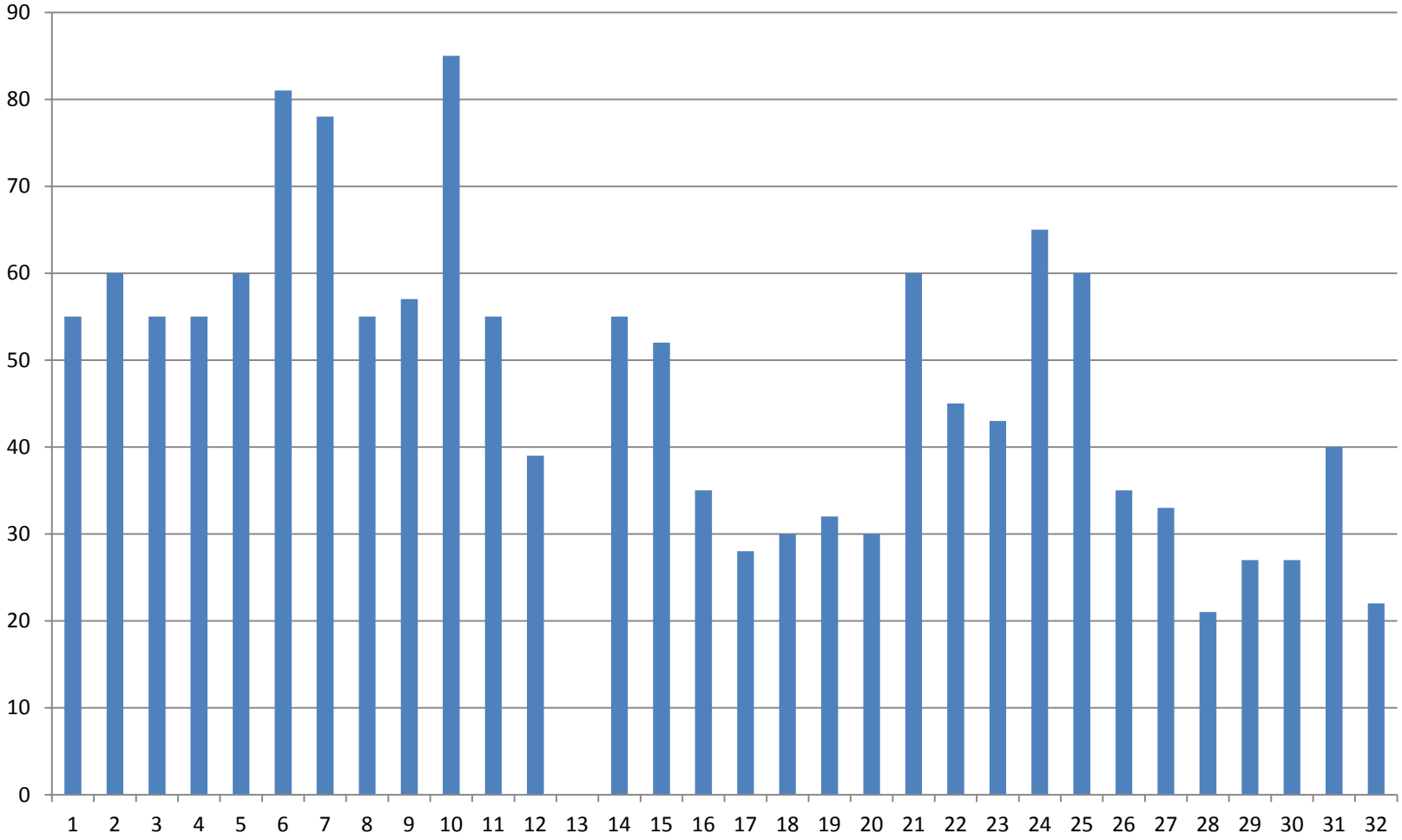
Algoritma & Pemrograman Saintifik

Gatot F. Hertono, Ph.D

Departemen Matematika

SCMA601401

Breaking News



Sebaran Nilai UTS

Statements

In general, in the programming, statements can be distinguished as:

- **Assignment**:
  To "assign" a variable means to symbolically associate a specific piece of information with a name. Any operations that are applied to this "name" (or variable) must hold true for any possible values.

- **Branching:**
  When an "Algorithm" makes a choice to do one of two (or more things) this is called branching. The most common programming "statement" used to branch is the "IF" statement.

- **Looping:**
  A Loop is used to repeat a specific block of code a over and over. There are two main types of loops, for loops and while loops.

Source: http://www.cs.utah.edu/~germain/PPS/Topics/

Variable Assignment

The '=' symbol is the assignment operator which **SHOULD NEVER** be used for equality (which is the double equals sign).

Warning, while the assignment operator looks like the traditional mathematical equals sign, this is NOT the case. The equals operator is '=='

Syntax: `variable_name = expression;`

- ✓ Expression could be a number: variable_name = 5;
- ✓ or a math expression : variable_name = 10 + 5 / 3 - 7;
- ✓ or a funciton call : variable_name = sin(5);

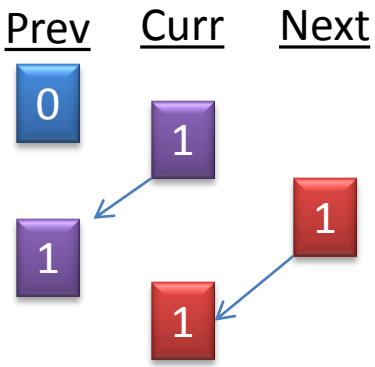
To evaluate an assignment statement:

1. Evaluate the "right side" of the expression (to the right of the equal sign).
2. Once everything is figured out, place the computed value into the variables bucket.

Example

```
B1 = 10.5;
A1 = input('What is the current odometer reading?');
A2 = input('How many gallons of gas did you pump?');
Miles = A1 - B1;
Mileage = Miles/A2;
A = 'The mileage is';
Tally = Tally + 1;
disp(A);
disp(Mileage);
```

Prev = 0;
Curr = 1;
Next = Prev + Curr;
Prev = Curr;
Curr = Next



Branching

In a computer program, the algorithm often must choose to do one of two things depending on the "state" of the program.

As an example:

If the grade is greater than 90, then
give the student an A,
otherwise
if the grade is greater than 80,
give the student a B,... etc.

Syntax:

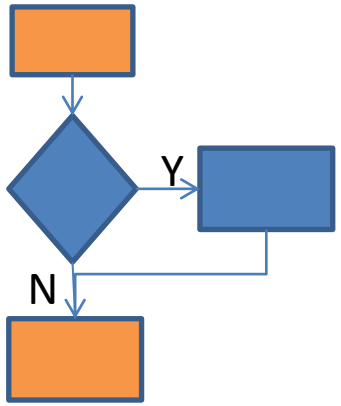
```
if ( something is true )
    do this code;
    do all code before the end or else;
    do not do anything in the else "block" of code
 else
    % if the something is false (NOTE: we don't have to test this)
    do other code;
 end
```

Source: http://www.cs.utah.edu/~germain/PPS/Topics/

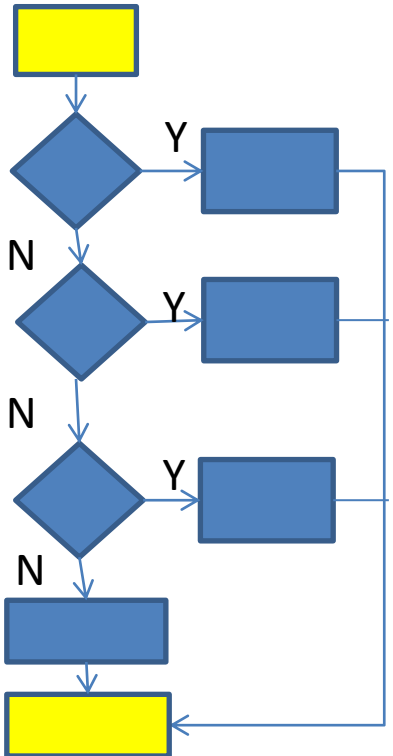
Examples

```
grade =  % some_number;

 if ( grade > 75 )
      fprintf('congrats, your grade %d is passing\n', grade);
end
```




```
if (money < 5)
        do this;
    elseif (money < 10)
        do that;
    elseif (money < 1000)
         do a lot of stuff;
    else
         do a default action when nothing above is true;
    end
```



Looping

Loops - or repeating yourself

Loops allows us to repeat a single (or several) lines of code over and over again. This allows us to "write once" and then "execute many times".

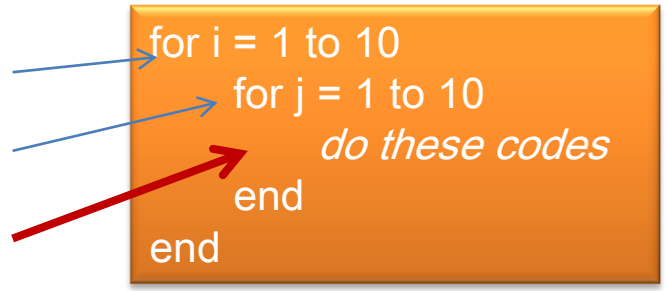
There are **TWO** loops that you must memorize.

- **The For Loop**: Which is used when we know how many times the loop will execute.
- **The While Loop:** When we don't know how many times, but want to continue until **a certain condition is not true**.

Nesting Loops

It is perfectly legal to place the code for one loop **inside** the code of another loop. What this means is that the "inner" loop is executed one time in its entirety for **every** time the outer loop executes.

As an example, suppose the **outer loop** executes 10 times and the **inner loop** executes 10 times, the code inside the inner loop is executed 100 times!



Source: http://www.cs.utah.edu/~germain/PPS/Topics/

Looping (cont.)

Syntax:

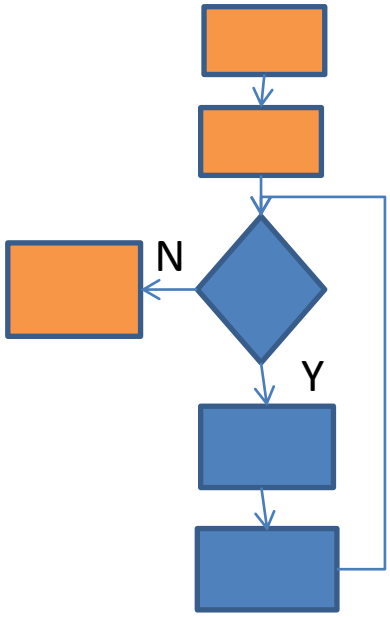
```
for  var_index = start_value : increment_value : end_value
        % Do this code
end


% implied increment by 1
 for var_index = start_value : end_value
        % Do this code
 end


while ( condition is true / boolean expression )
    % do something
    % Note: the "something" should eventually result
    % in the condition being false
  end
```

Examples

```
number = input('Input any number: ');
count = 0;
while (number > 1)
        number = number / 2;
        count  = count + 1;
end ;
```



Infinite loops

```
while ( y < 10 )
    x = x + 1;
end;
```

```
while (  true )
    disp('hello');
end;
```

```
for i=1:10
    disp(i);
    for j=1:i
        disp(j);
        for k=1:j
            disp(i*j*k)
        end
    end
end
```

Exercises

Build a program to compute:

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots \frac{x^n}{n!}, \qquad -\infty < x < \infty$$

and

Number Guess Games:
The computer will come up with a random number between 1 and 1000. The player must then continue to guess numbers until the player guesses the correct number. For every guess, the computer will either say "Too high" or "Too low", and then ask for another input. At the end of the game, the number is revealed along with the number of guesses it took to get the correct number.