

Algorithms and algorithmization of tasks

doc.Ing. Marcela Hallová, PhD.

Algorithm and algorithmization of tasks

- **Why algorithms?**

- The possibility to implement the given sequence of steps by another person, or by machine.
- The procedure expressed using precisely defined rules and procedures becomes understandable not only for the author, but also for other implementers.

What is an algorithm?

The exact prescription of the procedure for solving the task.

The process of transforming input data into a result.

A sequence of elementary steps (operations) that are performed in a certain order.

The origin of the word algorithm

- **9th century** – Muhammad ibn Músá Al-Chwárizmí – geometric solution of quadratic equations - devised a simple algorithm for multiplying a two-digit number by a one-digit number.
- **Algorithmic way of solving tasks** – Egypt – Moscow Papyrus (about 1850 BC) - procedure for calculating the volume of a truncated pyramid.



What can be an algorithm

instructions for assembling
furniture from individual parts,

the procedure for solving a
mathematical problem,

recipe for food preparation,

a program written in any
programming language...

Properties of algorithms

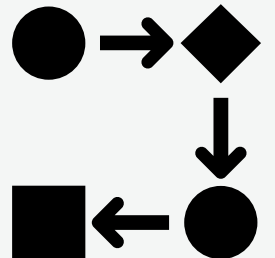
Determination

Finiteness

Mass output

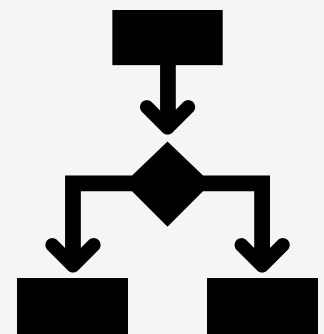
Determination

- the algorithm determines, i.e. it precisely determines the process of transforming input data into results,
- in each step, it must be specified exactly which step is performed next,
- performing operations does not depend on the executor of the algorithm.



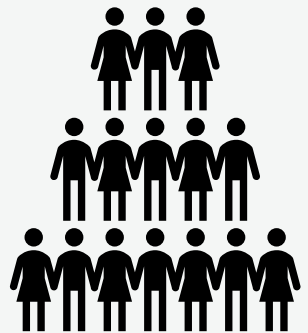
Finiteness

- for any set of input data from a certain set M , the algorithm always leads to the desired result after a finite number of steps,
- we call the set M "the area of applicability of the given algorithm".



Mass output

- the algorithm must be constructed to solve a large, usually infinite, class of problems of the same type,
- it must be a description of the solution not of one specific task, but of a whole group of related tasks that differ only in the values of the input data.



Other properties of algorithms

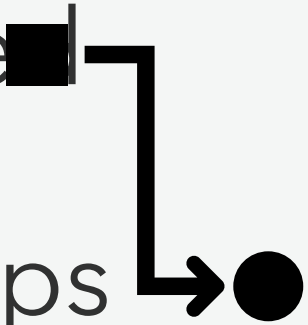
- In addition to the basic properties (determinism, mass and efficiency), the algorithm should be:
 - **elementary,**
 - **effective,**
 - **readable,**



Elementary algorithm

- each procedure can be implemented in different ways,
- when designing the algorithm, it is necessary to ensure that the individual steps are understandable and unambiguous for the algorithm implementer.

Efficiency of the algorithm

- it is determined especially with regard to the necessary computing time and the required memory capacity,
 - an effective algorithm is a sequence of steps that solves the given problem with the minimum number of resources used in the shortest possible time.
- 

Readability of the algorithm

- it is advisable to design an algorithm that is appropriately divided into smaller, relatively independent, logically connected units,
- a structured algorithm is easier to understand and modify.

Algorithmization stages of tasks

Task formulation.

- clear and unambiguous wording and identification
- setting the goal of solving the task

Task analysis.

- finding a solution algorithm

Compilation of the solving algorithm.

- synthetic stage - description of the logic and procedure of the solution
- the result is a solving algorithm

Algorithmization

The ability to actively create algorithms designed for non-thinking devices.

It is an essential part of being able to program on a computer.

To create effective and correct algorithms, it is not enough just to master an algorithmic or programming language.

Knowledge of the problem environment and experience in formulating algorithms is required.

Programming

- A program is an algorithm written in a programming language.
- Programming is a constructive thought, but also a practical activity, when we create new program products that can be implemented on a computer.
- By programming, we learn to think, organize our thoughts and be able to entrust the computer with their implementation ...

Creating a program - activities

Algorithmization of the given problem - determination of input and output conditions.



Creation of a program (program product) and appropriate program documentation.



Write and debug the program directly on the computer.

Algorithmic language

- To write algorithms, it was necessary to create formal languages - artificially created especially for this purpose, the so-called algorithmic languages.
- We differentiate between two groups :
 - **Graphical algorithmic languages** - flowcharts, structure diagrams ...
 - **Linear algorithmic languages** - e.g. verbal notation in the national language, programming language ...

Write the algorithm

- The algorithm can be e.g. a calculation process, a description of a geometric structure, a legal regulation, instructions for performing a certain activity, etc., can be expressed in different ways:
 - verbal description,
 - mathematical symbols (equations, matrices, formulas, etc.),
 - construction procedure,
 - graphic signs.

Flowchart

- graphical form of notation of the algorithm,
- block diagram of a certain process, which expresses its structure and the continuity of individual operations,
- we use graphics markers to draw flowcharts.

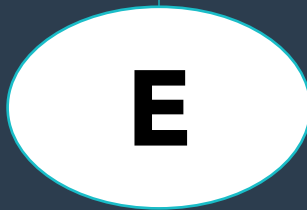


Flowcharts symbols

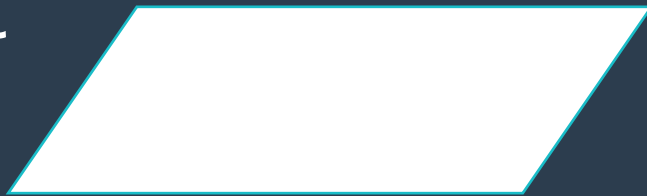
- symbol for processing
- decision tag



- Start/End

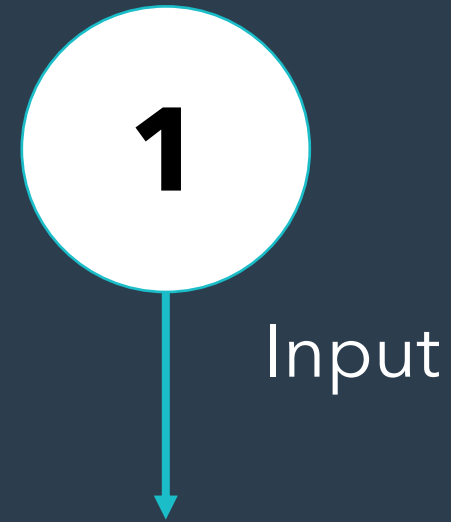
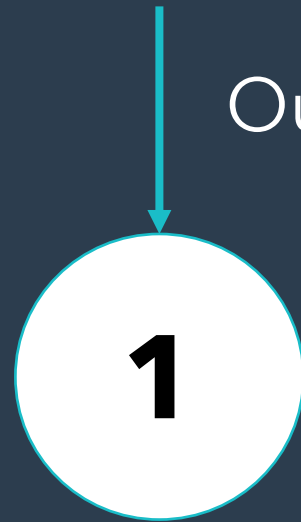


- Input/Output



Flowcharts symbols

- connector



- connector lines



Types of flowcharts



Gross

- solving extensive tasks that are divided into several subtasks (subprograms)

Analytical

- more detailed solution algorithms

Flowcharts of program

- detailed solution procedure - there is usually one operation in each block

Flowcharts of the program

Straight – there is no alternative procedure, the process is straightforward (e.g. addition of 4 numbers).

Flowcharts with decision:

Without a cycle -
without repetition
(e.g. a maximum of
3 numbers)

With cycle-
with repetition (e.g.
sum of elements of
a vector, matrix)

Straight flowcharts

The simplest type - consists only of blocks.

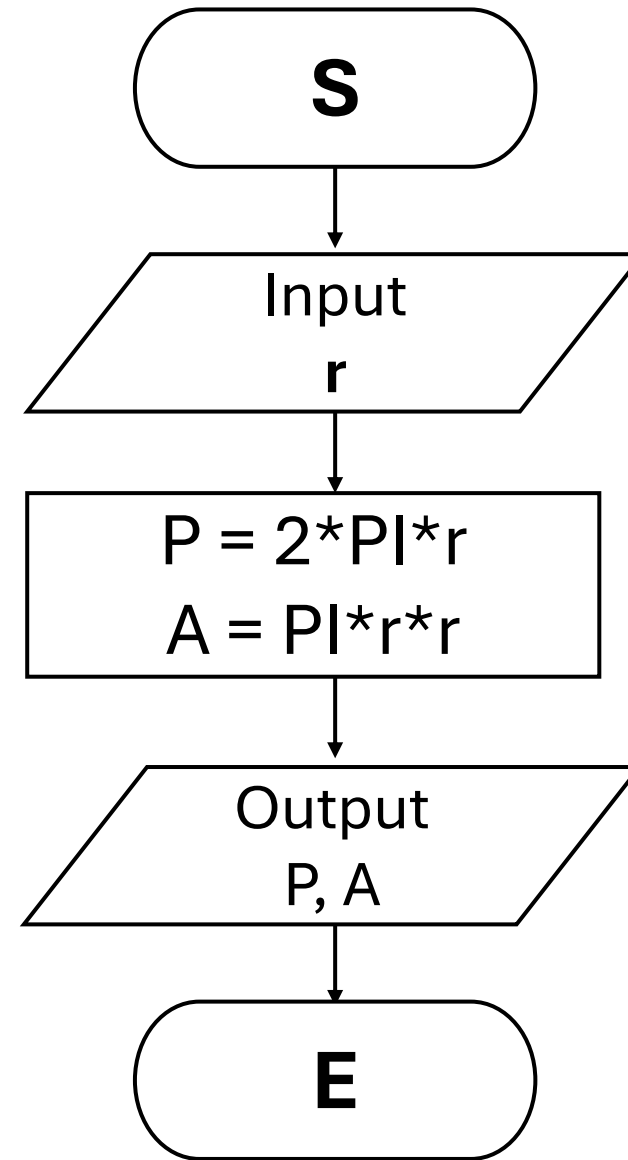
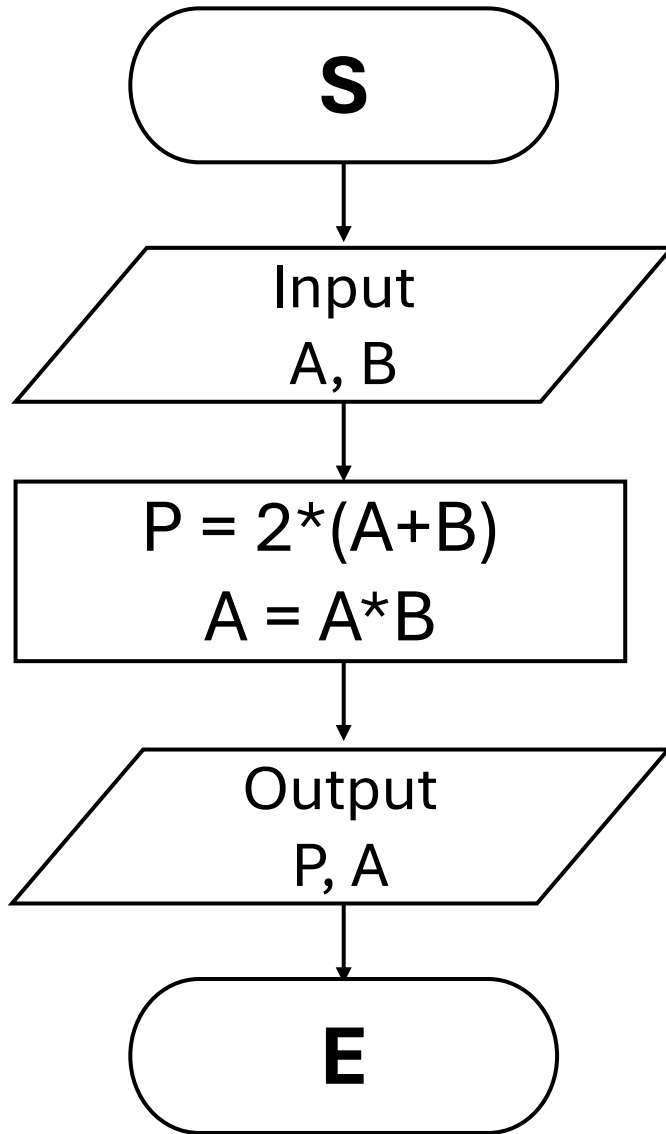
In the case of direct flowcharts, there must be no branching of the algorithm, nor a return (cycle).

Straightforward flowcharts are often part of more complex flowcharts.

Examples

- Given a rectangle with sides A , B . Create a VD to calculate the perimeter and area of this rectangle. Use the variables P - the perimeter of the rectangle and A - area of the rectangle.
-

- A circle with radius r is given. Create a flowchart to calculate the perimeter and area of this circle. Use the variables P - the perimeter and A - the area of the circle.



Decision flowcharts without cycle

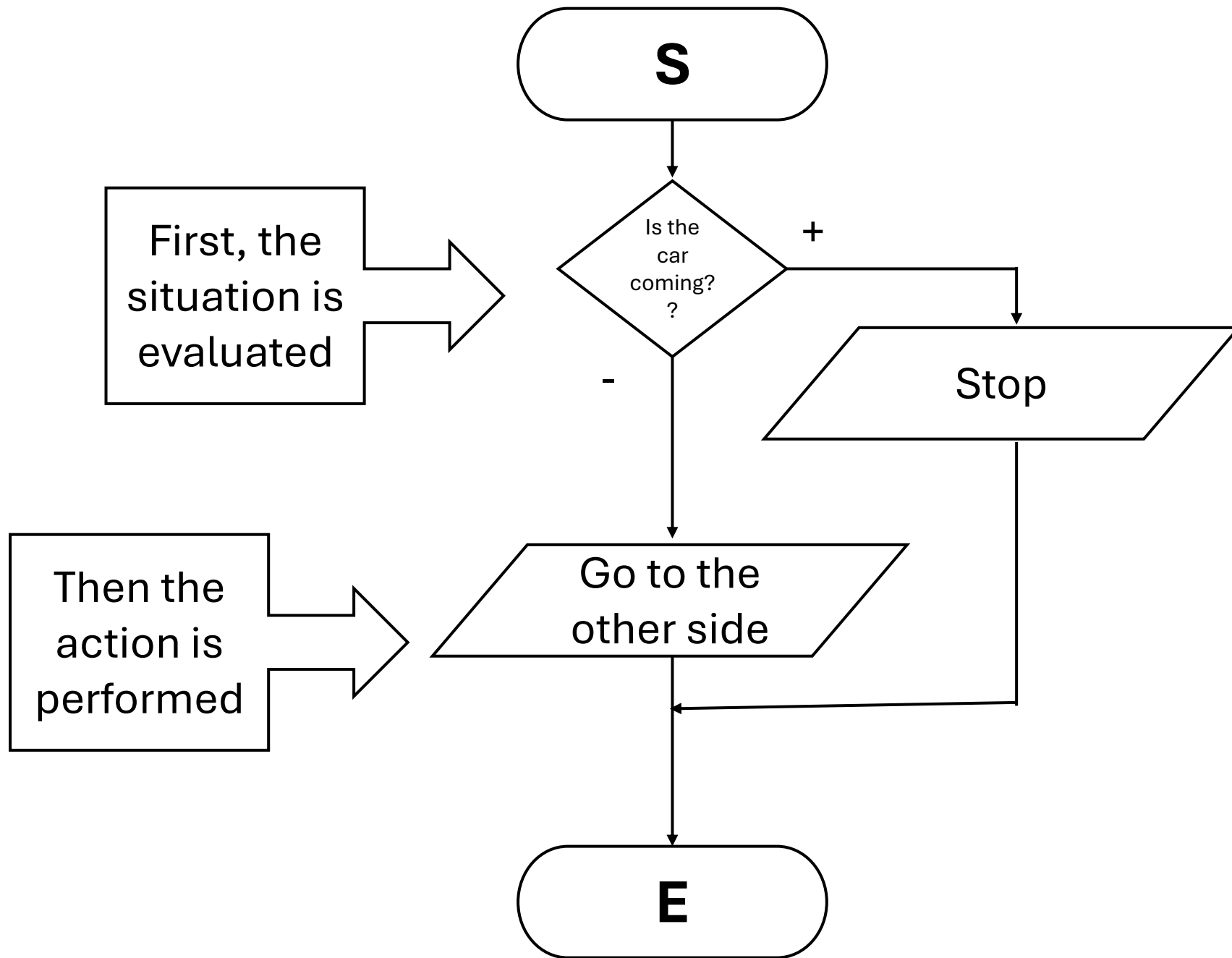
Branching flow diagrams contain decision markers representing the decision operations that branch the diagram, determining alternative courses of action.

Branching in flowcharts can be double (dichotomous), i.e. the decision marker has two outputs. The test result is yes (marked +) or no (marked –).

Branching in flow diagrams can also be triple (trichotomy), i.e. the decision marker has three outputs.

Examples

Imagine that you are standing on one side of the road, and you want to cross to the other side. First you have to look around to see if the car is running. If nothing works, you can pass without worry. If a car is moving, you must stand still.

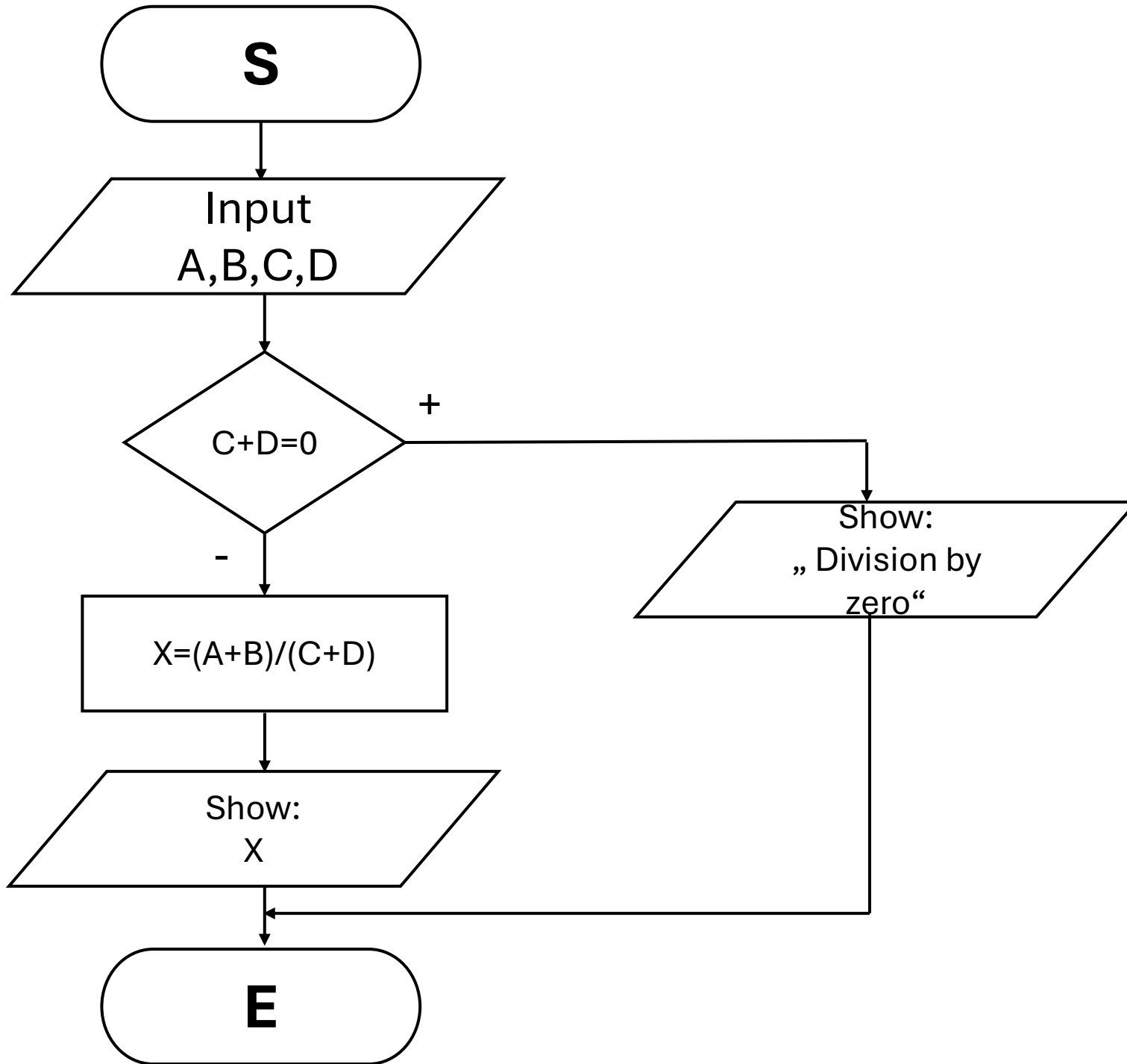


Examples

Create a flowchart to calculate the expression $X = \frac{A+B}{C+D}$

Replace the fractional line with a slash.

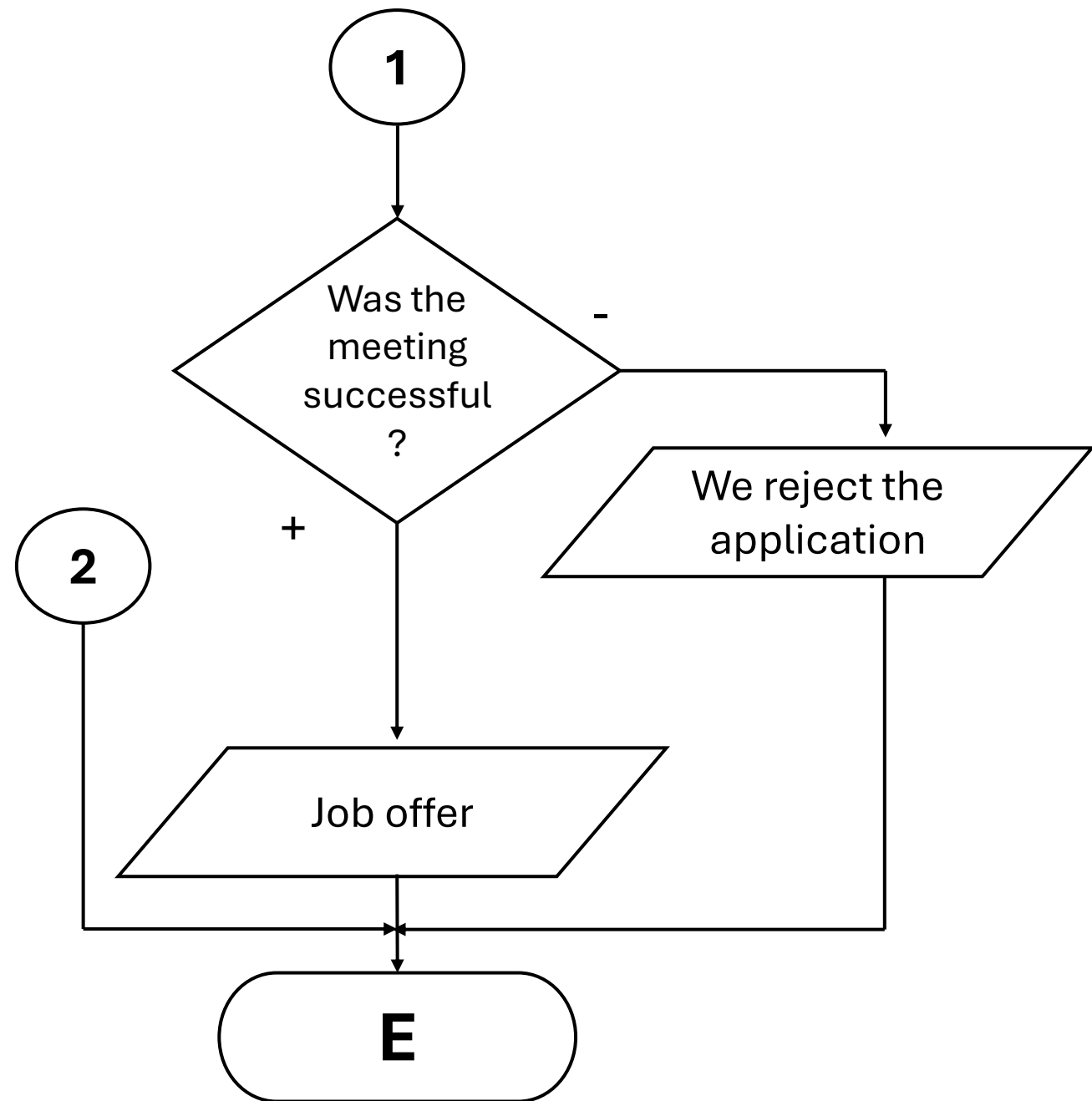
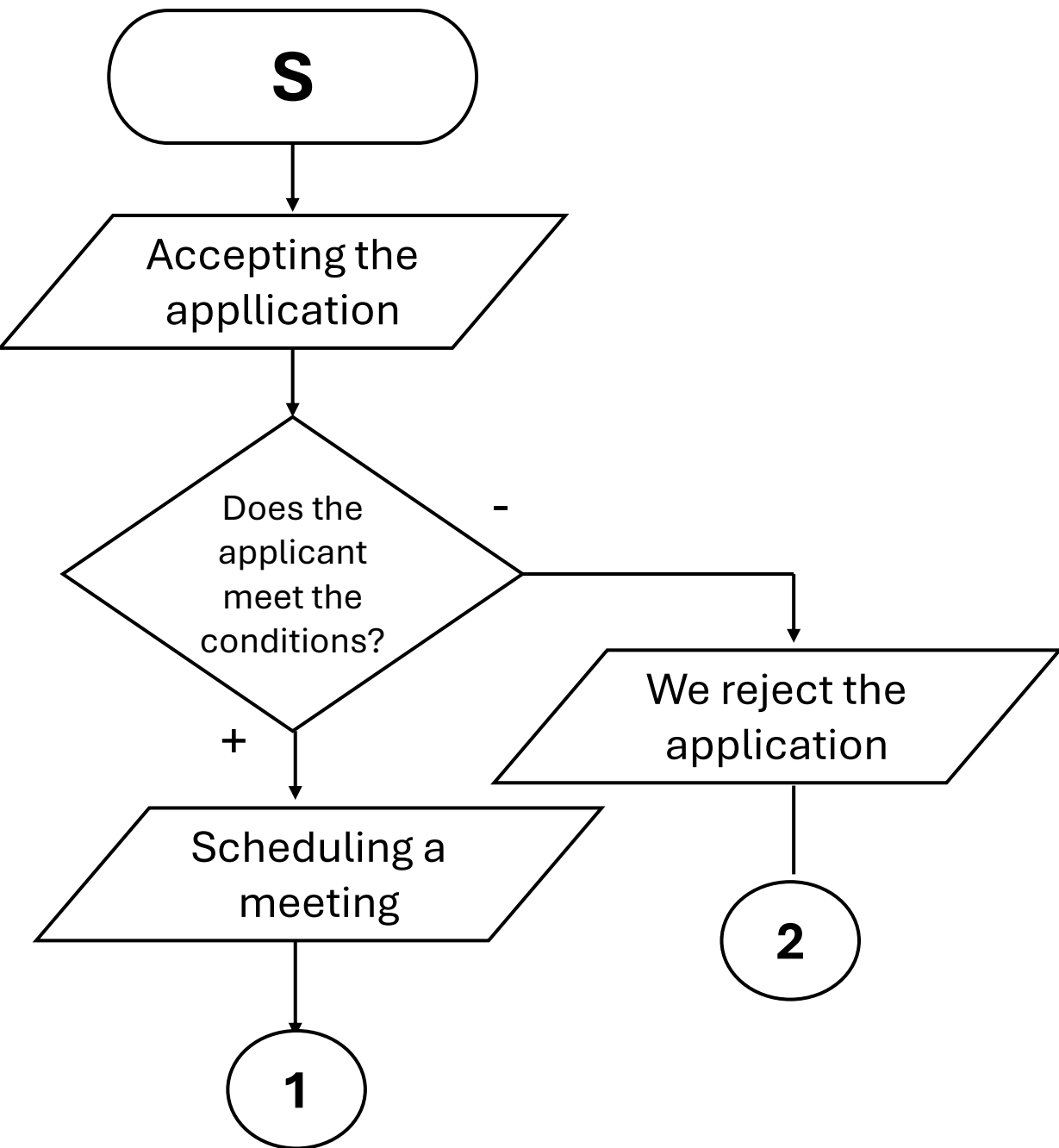
Important - it is not possible to divide by zero!



Example

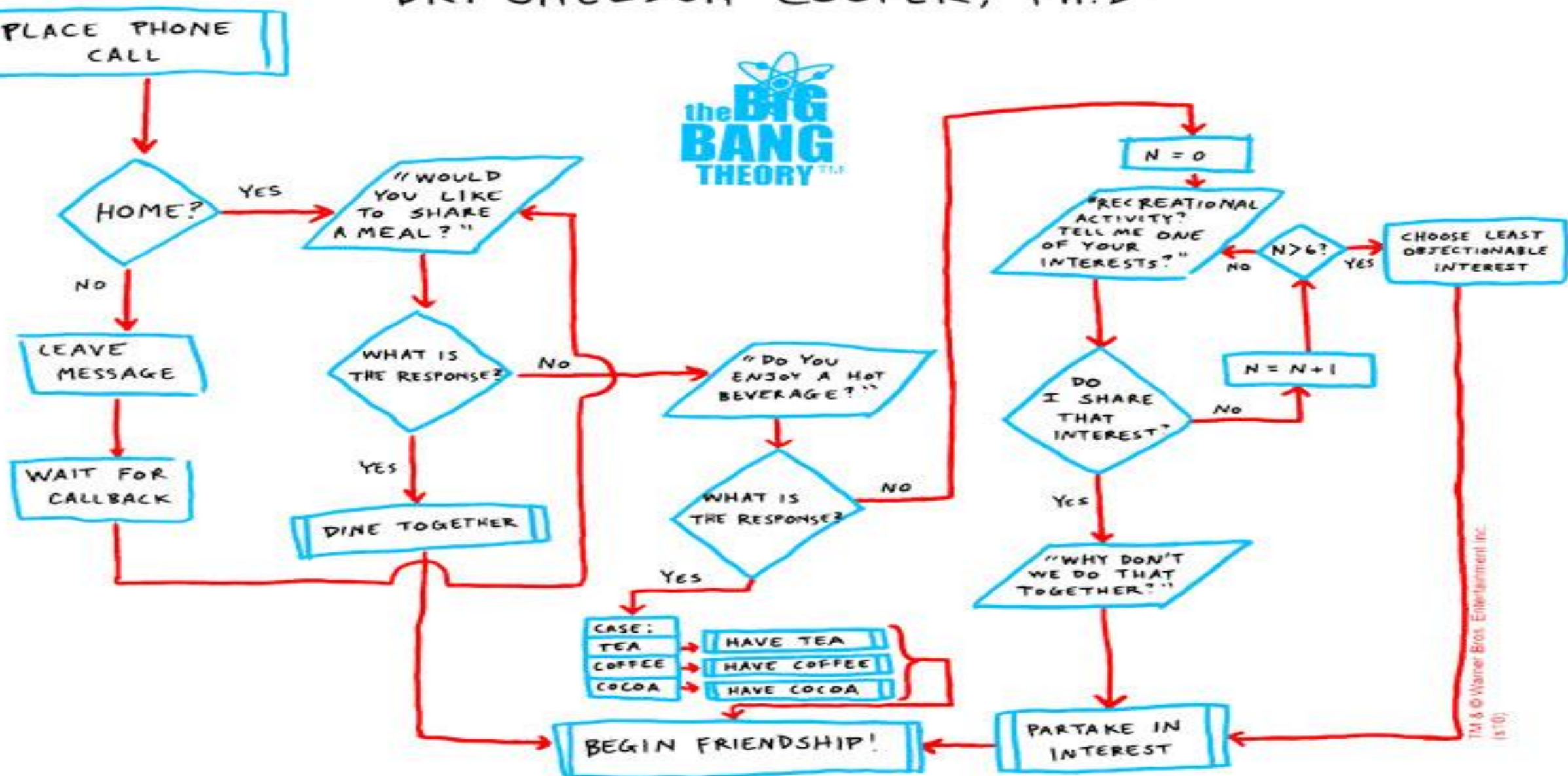
Create a flowchart for hiring an employee. The process begins with the received of an application from the applicant (introduction information). The first step in making a decision is to check the application to see if it meets the requirements for the position. If not, the request is rejected. If so, a meeting will be scheduled (input information after the first decision). The meeting is followed by an evaluation of the meeting. Was the meeting successful? If so, we will offer the applicant a job. If not, we reject the applicant.





THE FRIENDSHIP ALGORITHM

DR. SHELDON COOPER, Ph.D



Examples

- 1) Use the flowchart to determine whether the number **A** is positive, negative, or equal to zero.
- 2) Read two numbers as input and then determine which number is smaller and which is larger. Finally, display them - first the smaller number and then the larger. Use variables **A, B**.
- 3) Find the largest number from three numbers **X, Y, Z**. First compare the first two numbers, then compare the larger of them with the third number.



Thank you for
your attention!

$$\frac{a(\frac{b}{c})}{c} = \frac{ab}{c^2}$$
$$\frac{(\frac{a}{b})}{c} = \frac{a}{bc}$$
$$\frac{a}{\frac{b}{c}} = \frac{ac}{b}$$
$$\frac{a}{\frac{b}{c}} + \frac{c}{d} = \frac{ad+bc}{bd}$$

$$X^2 - 4X + 5 \leq 5$$
$$X^2 - 4X \leq 0$$

$$n(B \cap C) = 22$$
$$n(B) = 68$$
$$n(C) = 84$$
$$n(B \cup C) = n(B) + n(C) - n(B \cap C)$$



$$= \frac{1+3+3+6+8+9}{6} = 5$$
$$= \frac{2+4+4+8+12}{6} = 30$$
$$= \frac{4+7+1+6}{6} = 18$$

$$b^x = x$$
$$x = \frac{\log_b x}{\log_b a}$$
$$x' = r \log_b x$$
$$y = \log_b x + \log_b x$$
$$\frac{x}{y} = \log_b x - \log_b y$$



$$26 = 6xy$$
$$2x + 2y = 20$$

$$(100^2)a + 100b + c = 0$$
$$10000a + 100b - 5000 = 0$$

$$a_n = \frac{1}{2^{n-1}} = \frac{1}{2^{10-1}}$$
$$= \frac{1}{2^9} = \frac{1}{512}$$

$$y = ax + b$$

$$v = \dots$$
$$A = \pi r^2 h$$

$$\cos(B) = \frac{y}{r}$$

$$|a| = |-a|$$
$$|a| \geq 0$$

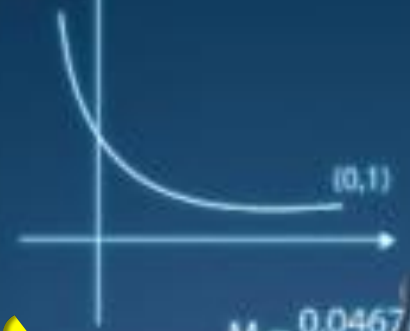
$$ab+ac = a(b+c)$$
$$\frac{a(b/c)}{c} = \frac{ab}{c^2}$$
$$\frac{a}{\frac{b}{c}} = \frac{ac}{b}$$

$$2\pi rh$$
$$2\pi r$$
$$\pi r^2 h$$

$$a \geq 0$$
$$-d = a + c$$
$$d = a - c$$
$$b = ac - bd$$
$$i = a^2 + b^2$$
$$a^2 + b^2$$



$$M = \frac{0.046755 \text{ m}}{3 \text{ l}}$$



$$\frac{a+b}{c} = \frac{a}{c} + \frac{b}{c}$$
$$\frac{ab+ac}{a} = b+c, a \neq 0$$
$$\left(\frac{a}{b}\right) \div \left(\frac{c}{d}\right) = \frac{ad}{bc}$$

$$\frac{2}{x}$$
$$x\sqrt{3} = 8\sqrt{3}$$

$$f = \frac{R}{2}$$

$$+\frac{10}{15} + \frac{2}{5} + 3\frac{1}{3} = \dots$$