Designing an Algorithm

Assoc. Prof. Tunç Durmaz

March 15, 2021

Introductory Computer Science
ENF1170

Designing an algorithm

- An **algorithm** is a plan, a logical step-by-step process for solving a problem.
  - *Algorithm:* A sequence of logical instructions for carrying out a task. In computing, algorithms are needed to design computer programs
- Algorithms are normally written as a flowchart or in pseudocode.
  - ✓ *Flowchart (akış diyagramı):* A diagram that shows a process, made up of boxes representing steps, decision, inputs and outputs.
  - ✓ *Pseudocode (sözde kod):* A method of writing up a set of instructions for a computer program using plain language. This is a good way of planning a program before coding.

✓ PLAIN TEXT: All the steps are written in plain text with sentences.

In daily life we make plans:

Ex: (After pandemic is over) Attending the class and coming back home

1. wake up
2. Get prepared
3. Have breakfast
4. Go to the university
5. Attend the class
6. Return back home

1. Wake up ✓
2. Get prepared ✓ ?
3. Do you have food for breakfast? ✓
   3.1. If no, go to the grocery ✓
4. Have breakfast ✓
5. ~~Go to the bus stop~~
6. ~~wait for the bus~~
7. ~~Is the bus on time?~~
   - ~~Take the bus~~
   - ~~No, go to the subway station~~
     - ~~Wait for the train / subway~~
     - ~~Take the train~~
8. Attend the class ✓
9. ~~Return back home~~

ALGORITHM is the equivalent of a plan in the computer sphere.

Example: Sum up two numbers that are entered by the keyboard and printed on the screen:

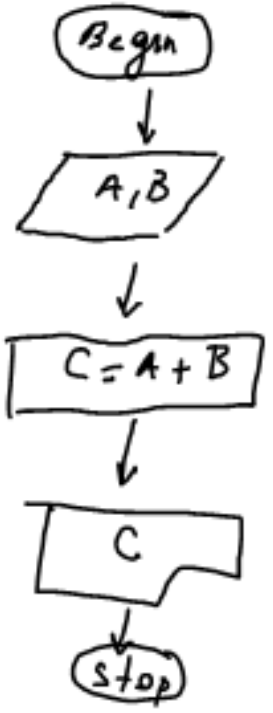
PLAIN TEXT

1. Begin
2. Read the first number entered by the keyboard
3. Read the second number entered by the keyboard
4. Sum up the two numbers
5. Print the result on the screen / display
6. Stop

PSEUDO - CODES

1. Begin
2. Read A
3. Read B
4. C = A + B
5. Print C
6. Stop

FLOW CHART



Designing an algorithm (cont.)

- The key to any problem-solving task is to guide your thought process. The most useful thing to do is keep asking 'What if we did it this way?' Exploring different ways of solving a problem can help to find the best way to solve it.
- When designing an algorithm, consider if there is more than one way of solving the problem.

Designing an algorithm (cont.)

When designing an algorithm there are two main issues to look at:

- the **big picture** - What is the final goal?
- the **individual stages** - What obstacles should be overcome on the way to the goal?

Understanding the problem

- Before an algorithm can be designed, it is important to check that the problem is completely understood.
- There are a number of basic things to know in order to really understand the problem:
  - What are the inputs into the problem?
  - What will be the outputs of the problem?
  - In what order do instructions need to be carried out?
  - What decisions need to be made in the problem?
  - Are any areas of the problem repeated?
- Once these basic things are understood, it is time to design the algorithm.

Pseudocode

- Most programs are developed using programming languages.
  - Programming language: A language used by a programmer to write a piece of software. There are many programming languages; e.g., ...
- These languages have specific syntax that must be used so that the program will run properly.
  - Syntax (sözdizimi, sözdizimi kuralları): Rules governing how to write statements in a programming language.
  - Program: Sequences of instructions for a computer.
  - Instruction: A single action that can be performed by a computer processor.
- Pseudocode is not a programming language, it is a simple way of describing a set of instructions that does not have to use specific syntax.

Common pseudocode notation

- There is no strict set of standard notations for pseudocode
  - Notation: A system of written symbols or graphics used to represent something in order to aid communication and understanding.

Common pseudocode notation

- Some of the most widely recognised notations for pseudocode are:
  - INPUT: indicates a user will be inputting something
  - OUTPUT: indicates that an output will appear on the screen
  - WHILE: a loop (iteration that has a condition at the beginning)
    - Loop: A method used in programming to repeat a set of instructions.
    - Iteration: In computer programming, this is a single pass through a set of instructions.
    - Condition: In computing, this is a statement or sum that is either true or false. A computation depends on whether a condition equates to true or false.
  - FOR: a counting loop (iteration)
  - REPEAT - UNTIL: a loop (iteration) that has a condition at the end
  - IF - THEN - ELSE: a decision (selection) in which a choice is made

Using pseudocode

- Pseudocode can be used to plan out programs.
- Planning a program that asks people what the best subject they take is, would look like this in pseudocode

```
REPEAT
    OUTPUT 'What is the best subject you take?'
    INPUT user inputs the best subject they take
    STORE user's input in the answer variable
    IF answer = 'Computer Science'
    THEN
        OUTPUT 'Of course it is!'
    ELSE
        OUTPUT 'Try again!'
UNTIL answer = 'Computer Science'
```

Flowcharts

- A flowchart is a diagram that represents a set of instructions.
- Flowcharts normally use standard symbols to represent the different types of instructions.
- These symbols are used to construct the flowchart and show the step-by-step solution to the problem.

Some concepts used in algorithms:

* DATA. All of the info. by computers is called DATA. They can be of 2 types.
  1 - Numerical data: Integers, rational numbers, etc.
  2 - Alphanumeric data: Letters, symbols, etc. (no numerical values)
  ↳ shown in quotes (" ") in algorithms and programs

STRINGS.

* Variables: Data structures that can take different values in a program.
* Constants: — " — — " — fixed values throughout the program.

Example: Calculating the end of semester grade of a student. (and printing)

1. Start
2. midterm_weight = 0.4
3. Read the number of the student (num)
4. Read the midterm grade of the student (midterm)
5. Read the final grade of the student (final)
6. Average = midterm_weight * midterm + (1 - midterm_weight) * final
7. print (num) and (Average) grade
8. Stop.

Assign: You can assign a value or the result of an expression / calculation to a variable or constant.

variable / constant = expression

Example:

$c = a + b$; $a = 1, b = 2, c = 0$

↳ $c = 1 + 2$

↳ $c = 3$

Example:

$S = 1$;

$S = S + 1$;

$= 1 + 1$

$S = 2$

***Counter:** Example: Write an algorithm that prints numbers from 1 to 5 on the display

1. Start ✓
2. i = 1
3. If i > 5 go to 7.
4. print i ✓
5. i = i + 1 ✓ 3+1=4
6. Go to 3
7. Stop

old i	display	new i	
1	1	2	✓
2	2	3	✓
3	3	4	✓
4	4	5	✓

Loop: If you need to execute certain blocks of codes for a number of times, or until a condition is satisfied, you use LOOPS.

Ex:

1. Start
2. N = 5 ✓
3. T = 0 ✓
4. S = 0 ✓
5. If S > N-1 go to 10
6. S = S + 1
7. Enter (A) ✓ 11
8. T = T + A 9+11 = 20
9. Go to 5.
10. Average = T/N
11. Print Average
12. Stop ✓

5 > 4

Iteration num.	S_old	T_old	A	S_new	T_new	Avg.
1	0	0	3	1	3	-
2	1	3	1	(2)	4	-
3	2	4	5	3	9	-
4	3	9	11	4	20	-
5	4	20	35	(5)	55	-
6						55/5 = 11

Some rules: 2/3/21

(1) **When you start** writing down your algorithm,

- always start w/ "start"
- always end w/ "stop"

(2) All the steps must be numbered, ordered in a logical way,
All the variables and constants used in any step must be defined before reaching to that step.

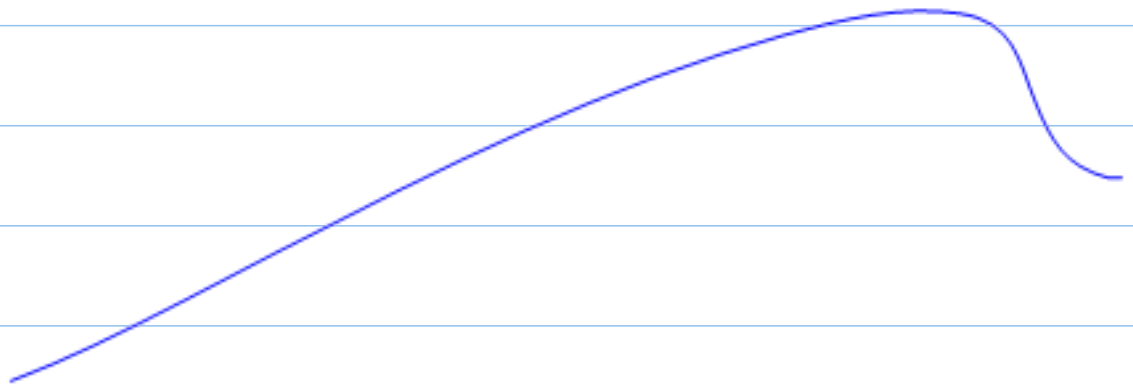
(3) There should be a finite number of steps

(4) The expressions must be definite (not vague / unclear)

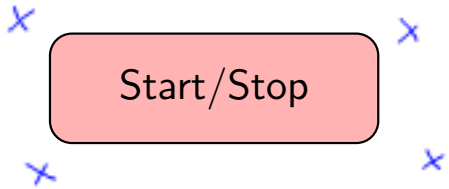
"If the result is big" big? ✗

"If the result is greater than 100..." ✓

FLOW CHARTS



Common flowchart symbols



- This shape tells you where the flowchart begins and ends.
- In other words, it shows the entry point of your flowchart and the exit point.

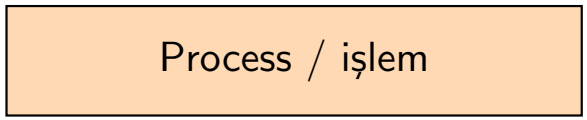
Common flowchart symbols

* The variables and constants that are going to be entered are written in the parallelogram.

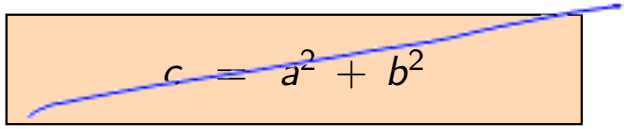
- Veri bilgi girişi
- This takes the shape of a parallelogram.
- The Data object, often referred to as the I shows the Inputs to a process.

* multiple inputs can be written inside, seperated w/ commas.

Common flowchart symbols



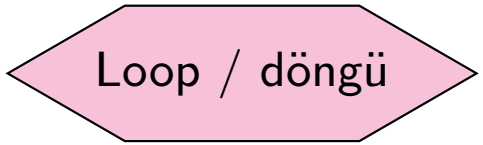
- Represents a process, action, or function.
- It's the most widely-used symbol in flowcharting.



Again, more than one process can be written, but needs to be seperated by a comma.

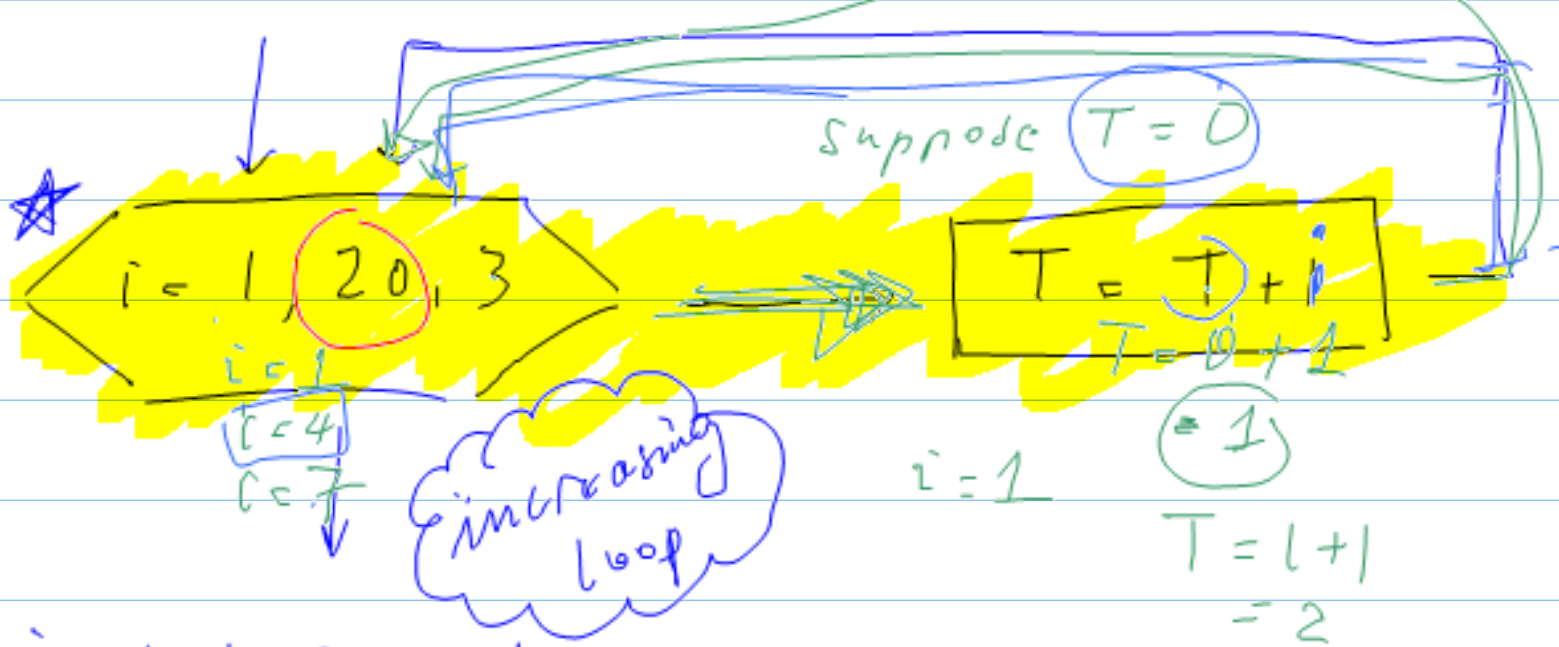
```
C = (a^2 + b^2)^(1/2)
```

Common flowchart symbols



- A loop is a sequence of instructions that is continually repeated until a certain condition is reached.

* The initial and final values, and step size are written inside
* Two types of loops — INCREASING
  — DECREASING



i starts from 1

increases by 3 (step size is 3)

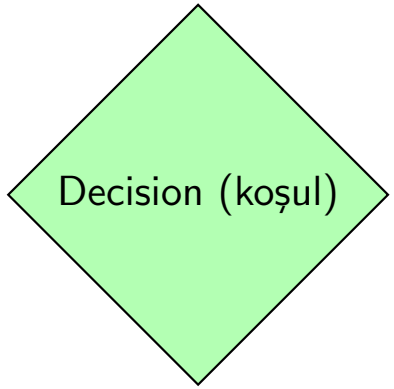
i = 1, 4, 7, 10, 13, 16, 19, ~~22~~

j = 30, 4, -2

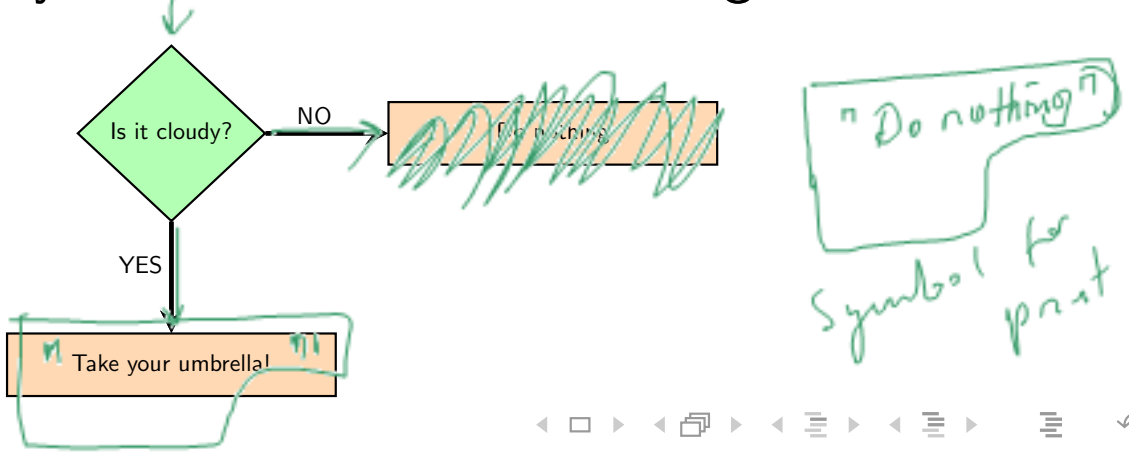
j : 30, 28, ... , 8, 6, 4

-2 -2 -2 -2

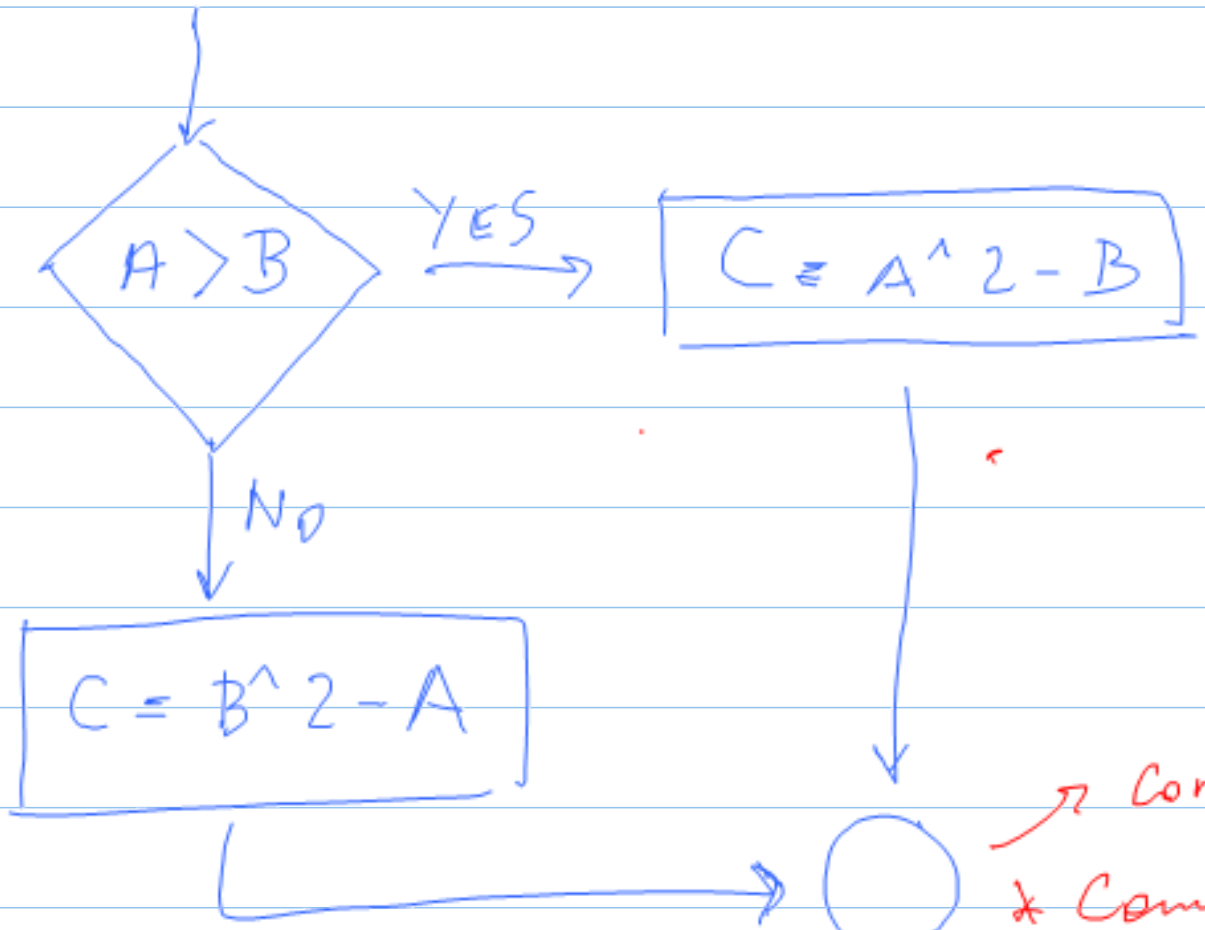
Common flowchart symbols



- A decision asks a question. The answer to the question determines which arrow you follow
- If it is cloudy, follow the arrow down near the word **Yes**. If it is not cloudy, follow the arrow to the right near **No**



Decision (ctd.)



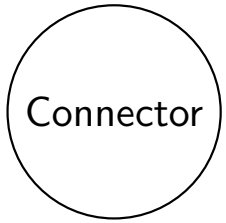
Connectors:

* Combine different flows into one flow
* When you run out of paper, put a connector ....

PRINT : "Istanbul", A

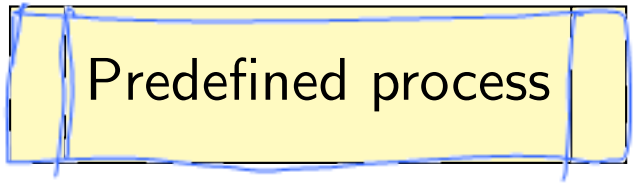
it is used to print some data or results on the screen.

Common flowchart symbols



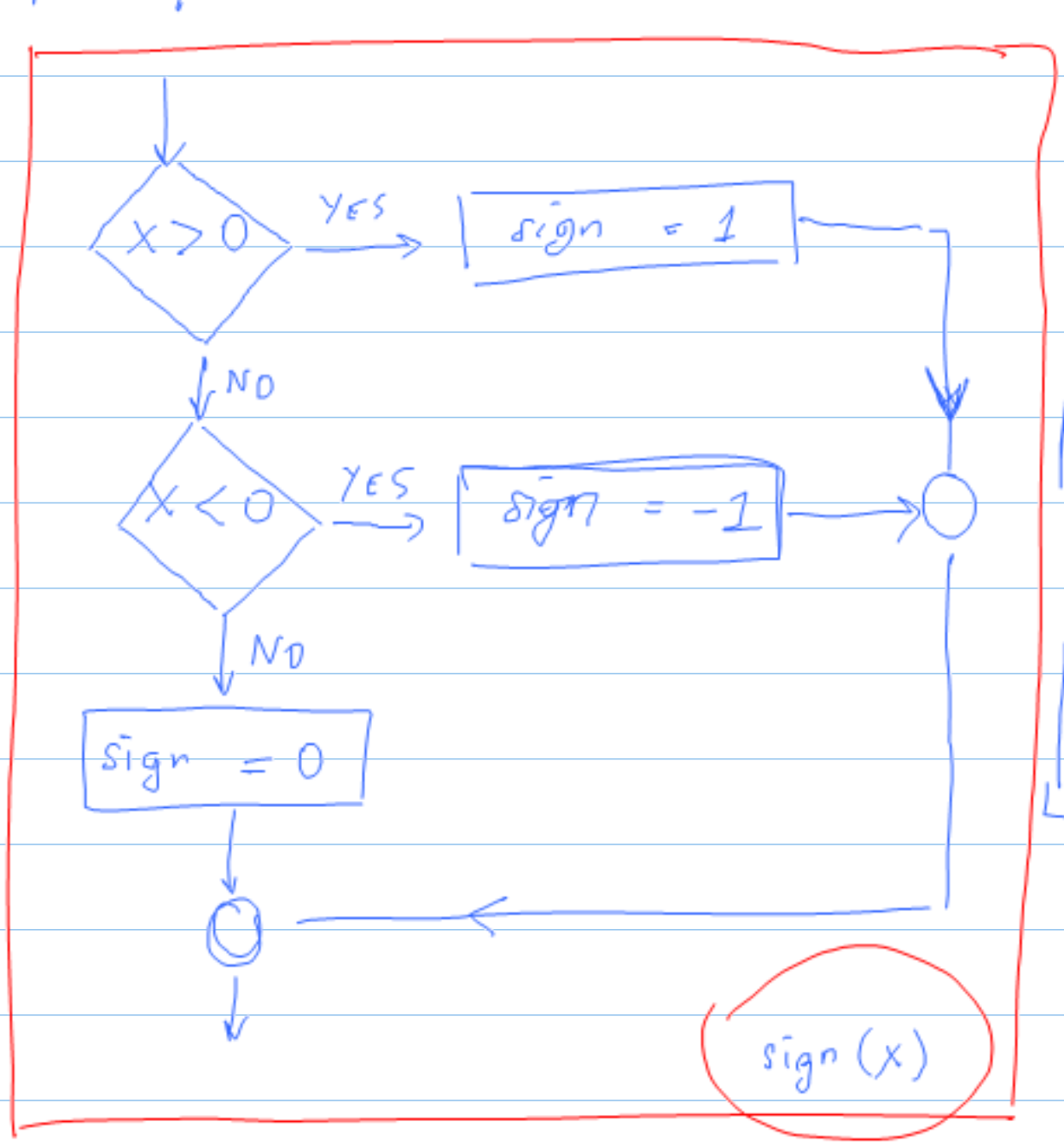
- Represents a jump/flow from one point in the sequence to another.

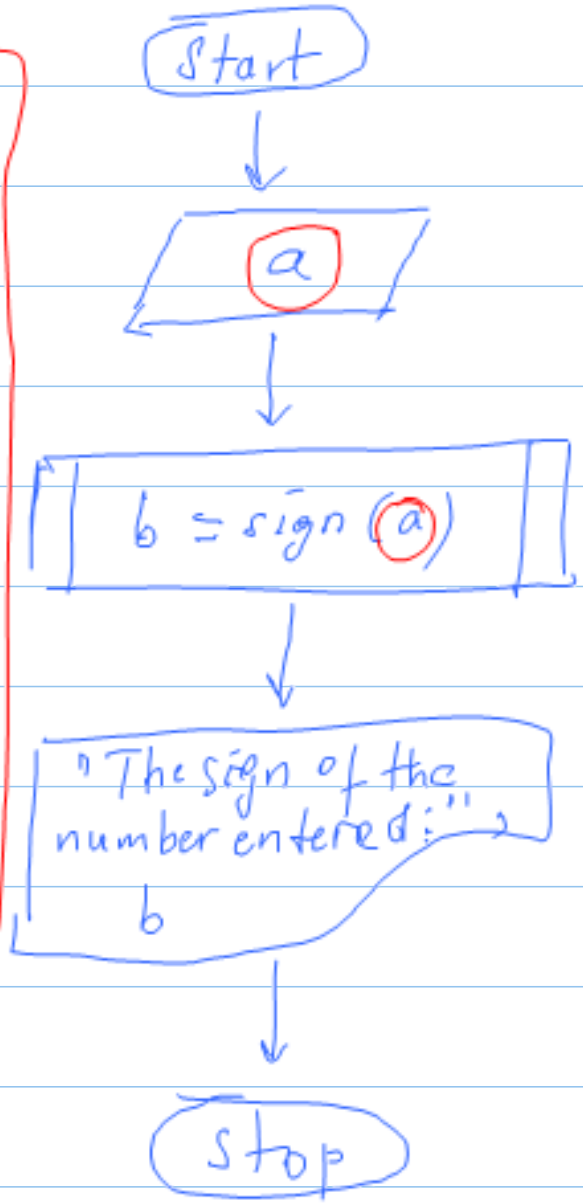
Common flowchart symbols



- Önceden tanımlanmış işlem
- One or more named operations or program steps specified in a subroutine or another set of flowcharts.

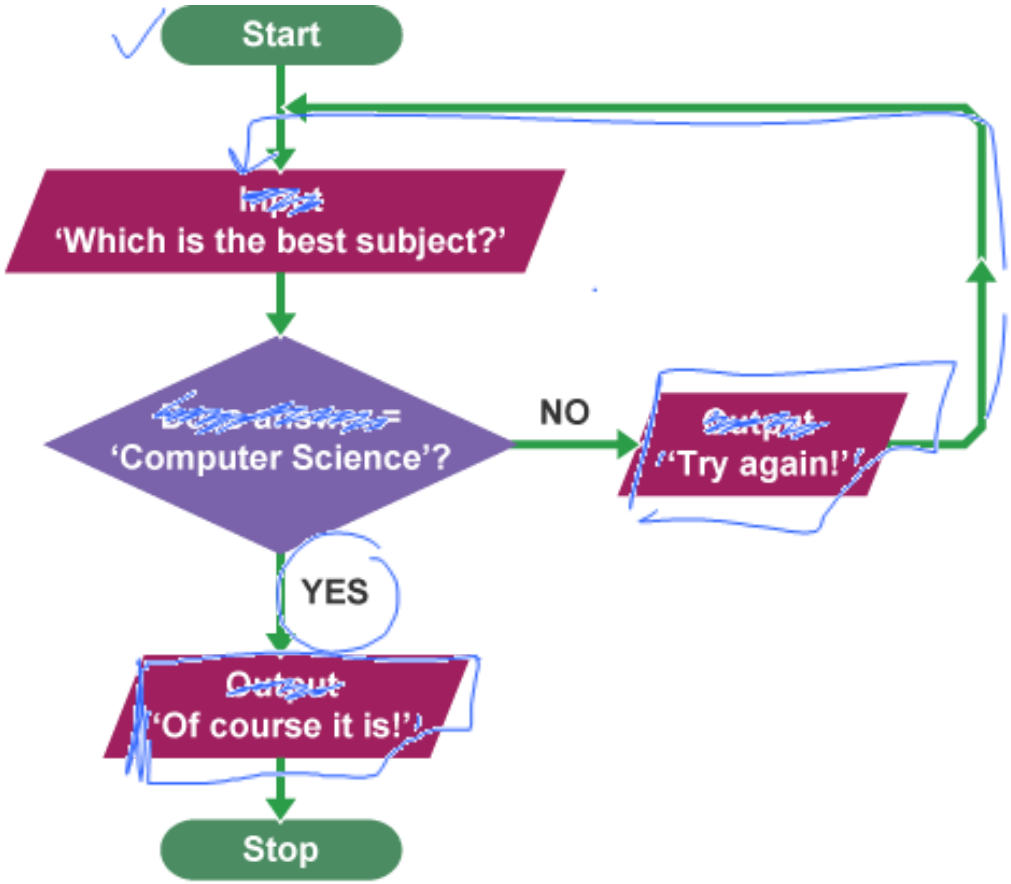
Predefined process:





Using flowcharts

- Flowcharts can be used to plan out programs.
- Planning a program that asks people what the best subject they take is, would look like this as a flowchart:

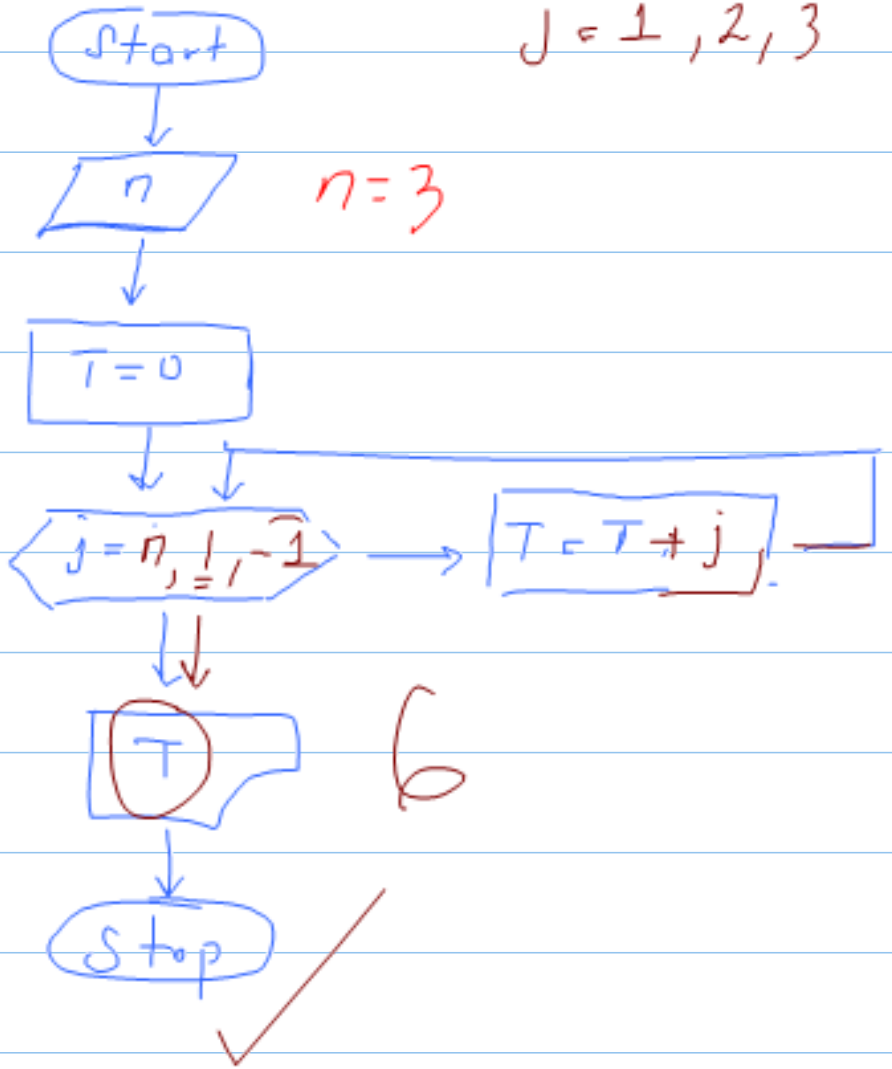


Advantages of algorithms/flowcharts

- makes it easier to write a program
- reduces errors when coding
- decreases the time needed to write the program
- as the flow of the processes are easily seen, it makes it easier to control the program and check for/spot errors
- makes it easier to do additional arrangements/designs

Nested loops...

A simple example for a loop



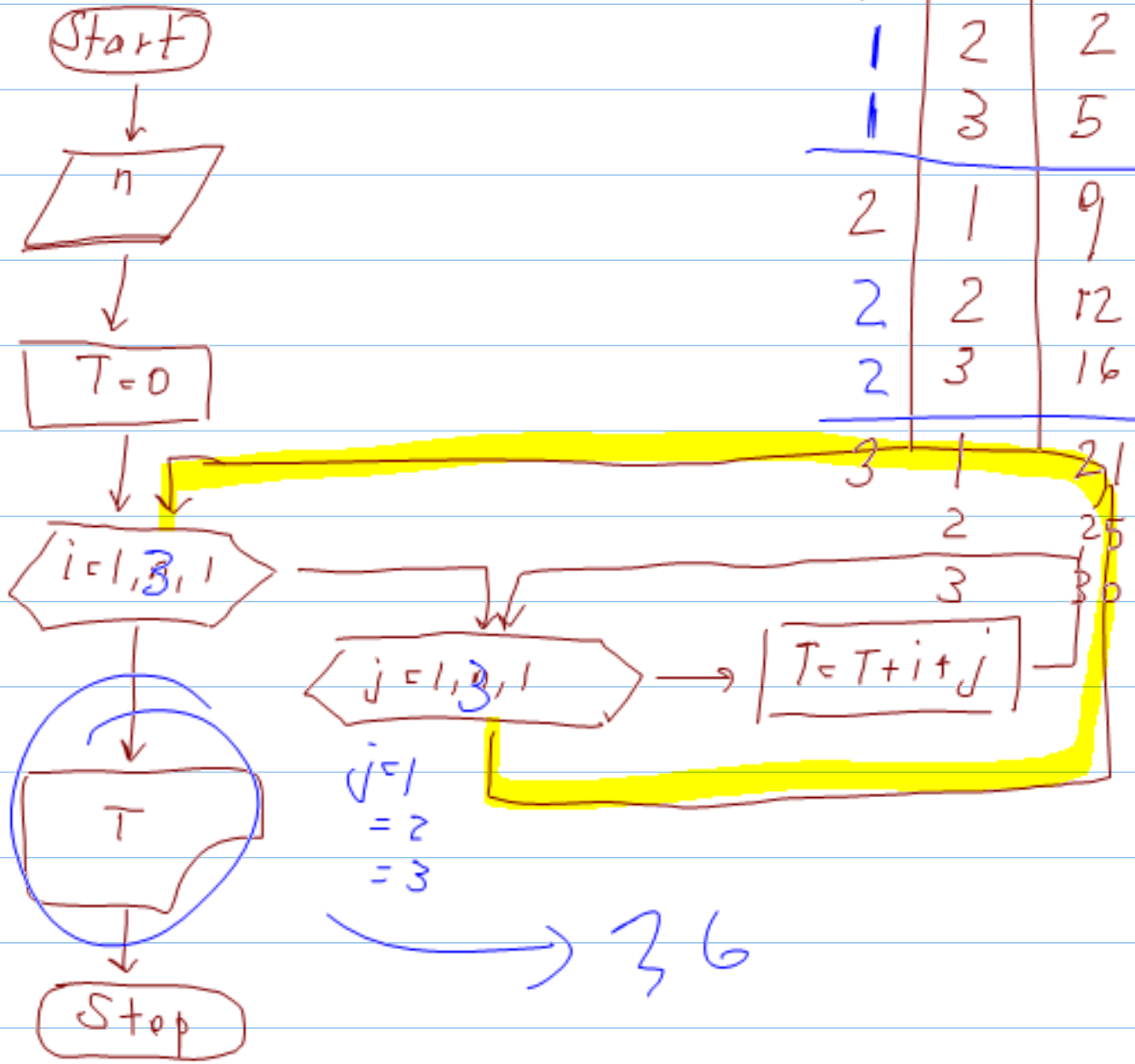
$j = 1, 2, 3$

$n = 3$

j	T_{old}	T_{new}
3	0	$T = 0 + 3 = 3$
2	3	$T = 3 + 2 = 5$
1	5	$T = 5 + 1 = 6$

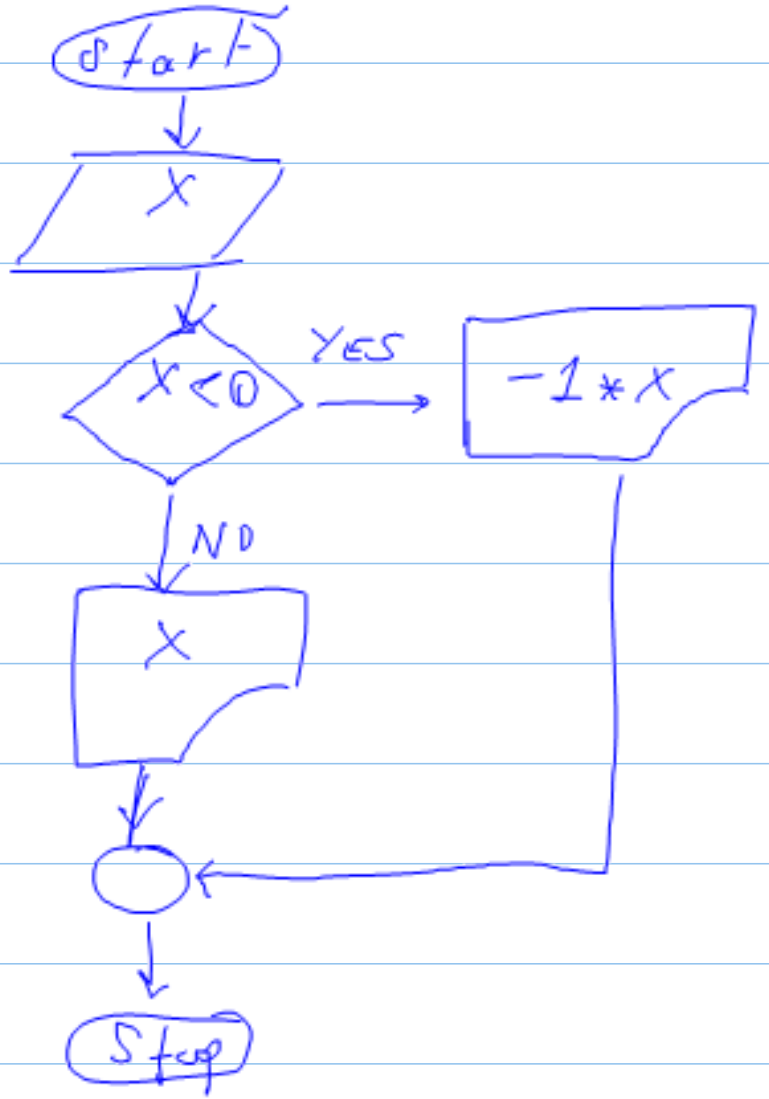
NESTED LOOPS

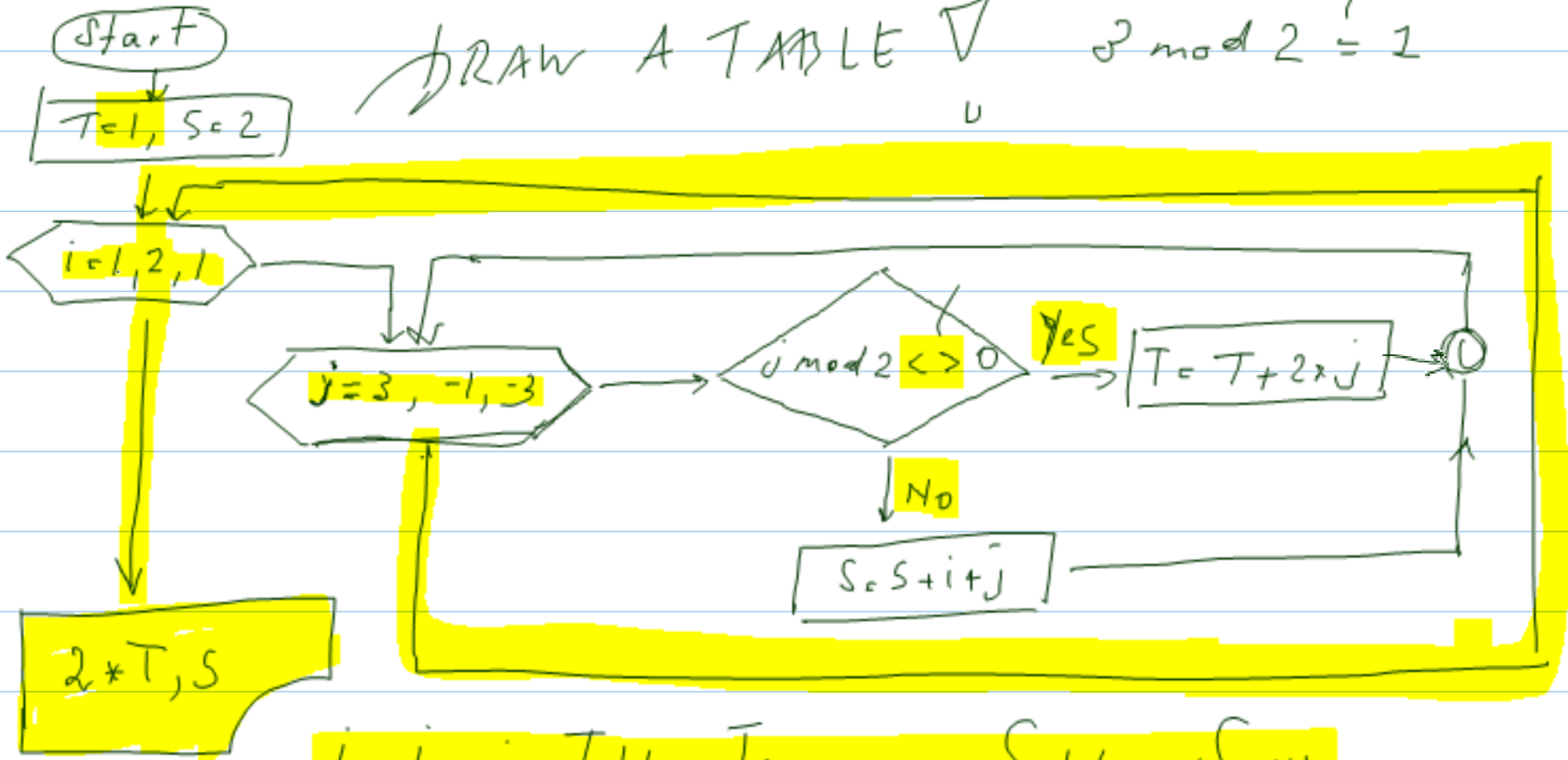
n=3



i	j	T old	T new
1	1	0	T = 0+1+1=2
1	2	2	T = 2+1+2=5
1	3	5	T = 5+1+3=9
2	1	9	T = 9+2+1=12
2	2	12	T = 12+2+2=16
2	3	16	T = 16+2+3=21
3	1	21	T = 21+3+1=25
	2	25	T = 25+3+2=30
	3	30	T = 30+3+3=36

Ex: Draw a flowchart of the program that prints the absolute value of a number assigned by the user:





DRAW A TABLE ∇ $3 \bmod 2 \stackrel{?}{=} 1$

i	j	T_{old}	T_{new}	S_{old}	S_{new}
1	3	1	T=1+2*3=7	2	2
1	0	7	7	2	S=2+1+0=3
2	3	7	T=7+2*3=13	3	3
2	0	13	13	3	S=3+2+0=5

26, 5