x = input('Enter a value: ');
        total = total + x;
        counter = counter + 1;
    end
    if (n > 0)
        avg = total / counter;
        disp(['The average is ' num2str(avg)]);
    end
    ```

Example: “Average computation”

- Compute the average of values entered by the user. A negative value indicates the end of the input.

  -
    ```
    counter = 0;
    total = 0;
    x = input('Enter the first value: ');
    while (x >= 0 )
        counter = counter + 1;
        total = total + x;
        x = input('Enter the next value: ');
    end
    if (counter > 0)
        avg = total / counter;
        disp(['The average is ' num2str(avg)]);
    end
    ```

Example: “Change case”

- Change the case of letters in a sentence.

  -
    ```
    s = input('Enter a sentence: ','s');
    k = 1;
    while (k <= length(s))
        if (s(k) >= 'A') & (s(k) <= 'Z')
            new_s(k) = char('a' + (s(k) - 'A'));
        elseif (s(k) >= 'a') & (s(k) <= 'z')
            new_s(k) = char('A' + (s(k) - 'a'));
        else
            new_s(k) = s(k);
        end
        k = k + 1;
    end
    new_s
    ```

“for” Loop

- Statements are executed a specified number of times
  - No of repetitions is known before the loop starts

```
for index = expression,
    statement 1
    statement 2
    ...
end
```

statement group

- Expression is usually a vector in shortcut notation first:increment:last
  - for x = 1:3:12
    ```
    for x = 1:3:12
        x
    end
    ```
  - for x = [3 8 19]
    ```
    for x = [3 8 19]
        x
    end
    ```

Example: “Average computation”

- **Using “for” loop,** compute the average of *n* values entered by the user. The number of values (*n*) will be specified by the user.

  -
    ```
    n = input('Enter the number of values: ');
    total = 0;
    for counter = 1 : n
        x = input('Enter a value: ');
        total = total + x;
    end
    if (counter > 0)
        avg = total / counter;
        disp(['The average is ' num2str(avg)]);
    end
    ```

Example: “Factorial calculation”

- Calculate the factorial (*N!*) of an integer *N*; the factorial of negative integers is not defined.

  -
    ```
    N = input('Enter a non-negative integer: ');
    if N < 0
            disp(['It is a negative integer']);
    else
            result = 1;
            for i = 1 : N,
                    result = result *i;
            end
            disp([num2str(N) '! = ' num2str(result)]);
    end
    ```

Example: “Perfect numbers”

- Write a program that finds the first three perfect numbers.
    - A positive integer *n* is a perfect number if the sum of its positive divisors excluding *n* itself is equal to *n*.
        - e.g., 28 is a perfect number; 1+2+4+7+14=28
- Loops can be nested too.

Example: “Perfect numbers”

```
counter = 1;
no = 1;
while counter <= 3
    total = 0;
    for ii = 1 : no/2
        if mod(no,ii) == 0
            total = total + ii;
        end
    end
    if no == total
        perfect_nos(counter) = no;
        counter = counter + 1;
    end
    no = no + 1;
end
fprintf('The first three perfect numbers are ')
for counter = 1 : 3
    fprintf('%d ',perfect_nos(counter));
end
fprintf('\n');
```

Example: “Matrix multiplication”

- Compute the multiplication of two matrices.
  - If *A* is an m-by-n matrix and *B* is an n-by-p matrix, their product is an m-by-p matrix *C* which is given by

$$C_{ij} = \sum_{k=1}^{n} A_{ik} \cdot B_{kj}$$

  - The matrix multiplication is defined between two matrices only if the number of columns of the 1st matrix is the same as the number of rows of the 2nd matrix.

Example: “Matrix multiplication”

```
[row_A,column_A] = size(A);
[row_B,column_B] = size(B);

if column_A ~= row_B
    disp('Matrix dimensions must agree');
else
    for ii = 1 : row_A
        for jj = 1 : column_B
            C(ii,jj) = 0;
            for k = 1 : column_A
                C(ii,jj) = C(ii,jj) + A(ii,k) * B(k,jj);
            end
        end
    end
end
```

Important details

- Use indentation to improve the readability of your code
- Never modify the value of a loop index inside the loop
- To have faster programs in Matlab:
  - Allocate all arrays used in a loop before executing the loop
  - If it is possible to implement a calculation either with a loop or using vectors, always use vectors
  - Use built-in MATLAB functions as much as possible instead of reimplementing them

Comparison of the execution times

-
  ```
  A = rand(100,200); B = rand(200,50);
  ```

-
  ```
  tic
  for ii = 1 : 1000
      C1 = A * B;
  end
  t1 = toc / 1000;
  ```

-
  ```
  tic
  for ii = 1 : 100
      for jj = 1 : 50
          C2(ii,jj) = 0;
          for k = 1 : 200
              C2(ii,jj) = C2(ii,jj) + A(ii,k) * B(k,jj);
          end
      end
  end
  t2 = toc;
  ```

-
  ```
  tic
  for ii = 1 : 100
      for jj = 1 : 50
          C3(ii,jj) = A(ii,:) * B(:,jj);
      end
  end
  t3 = toc;
  ```

-
  ```
  tic
  C4 = zeros(100,50);
  for ii = 1 : 100
      for jj = 1 : 50
          for k = 1 : 200
              C4(ii,jj) = C4(ii,jj) + A(ii,k) * B(k,jj);
          end
      end
  end
  t4 = toc;
  ```

- t1 = 0.0020, t2 = 5.0160, t3 = 0.1100, t4 = 4.9060

“Break/continue” statements

- Break statement terminates the execution of a loop and passes the control to the next statement after the end of the loop
- Continue statement terminates the current pass through the loop and returns control to the top of the loop

“break” statements

- Example:

```
for ii = 1:5,
    if ( ii == 3 ),
        break;
    end
    fprintf( 'ii = %d\n', ii );
end
disp( 'End of loop' );
```

- Output:

```
ii = 1
ii = 2
End of loop
```

“continue” statement

- Example:

```
for ii = 1:5,
    if ( ii == 3 ),
        continue;
    end
    fprintf( 'ii = %d\n', ii );
end
disp( 'End of loop' );
```

- Output:

```
ii = 1
ii = 2
ii = 4
ii = 5
End of loop
```

Example: “Number guessing”

- Write a program in which the user tries to guess a number picked by the computer. The number is picked between 1 and 10 and the user has at most three tries.

  -
    ```
    num = fix(10 * rand + 1);
    if num == 11, num = 10; end

    for tries = 1:3,
        guess = input( 'Your guess? ' );
        if ( guess == num ),
            disp( 'Congratulations!' );
            break;
        end
    end
    if ( guess ~= num ),
        disp( 'You could not guess correctly' );
    end
    ```

Example: “Perimeter of a polygon”

- Compute the perimeter of a polygon whose size is specified by the user.

```
N = input('Enter the polygon size: ');
if N < 3
    disp('It is not a polygon');
else
    ii = 1; perimeter = 0;
    while (ii <= N)
        edge_length = input([‘Length of edge ' num2str(ii) ': ']);
        if edge_length <= 0
            disp('The length should be positive');
            continue;
        end
        perimeter = perimeter + edge_length;
        ii = ii + 1;
    end
end
disp(['Perimeter: ' num2str(perimeter)]);
```