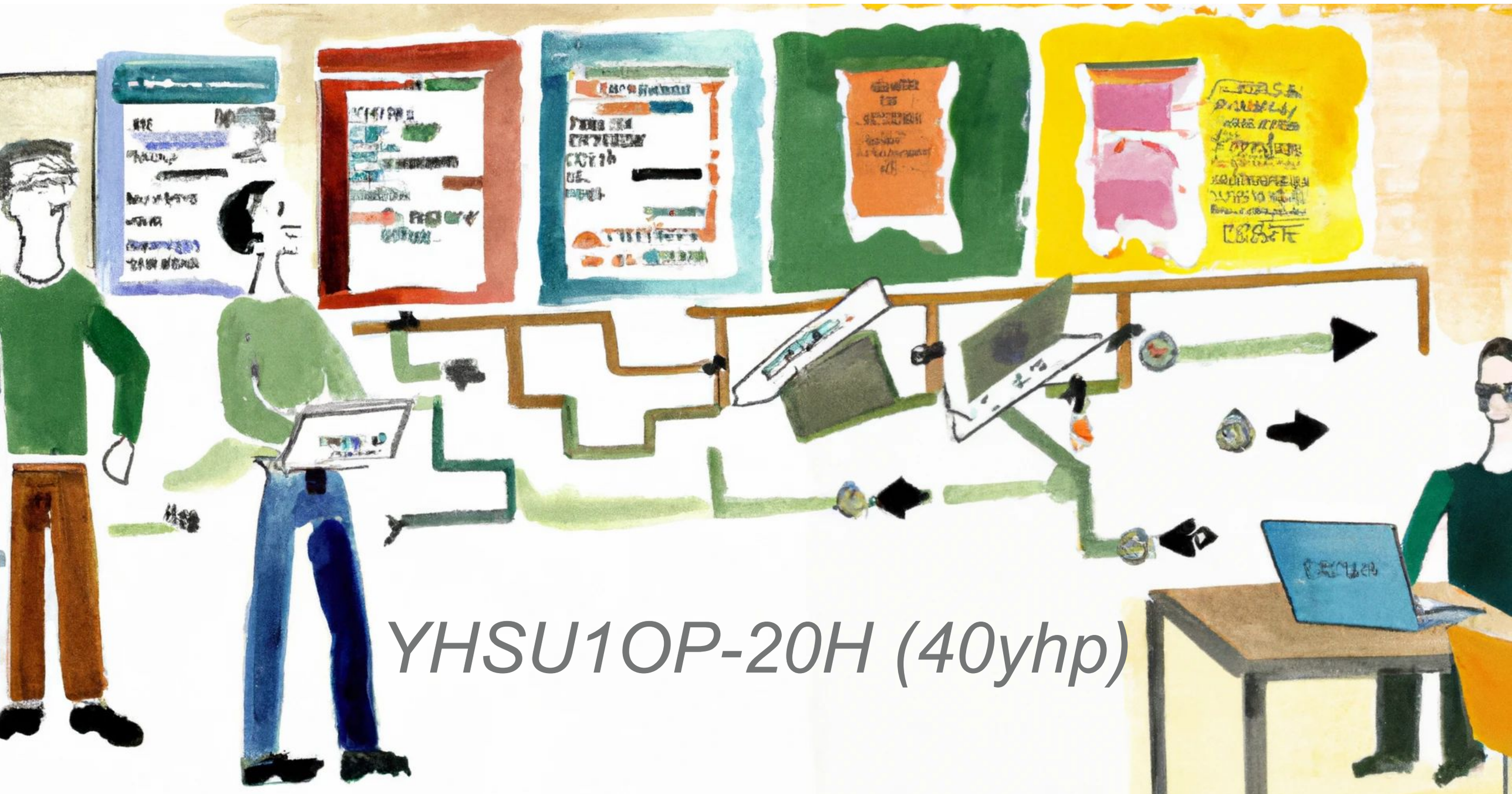
OBJEKTORIENTERAD PROGRAMMERING I C#.NET



YHSU1OP-20H (40yhp)

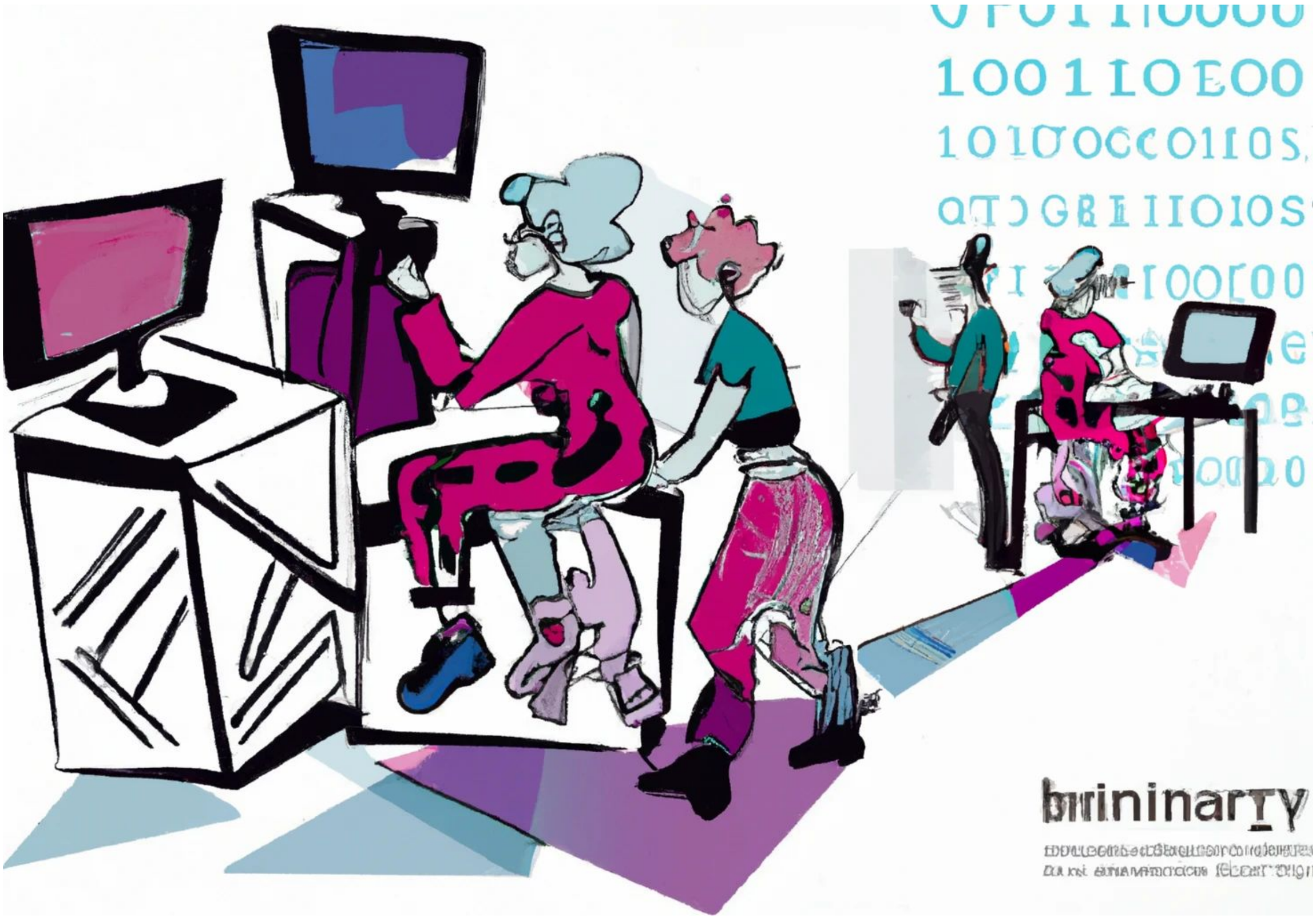
INTRODUKTION

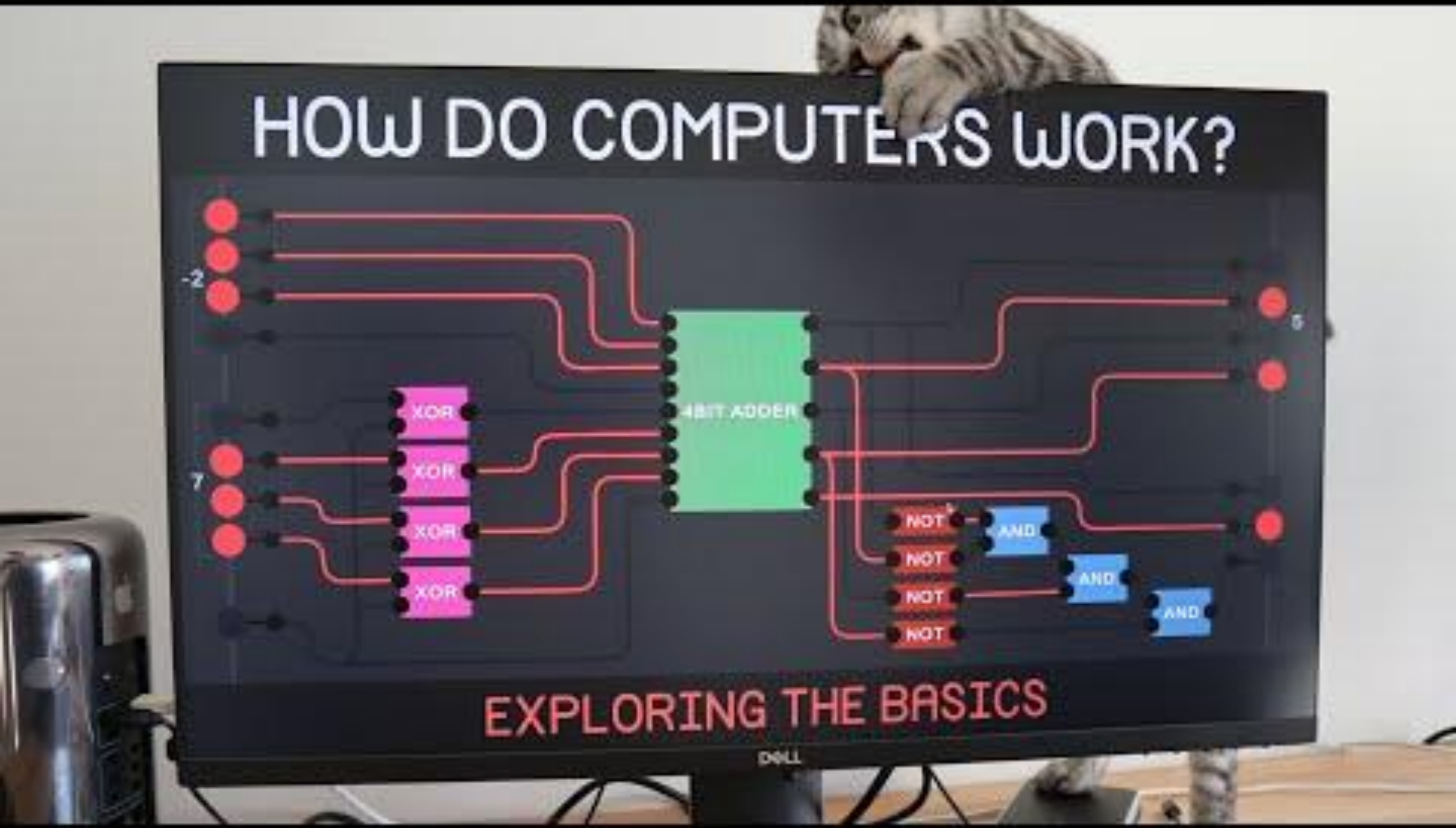
Vad är det vi håller på med?

Vi börjar med lite glosor!

- Ettor och nollor?
- Algoritm
- Programkod
- Applikation
- Kompilator/Parser
- Framework
- Miljö
- Källkod, Bytekod, Maskinkod

Ettor och nollor?




HOW DO COMPUTERS WORK?
-2
7
XOR
XOR
XOR
4BIT ADDER
NOT
NOT
NOT
NOT
AND
AND
AND
EXPLORING THE BASICS
DELL

ALGORITMER

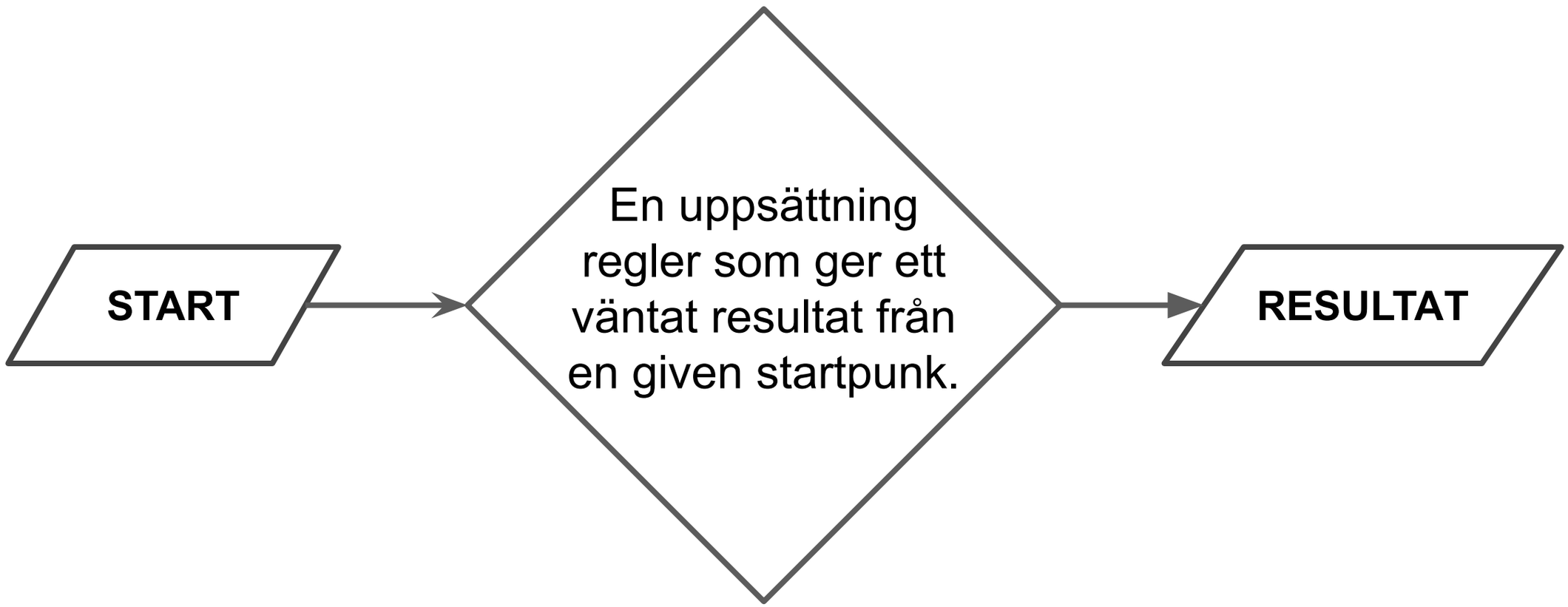
ALGORITMER

- Detaljerade steg-för-steg instruktioner för att lösa ett problem eller utföra en uppgift.

- Namnet kommer från den persiske astronomen och matematikern *Muhammad ibn Musa* ***al-Khwarizmi***

ALGORITMER

ALGORITMER


START
En uppsättning regler som ger ett väntat resultat från en given startpunk.
RESULTAT

ALGORITMER

Gör så här

1. Skala och skiva löken. Skala och finhacka vitlöken. Dela, kärna ur och skär paprikan i grova bitar. Dela, kärna ur och finhacka chilin. Skär tomaterna i grova tärningar.

2. Stek lök, vitlök och chili i rapsolja tills löken är gyllene.

3. Häll av och skölj bönorna.

4. Häll i tomater, paprika och kryddorna i grytan. Stek i 1 minut.

5. Tillsätt bönor och vatten så att det precis täcker. Låt koka ihop i ca 10 minuter.

6. Skär den kokta färskpotatisen i mindre bitar.

7. Tillsätt potatisen i grytan och blanda runt. Smaka av med salt och peppar.

8. Servera med gräddfil och koriander.

9. Glöm inte att matsortera dina lökskal och kärnhus till återvinning för mer hållbar matlagning och njut av din chiligryta! /Köket redaktionen

ALGORITMER : Bubble Sort

- Algoritm för att sortera en lista (exempelvis med tal) i ordning.
- Inte särskilt effektiv men lätt att förstå!

1. Jämför dom två första *elementen* med varandra
2. Flytta det största bakåt i listan
3. Gå sen vidare till dom två nästföljande
4. Upprepas tills en *iteration* sker där inga element flyttas

ALGORITMER : Bubbel Sort

6 5 3 1 8 7 2 4

ALGORITMER: Merge Sort

Sortera en lista med storlek *n*:

1. Dela upp listan i n lika stora dellistor (alla med längd 1)
2. Slå samman dellistorna parvis i sorterad ordning
3. Upprepa steg två, tills det bara finns en lista kvar

Den slutgiltiga listan är original-listan i sorterad ordning.

ALGORITMER: Merge Sort

6 5 3 1 8 7 2 4

ALGORITMER: Idag

- Facebook
- Google
- Deep Learning
- AI
- Självkörande bilar

ALGORITMER: Google

- Fred
- Intrusive Interstitials Update
- Mobilegeddon
- RankBrain
- Hummingbird
- Pigeon
- Payday
- EMD (Exact Match Domain)
- Panda
- Penguin
- Page Layout Algorithm

2. Penguin

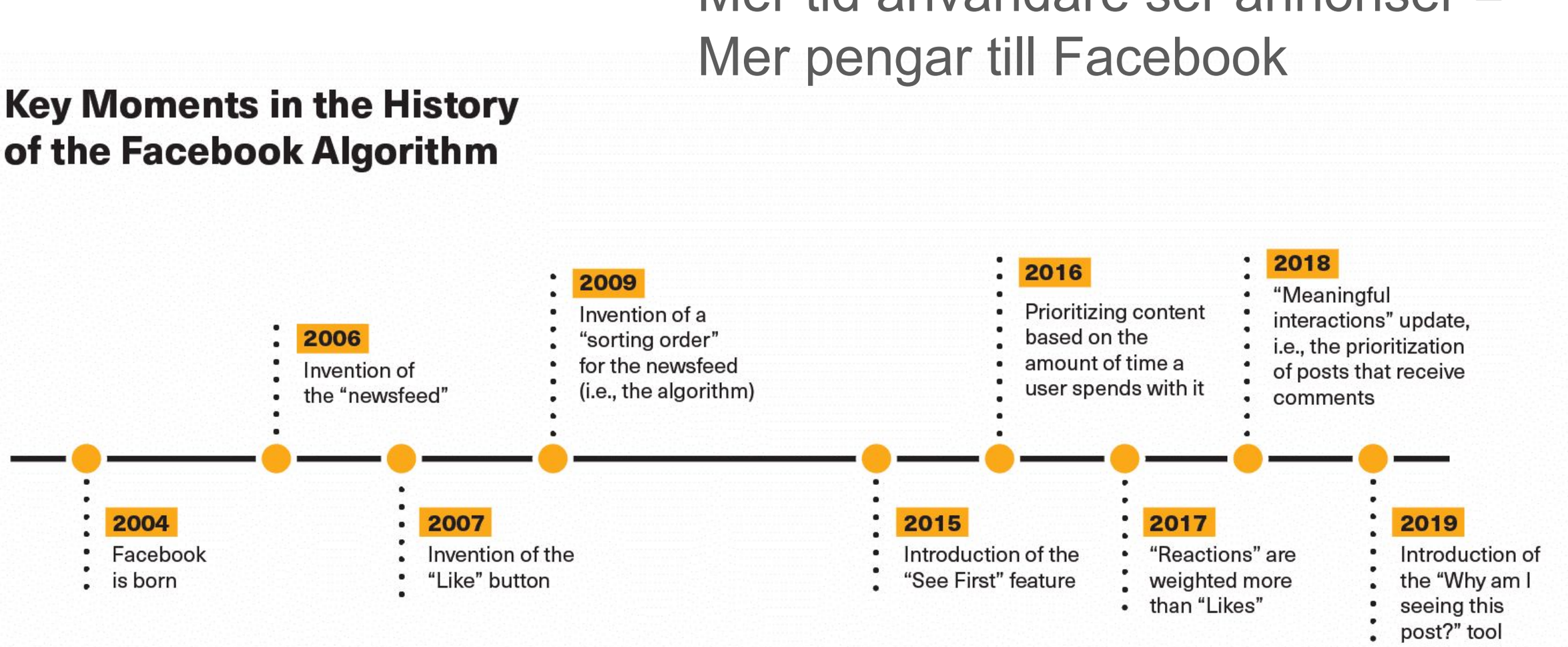
Launch date: April 24, 2012
Hazards: Spammy or irrelevant links; links with over-optimized anchor text

How it works: Google Penguin's objective is to down-rank sites whose links it deems manipulative. Since late 2016, Penguin has been part of Google's core algorithm; unlike Panda, it works in real time.

Analys av dina ord	Matcha sökningen	Rankning av användbara sidor	Returnera de bästa resultaten	Sammanhanget är viktigt

ALGORITMER: Facebook

Mer tid på Facebook =
Mer tid användare ser annonser =
Mer pengar till Facebook



Men vänta,

Vi spolar tillbaka tiden 84 år!

TURINGMASKINEN!

ALGORITMER: Turingmaskinen

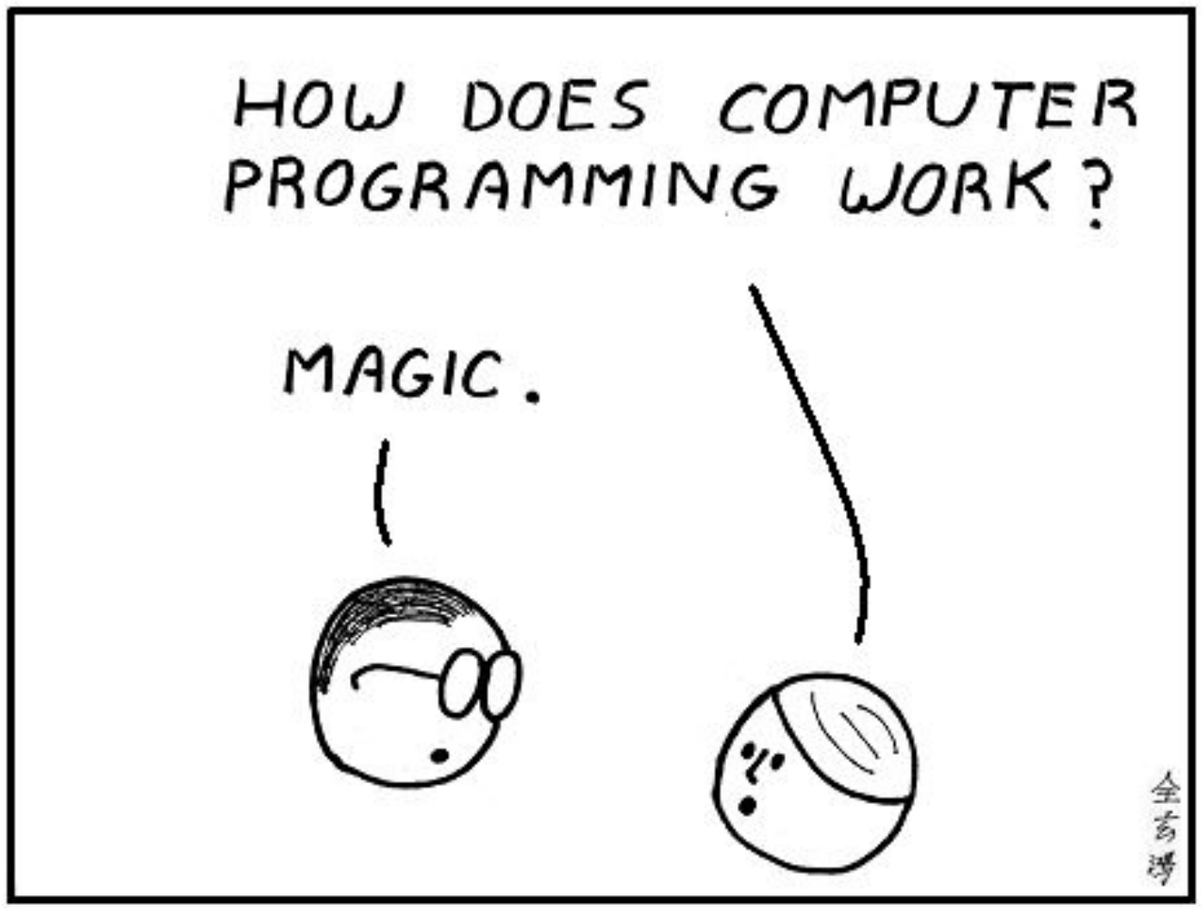
Kortfattat är en den **universella turingmaskinen** en teoretisk idé om en maskin som kan lösa vilket problem som helst.

ALGORITMER: Turingmaskinen

- Den har ett nuvarande tillstånd (**state**) som i praktiken i dagens datorer motsvarar minnet (hårddisk + RAM)
- Den har ett känt antal **instruktioner** den kan använda för att ändra sitt tillstånd.
- En dator är en universell turing maskin, vilket **program** som helst kan i teorin köras på alla andra universella turingmaskiner, förutsatt att instruktionerna skrivs om på ett tolkningsbart sätt.


<turing
machine
primer>

VAD ÄR PROGRAMMERING?



VAD ÄR PROGRAMMERING?

- Instruktioner
- Program
- Kod?

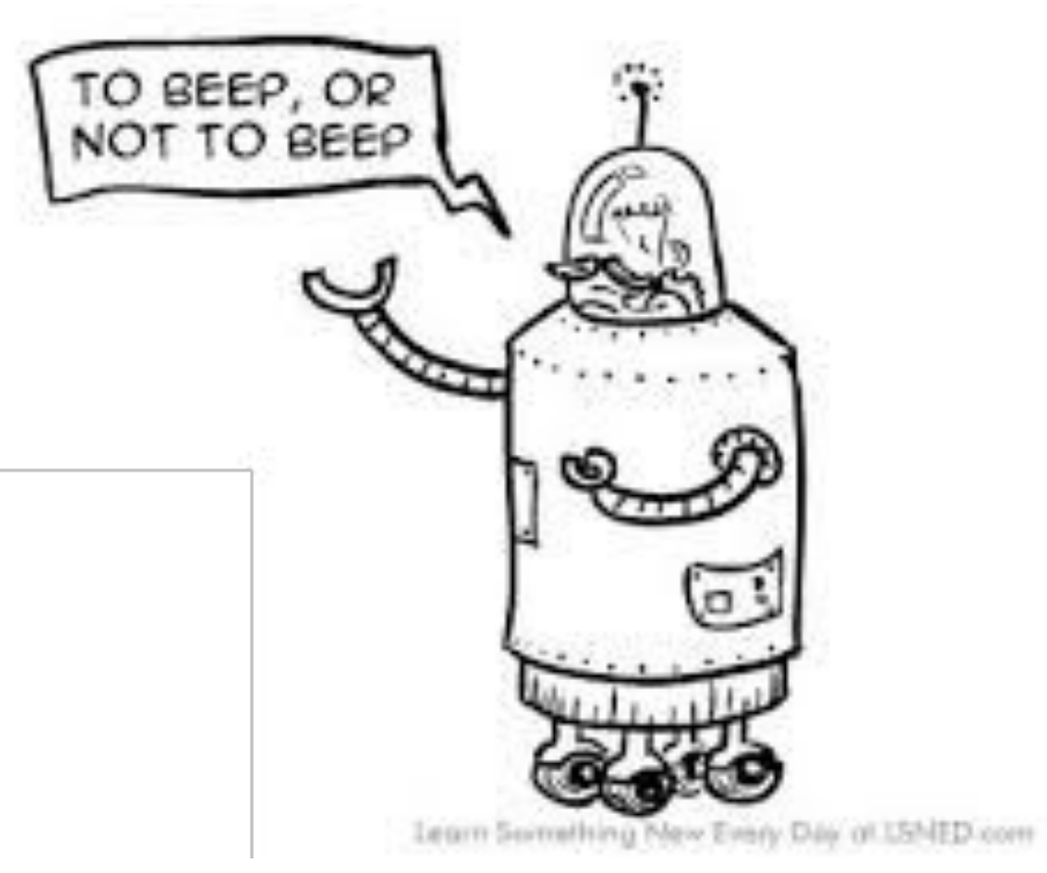
KOD

- "En överenskommelse över hur information ska överföras"
- I detta fall (programkod) mellan en programmerare och en ***parser***.

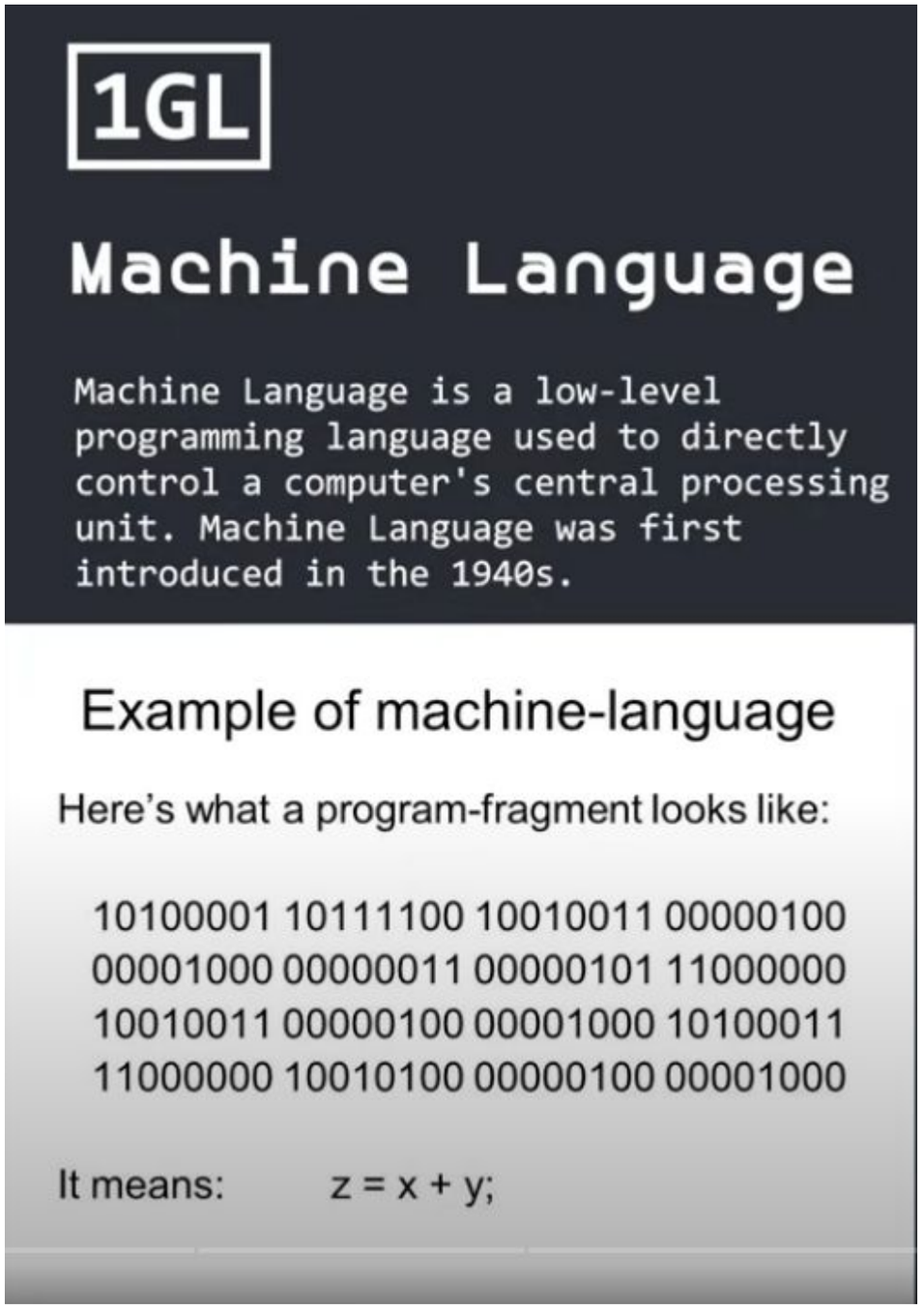
- Programkod är ***implementationer*** av en ***algoritm***.
- Samma algoritm kan implementeras i flera olika programspråk.

KOD

- Kort och gott instruktioner för vad en dator ska göra.



KORT HISTORIK

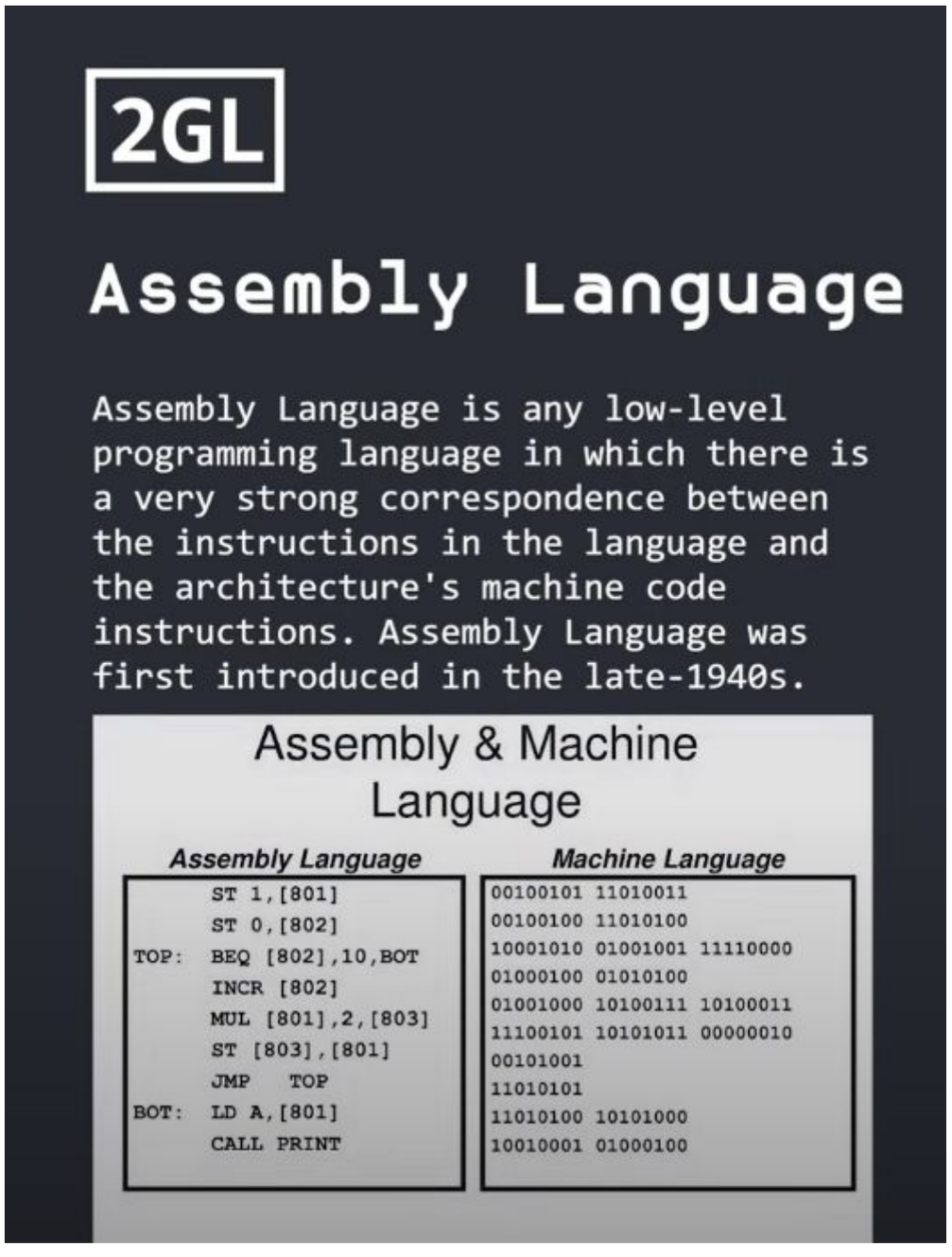


KORT HISTORIK

Sent 1940-tal – assembler

Assembler, eller assembly, var det första steget att göra programmering enklare. Istället för kod som ”0010100000001111” kan programmeraren skriva ”goto L41” som betyder att programmet ska hoppa till rad 41.

Kathleen Booth utvecklade det första assemblerspråket 1947.



KORT HISTORIK

1959 – Cobol

Grace Hopper utvecklar det plattformsoberoende språket Cobol och dess kompilator.

KOMPILATOR

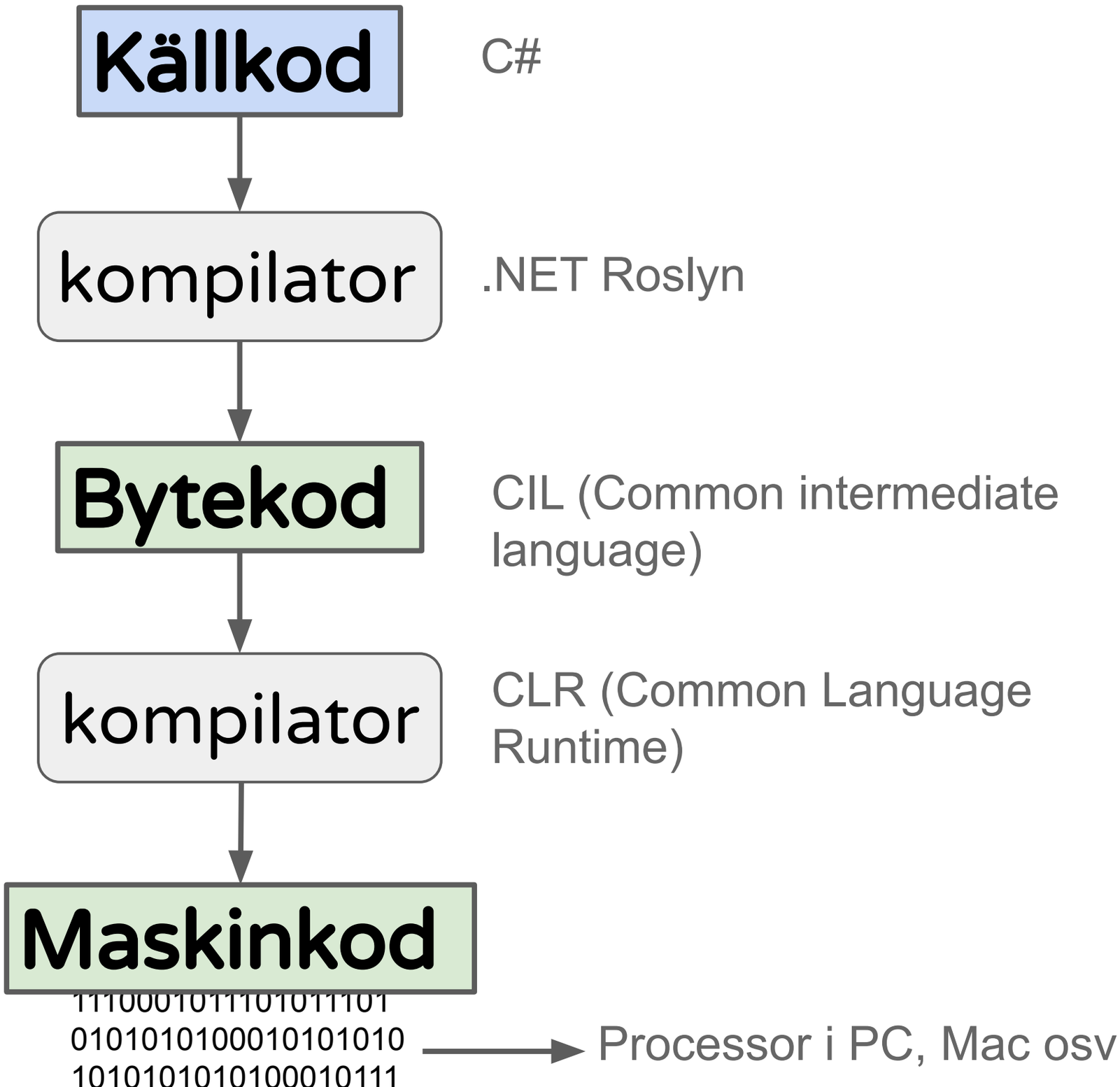
- En ***kompilator*** tolkar programkod (läsbart av människor) och gör om den till maskinkod, bytekod (som endast kan läsas av andra parsers) eller något annat språk

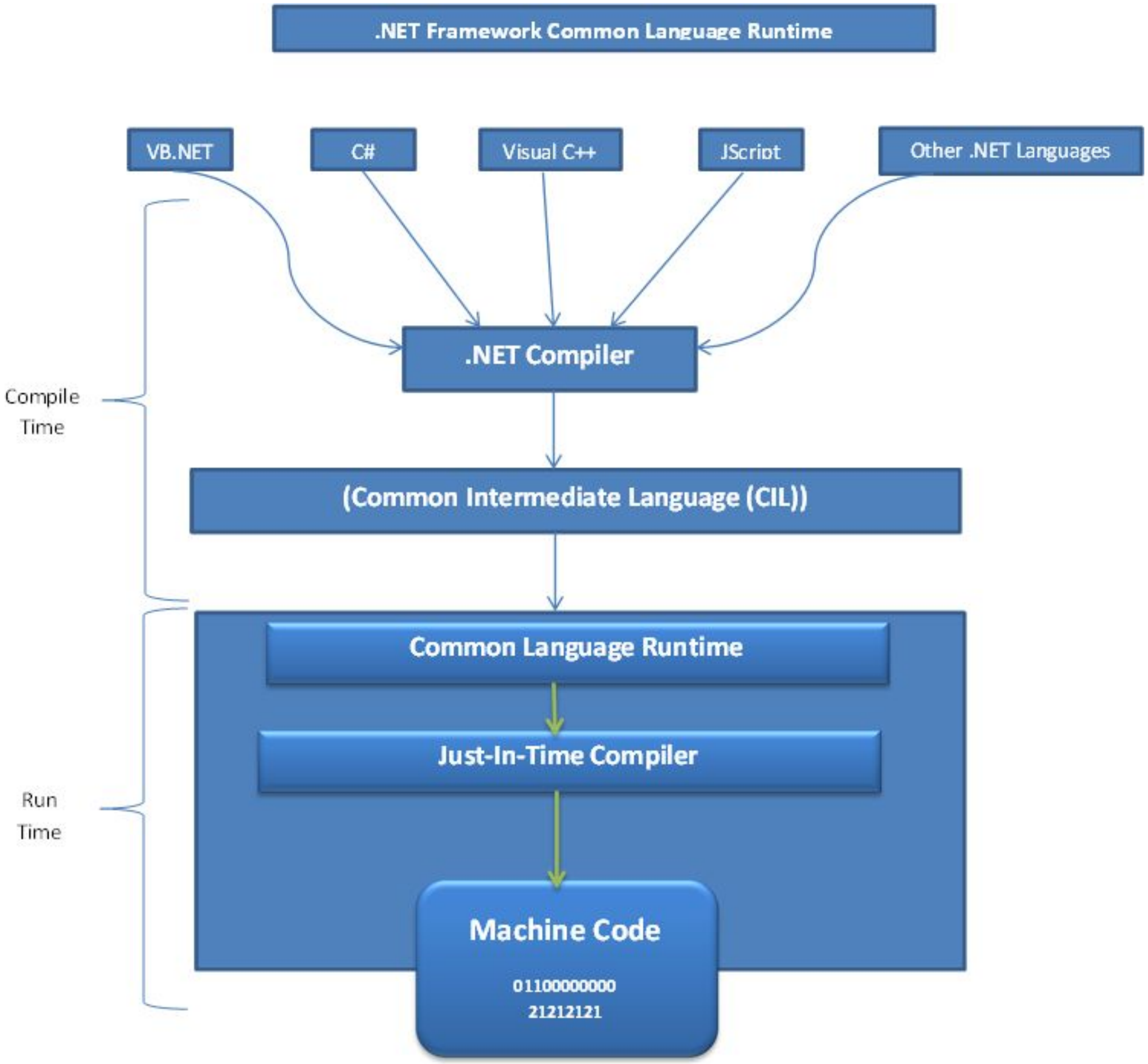
KOMPILATOR/PARSER

- a.ka. Syntaxanalyserare
- En parser är en del av kompilatorn. Den läser och tolkar den kod ni skrivit.

- Ni skriver alltså er kod till en parser!
- Kompilatorn är ofta del av ett ***framework***
  - *C# kompileras och parsas (tolkas) av .NET frameworket.*
  - *Javascript parsas av webbläsaren*
  - *HTML tolkas av en webbläsaren (OBS: inget programeringsspråk)*

Eh, va?


Källkod
C#
kompilator
.NET Roslyn
Bytekod
CIL (Common intermediate language)
kompilator
CLR (Common Language Runtime)
Maskinkod
1110001011101011101
0101010100010101010
1010101010100010111
Processor i PC, Mac osv


.NET Framework Common Language Runtime
VB.NET
C#
Visual C++
JScript
Other .NET Languages
.NET Compiler
Compile Time
(Common Intermediate Language (CIL))
Common Language Runtime
Just-In-Time Compiler
Run Time
Machine Code
01100000000
21212121

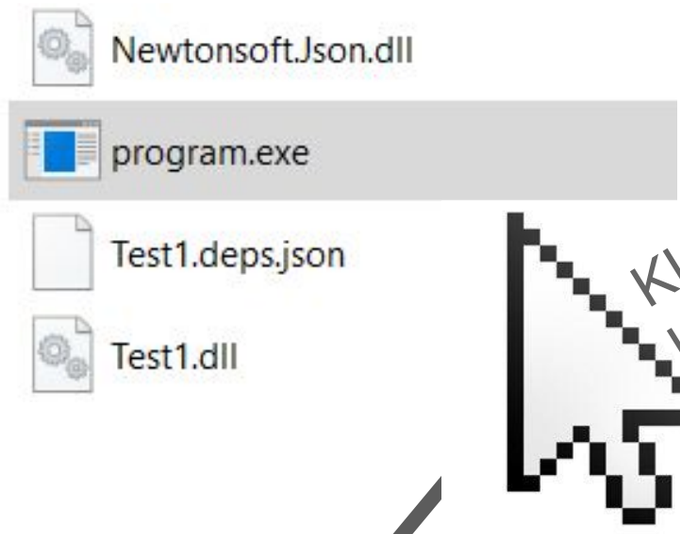
Vi tar det igen!

```
class Program
{
    static void Main(string[] args)
    {
        int k = 5;
        int i = -2;
        while (i <= k)
        {
            i = i + 2;
            k--;  // samma som:  K = K - 1;
            Console.WriteLine(i + k);
        }
```

Källkod i C#

kompileras till

CIL, **Bytekod**



CLR (Common Language Runtime)

Maskinkod

```
11100010111010111101
01010101000101010101
10101010101000101111
01011010000101010011
```

De stora skillnaderna

Källkod

Bytekod

Maskinkod

De stora skillnaderna

Källkod

Läsbart av människor,
likt mänskliga språk
Ej optimerat för snabb körning

Bytekod

Ej läsbart av människor,
Läsbart av .NETs parser
Plattformsoberoende
Delvis optimerat för snabb körning

Maskinkod

Ej läsbart av människor,
“Ettor och nollor”
Optimerat för snabb körning
Plattformsberoende

PARSER

PÅSTÅENDE:

För att bli bra på att koda är det viktigt att ha en förståelse för hur parsern läser och utför instruktionerna i er kod.

APPLIKATION

- Vad är en applikation?

APPLIKATION

- Program == applikation.

- En användbar kodsamling för att lösa ett eller flera specifika problem!

- I "vardagen" syftar en applikation oftast på en samling kod som är färdig att användas, i en given ***miljö***.

FRAMEWORK

- Framework, eller ramverk, är en kodsamling som möjliggör utvecklingen av mer komplexa applikationer
- .NET är ett framework
- Operativsystem (iOS, Android, Windows, Linux) är framework.

VAD MENAS MED MILJÖ?

- Ett program är som sagt en kodsamling för att lösa ett specifikt problem i en given ***miljö***.
- Miljön syftar till förutsättningen där programmet ska användas. dvs:
    - processor
    - kärna
    - parser
    - framework

- Framework och parser kan gemensamt kallas för: ***Runtime***
- I vissa sammanhang kallas detta för en stack, *vilket innebär en "hög" med frameworks*

SAMMANFATTNING

- **Algoritm**: En serie instruktioner för att lösa ett problem
- **Programkod**: Instruktioner för en parser

- **Applikation**: En samling körbara instruktioner för att lösa ett eller flera problem. Synonymt med program eller app.

- **Parser**: En applikation som tolkar och utför instruktioner
- **Framework**: En applikation eller samling med kod som möjliggör mer avancerade program.

- **Bytekod**: Instruktioner som enbart tolkas av ett annat program/framework
- **Maskinkod**: Instruktioner för en processor

SAMMANFATTNING

- Vad är C#?
- Vad är .NET?

- Problem: Vad är skillnaden på en algoritm och ett program?

INTRODUKTION II

Pseudokod och flödesdiagram

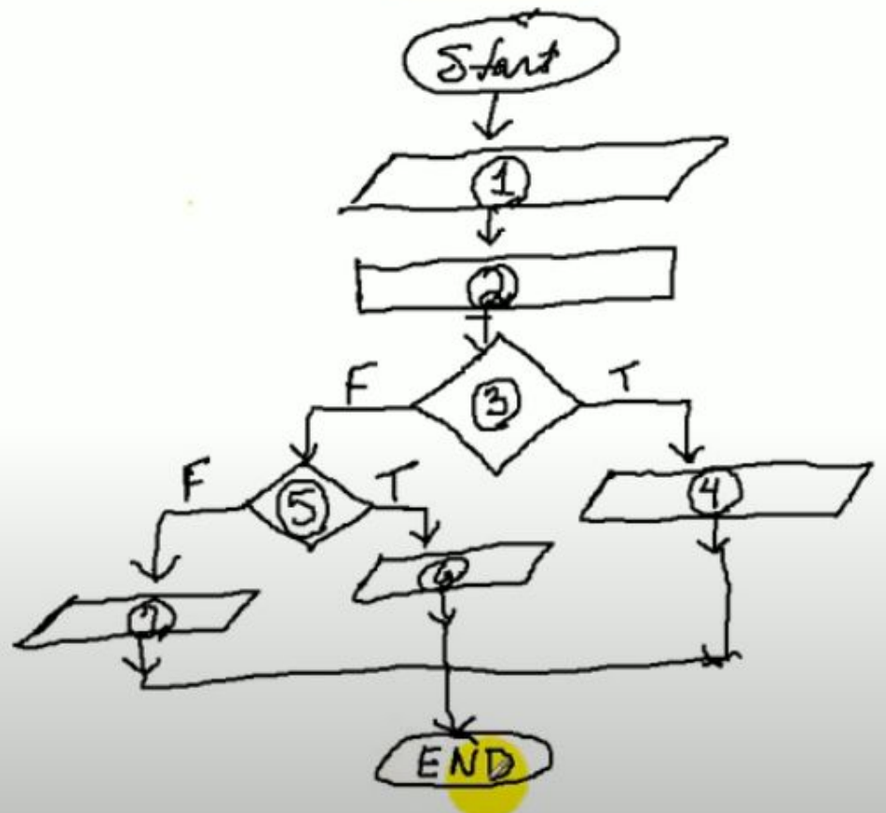
PSEUDOKOD

Create a pseudocode algorithm and flowchart for a solution to find the smallest of two numbers.

Pseudocode

1. prompt user for two integer values
2. save input to num1 and num2
3. check to see if num1 == num2
    4. if it is: print "num1 and num2 are equal"
5. if it is not: check to see if num1 is < num2
    6. if it is: print "num 1 is the smallest "
    7. if it is not: print "num2 is the smallest"

Flowchart



PSEUDOKOD

- Kan vara nästan vad som helst!
- Beskrivande kod men som inte är redo att köras i en parser.
- Oftast ganska nära riktig kod.

- Wikipedia: "*ett för ett icke-programspråk specifikt sätt att beskriva algoritmer*"

PSEUDOKOD

- Kan användas för att designa och tänka/samarbeta kring kod i ett tidigt skede.
- Eller för att tolka och förstå kod som redan finns.

PSEUDOKOD

💡 <u>Bordslampan funkar inte:</u>

Är lampan inkopplad till strömmen?

Nej: Koppla in lampan.

Ja: Är glödlampan trasig?

Ja: Byt glödlampan.

Nej: Laga eller byt hela bordslampan

PSEUDOKOD: C-Style

```
void function fizzbuzz
{
  for (i = 1; i <= 100; i++)
  {
    set print_number to true;
    If i is divisible by 3
    {
      print "Fizz";
      set print_number to false;
    }
    If i is divisible by 5
    {
      print "Buzz";
      set print_number to false;
    }
    If print_number, print i;
    print a newline;
  }
}
```

PSEUDOKOD: Mellanting

IF the hedges are unkempt
    Trim the hedges

LOOP!

WHILE there are toys on the lawn
        Remove one toy

IF the grass is long
        Mow the grass

The Structure Theorem

Det är möjligt att representera vilken algoritm som helst med bara tre grundläggande kontrollstrukturer:

- Sequence
- Repetition (WHILE)
- Selection (IF THEN ELSE)

https://en.wikipedia.org/wiki/Structured_program_theorem

Sequence

Kör frammåt

Bromsa

Kör åt vänster

Bromsa

Kör åt vänster

Bromsa

Kör åt vänster

Kör frammåt

Bromsa

Kör åt vänster

Bromsa

Kör åt vänster

Bromsa

Kör åt vänster

Kör frammåt

Bromsa

Kör åt vänster

Bromsa

Kör åt vänster

Bromsa

Kör åt vänster

Repetition

WHILE AntalVarv är mindre än 4

- Kör frammåt
- Bromsa
- Kör åt vänster

Bromsa

Kör åt vänster

Bromsa

Kör åt vänster

Lägg till **ett** till AntalVarv

Selection

IF Batterinivå är mer än 25%

WHILE AntalVarv är mindre än 4

Kör frammåt

Bromsa

Kör åt vänster

Bromsa

Kör åt vänster

Bromsa

Kör åt vänster

Lägg till **ett** till AntalVarv

ELSE Åk till laddstation

ÖVNING: Pseudokod

1. Göra en kopp te

2. Ett program som låter användaren skriva in en mening. Programmet skriver sedan ut meningen men med slumpvis ordföljd.

3. Logga in på en hemsida med användarnamn och lösenord (eller med med BankID)

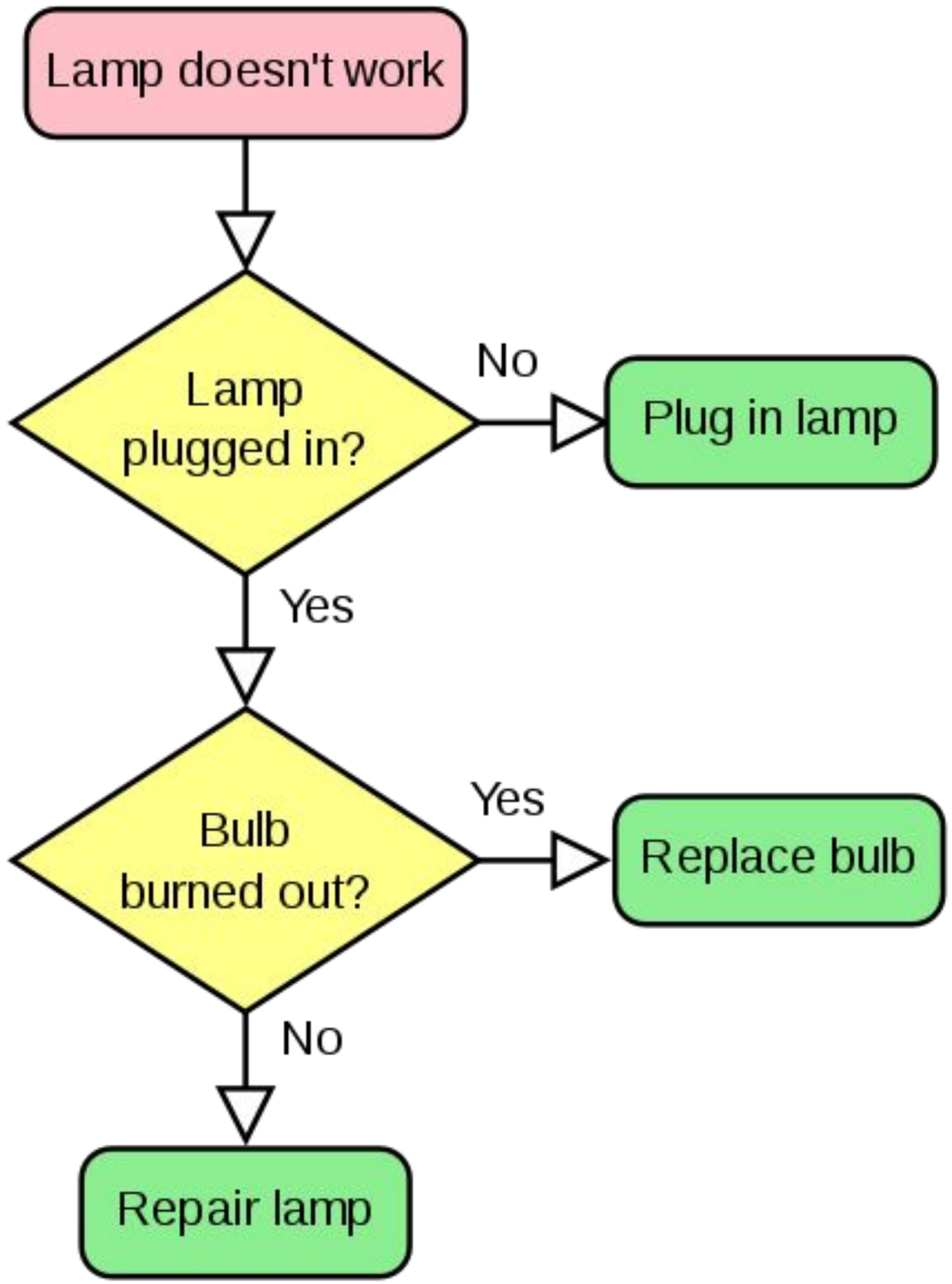
Börja simpelt med få steg, se sedan om fler steg behövs. Vad är rimligt att ta med och inte?

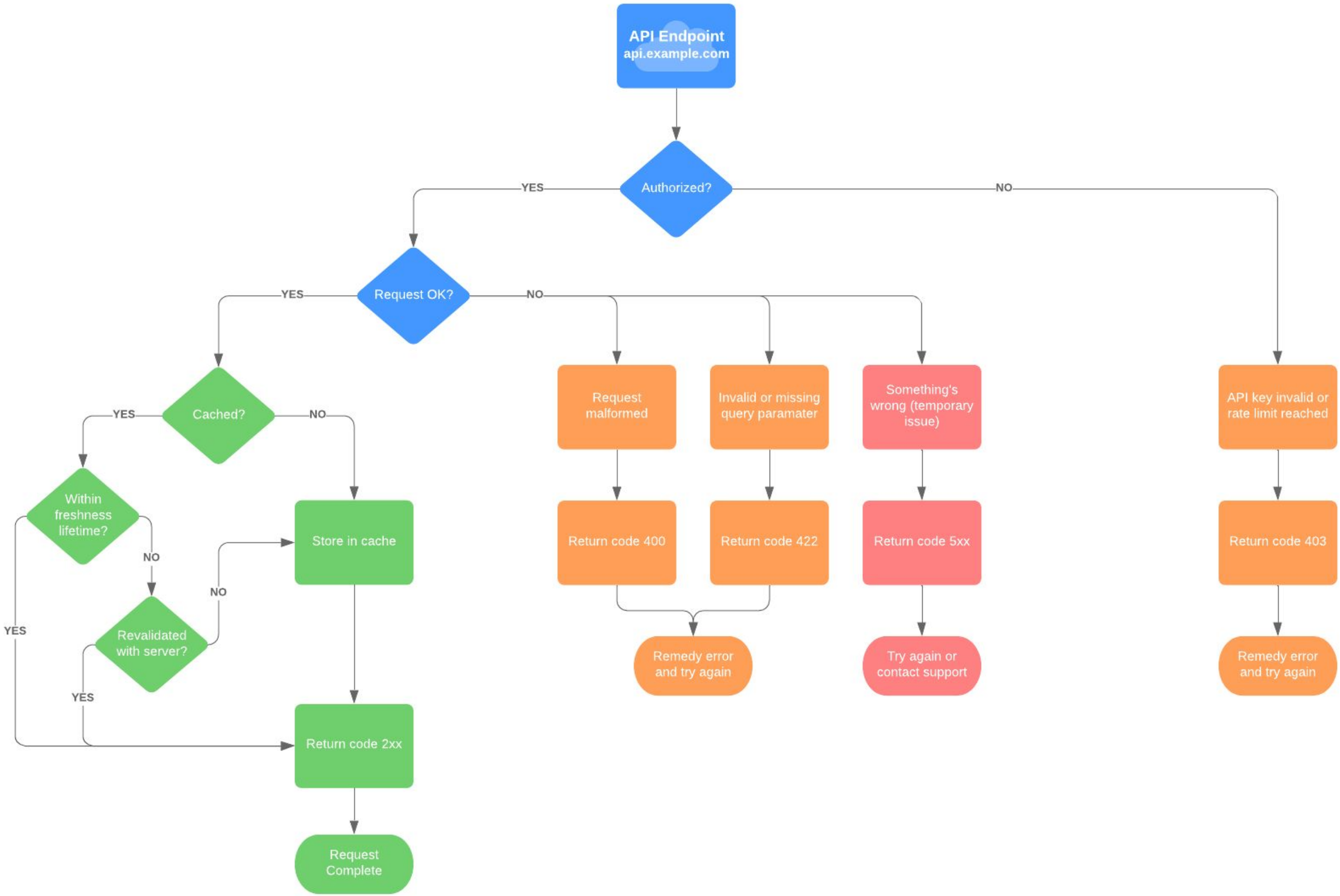
FLÖDESDIAGRAM

FLÖDESDIAGRAM (flowchart)

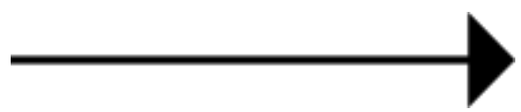
- Används till att *visualisera en förändring* mellan två eller flera tillstånd, exempelvis:
  - En arbetsprocess
  - En lösning på ett problem
  - En programåtgärd

- Används i praktiken för **dokumentation** och som en typ av visuell **pseudokod**


Lamp doesn't work
Lamp plugged in?
No
Plug in lamp
Yes
Bulb burned out?
Yes
Replace bulb
No
Repair lamp


API Endpoint
api.example.com
Authorized?
YES
NO
Request OK?
YES
NO
Request malformed
Invalid or missing query paramater
Something's wrong (temporary issue)
API key invalid or rate limit reached
Cached?
YES
NO
Within freshness lifetime?
NO
YES
Revalidated with server?
NO
YES
Store in cache
Return code 2xx
Request Complete
Return code 400
Return code 422
Return code 5xx
Return code 403
Remedy error and try again
Try again or contact support
Remedy error and try again

FLÖDESDIAGRAM - ISO 5807



Övergång

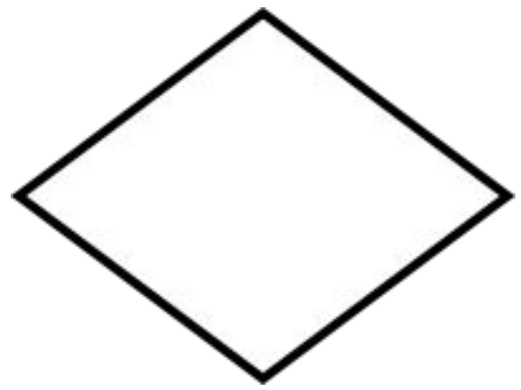
FLÖDESDIAGRAM - ISO 5807

Start / slut på flödet

FLÖDESDIAGRAM - ISO 5807

Tillstånd

FLÖDESDIAGRAM - ISO 5807



Beslut

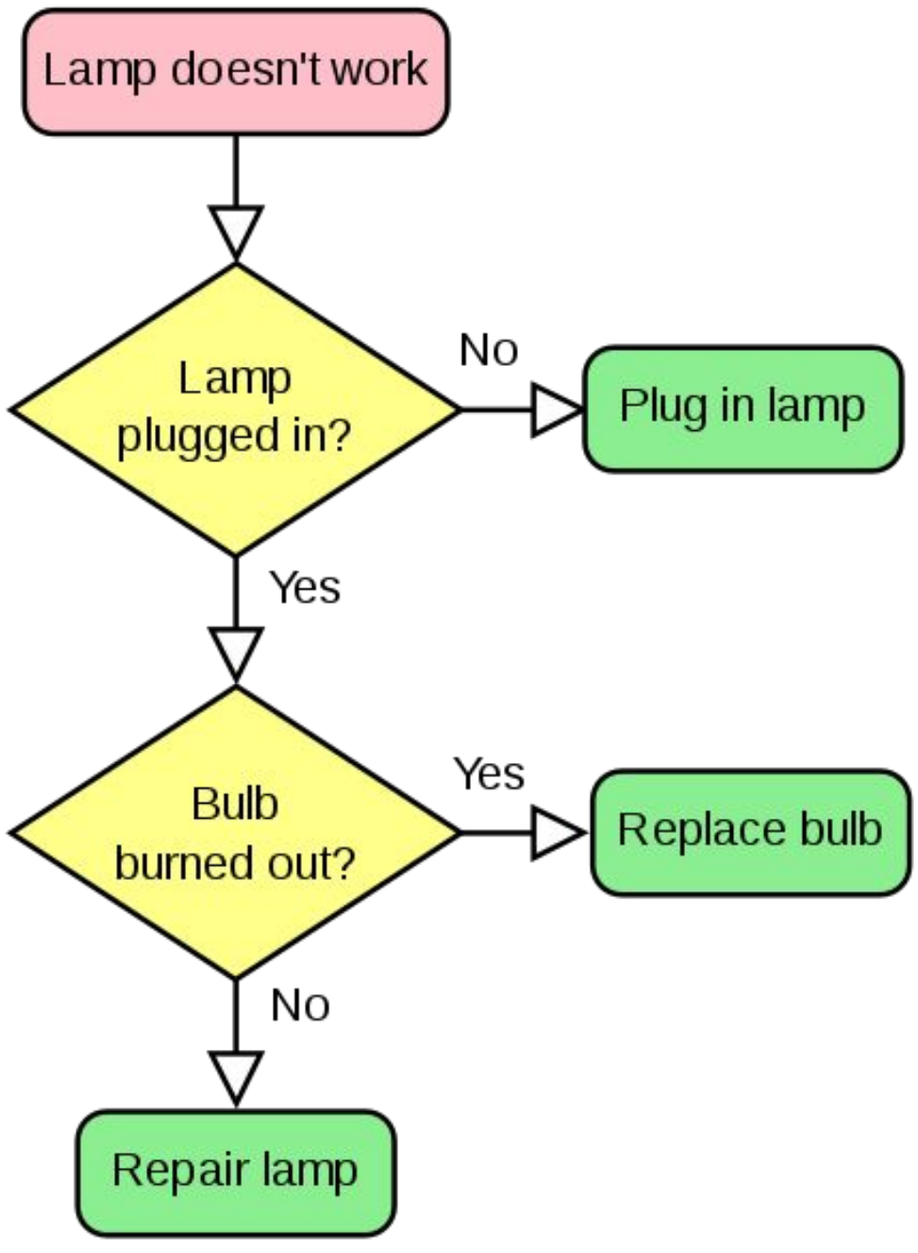
Innehåller en fråga och flera alternativa övergångar

FLÖDESDIAGRAM - ISO 5807

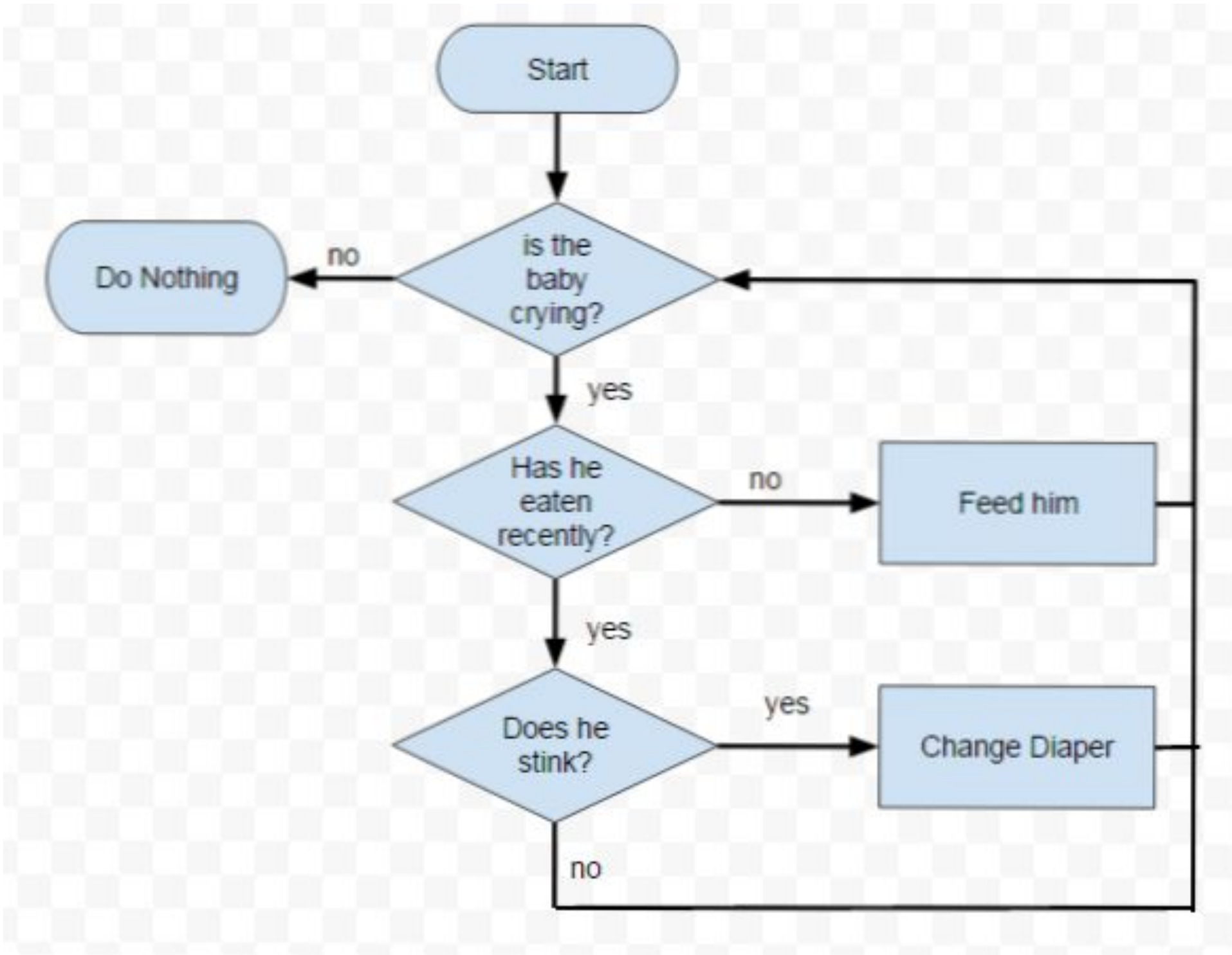
Inmatning

Innebär att ytterligare information behöver införas

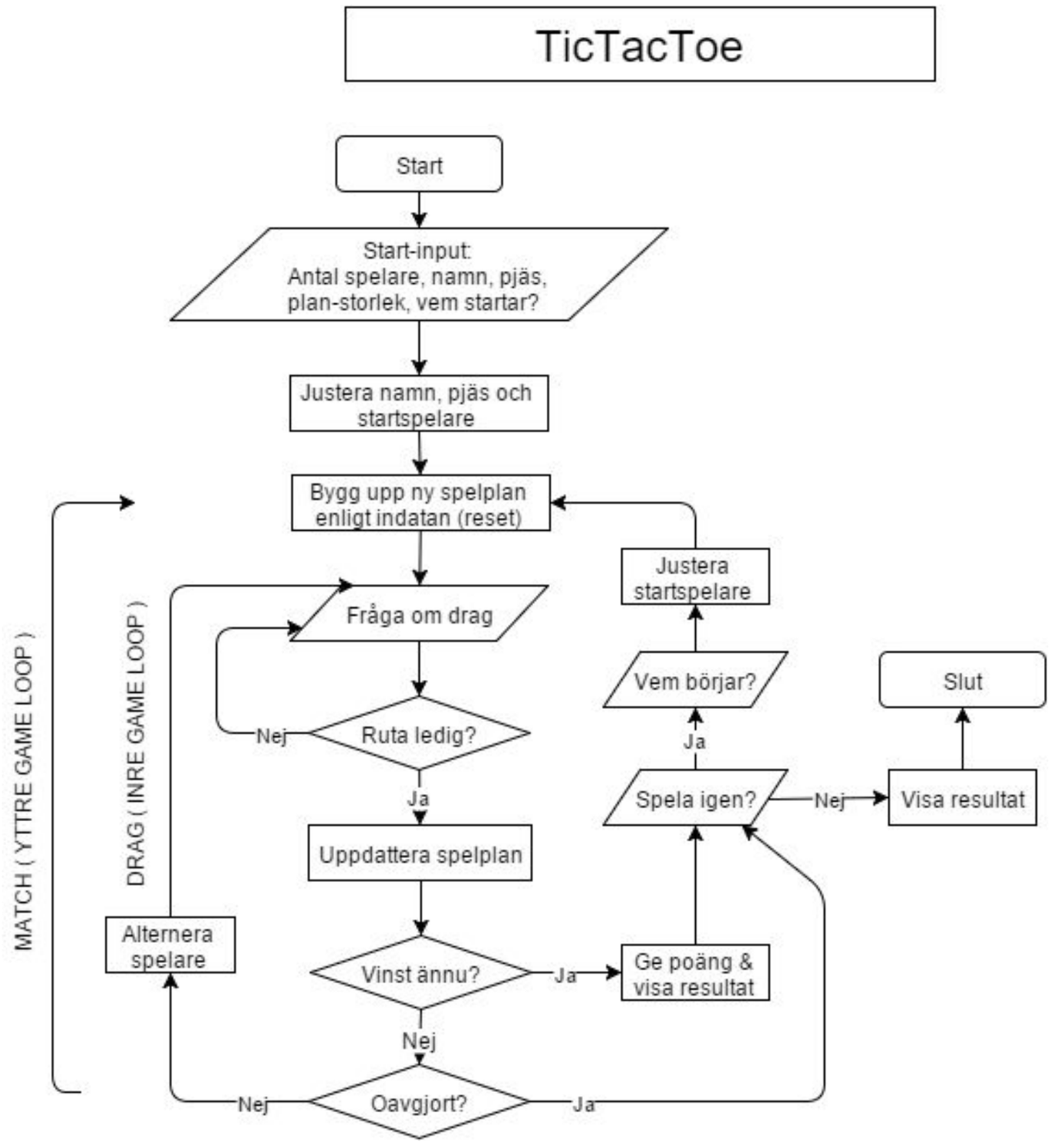
EXEMPEL


Lamp doesn't work
Lamp plugged in?
No
Plug in lamp
Yes
Bulb burned out?
Yes
Replace bulb
No
Repair lamp

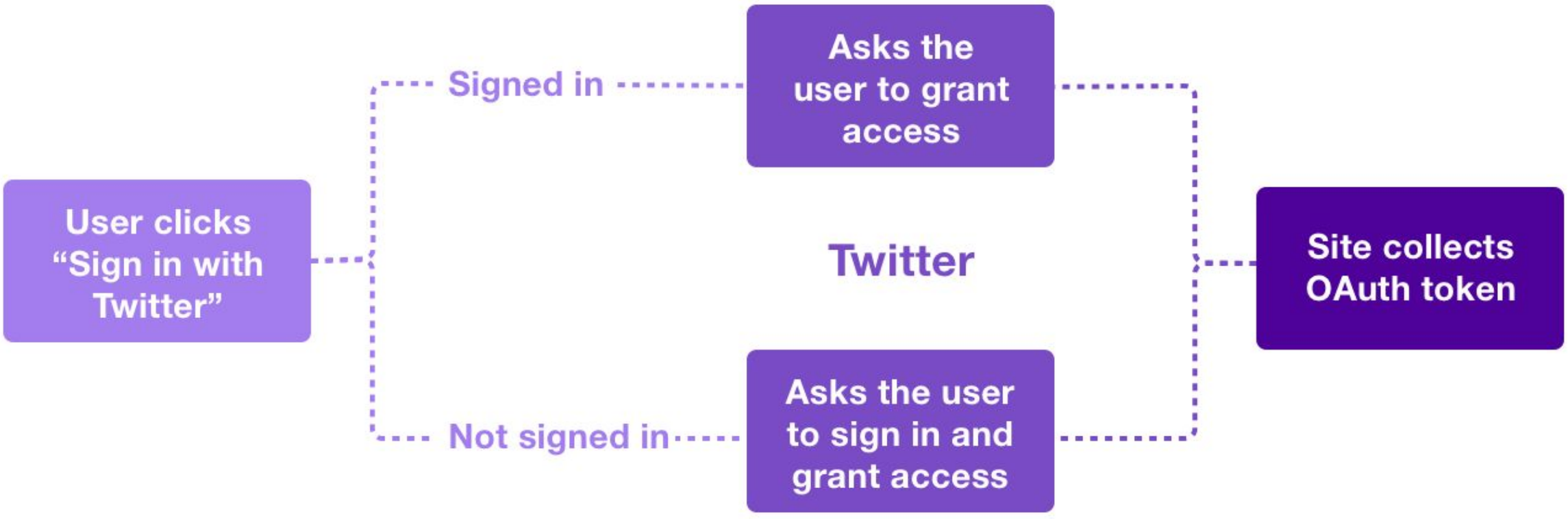
EXEMPEL


Start
is the baby crying?
no
Do Nothing
yes
Has he eaten recently?
no
Feed him
yes
Does he stink?
yes
Change Diaper
no

EXEMPEL


TicTacToe
Start
Start-input:
Antal spelare, namn, pjäs,
plan-storlek, vem startar?
Justera namn, pjäs och
startspelare
Bygg upp ny spelplan
enligt indatan (reset)
MATCH (YTTRE GAME LOOP)
DRAG (INRE GAME LOOP)
Fråga om drag
Nej
Ruta ledig?
Ja
Uppdattera spelplan
Alternera
spelare
Vinst ännu?
Ja
Ge poäng &
visa resultat
Nej
Nej
Oavgjort?
Ja
Justera
startspelare
Vem börjar?
Ja
Spela igen?
Nej
Visa resultat
Slut

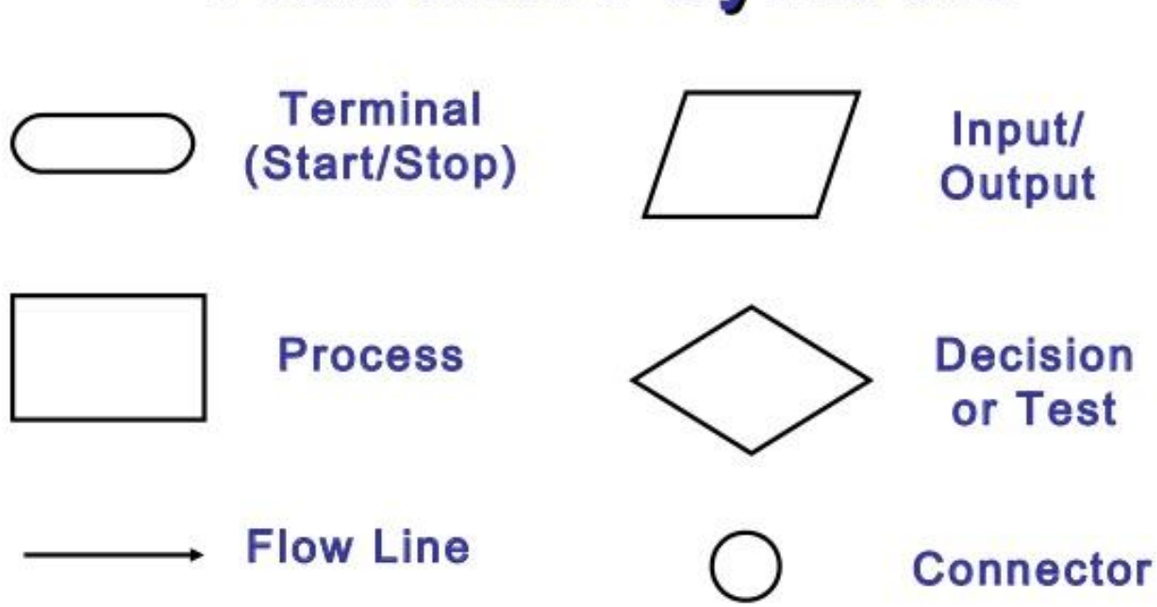
Alla följer inte standard...



ÖVNING

Rita ett flödesdiagram för följande problem

- **Koka en kopp té**
- **Sortera tvätt**
- **Bubble Sort**



www.draw.io

eller google drawing

FRÅGOR

Vad säger ett flödesdiagram oss?

Vad har vi för avgränsningar? Vad har vi för nivå?

Vad innebär att iterera?

ALGORITMER: Komplexitet

ALGORITMER: Komplexitet

- Matematiksbetydelse: Antalet beräkningar som behöver genomföras
- Anges i stort sett i alla relevanta fall med Ordo-notation (Big O Notation)

- Det som är relevant gällande komplexitet är hur en algoritm växer i takt med sin datamängd

- Man utgår alltid från det värsta tänkbara fallet när man räknar ut komplexiten.

ALGORITMER: Big O Notation

- **O(n)**
  I detta enkla exempel växer algoritmens komplexitet i samma takt som att den inmatade data *n* ökar (Konstant eller Linjär ökning)

  Exempel: Hitta värde i en lista.

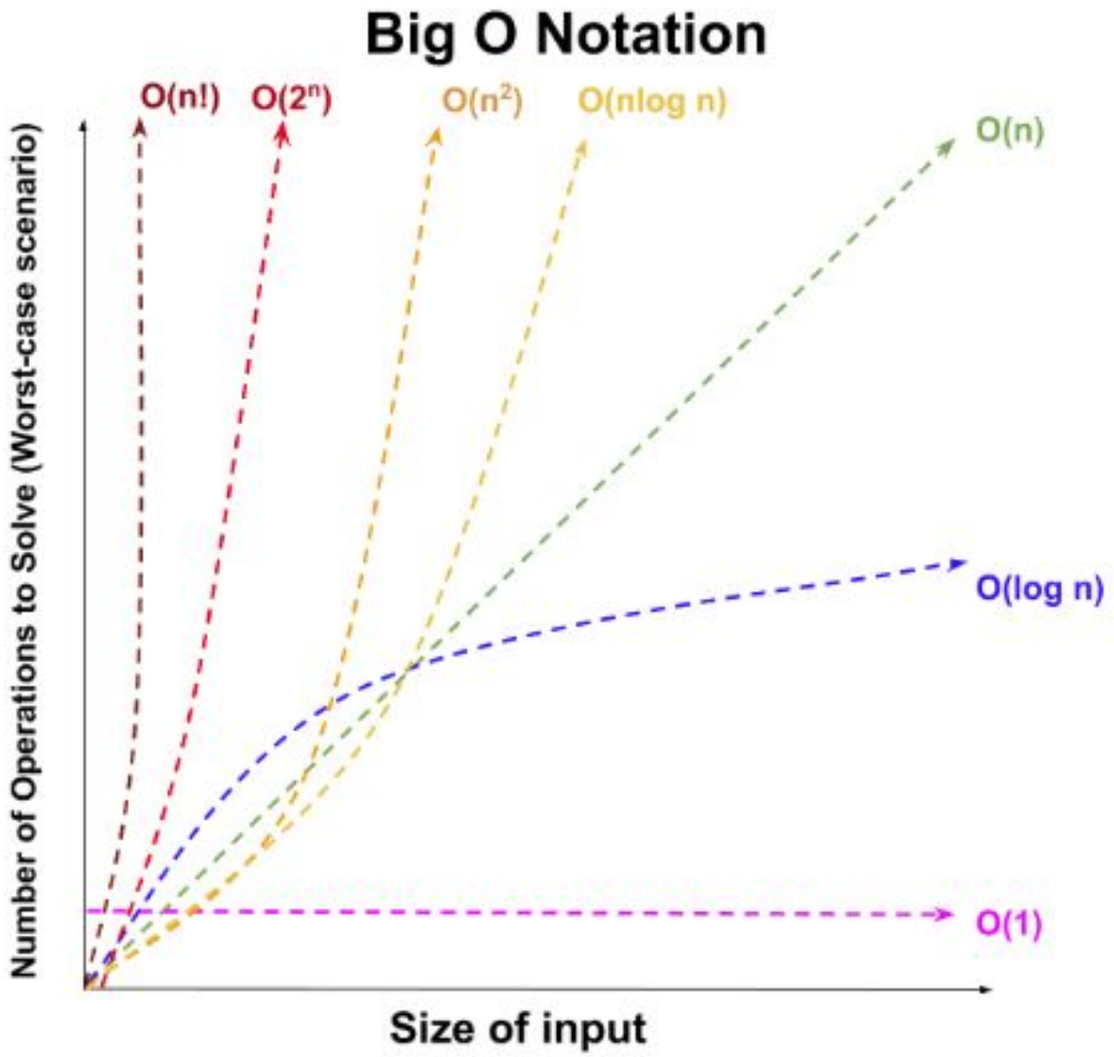
- **$O(N^2)$**
  För varje inmatning växer komplexiteten med två

  Exempel: Nästlade iterationer

- **$O(2^N)$**
  Komplexiteten ökar exponentiellt.

  Exempel: *Rekursiva* funktioner

ALGORITMER: Big O Notation



ALGORITMER: Olika komplexiteter

Comparison sorts

Name	Best	Average	Worst	Memory	Stable	Method
Block sort	n	$n \log n$	$n \log n$	1	Yes	Insertion & Merging
Bubble sort	n	n^2	n^2	1	Yes	Exchanging
Cocktail shaker sort	n	n^2	n^2	1	Yes	Exchanging
Comb sort	$n \log n$	n^2	n^2	1	No	Exchanging
Cubesort	n	$n \log n$	$n \log n$	n	Yes	Insertion
Cycle sort	n^2	n^2	n^2	1	No	Insertion
Franceschini's method[14]	—	$n \log n$	$n \log n$	1	Yes	?
Gnome sort	n	n^2	n^2	1	Yes	Exchanging
Heapsort	$n \log n$	$n \log n$	$n \log n$	1	No	Selection
In-place merge sort	—	—	$n \log^2 n$	1	Yes	Merging
Insertion sort	n	n^2	n^2	1	Yes	Insertion
Introsort	$n \log n$	$n \log n$	$n \log n$	$\log n$	No	Partitioning & Selection
Library sort	n	$n \log n$	n^2	n	Yes	Insertion
Merge sort	$n \log n$	$n \log n$	$n \log n$	n	Yes	Merging
Odd–even sort	n	n^2	n^2	1	Yes	Exchanging
Patience sorting	n	—	$n \log n$	n	No	Insertion & Selection
Quadsort	n	$n \log n$	$n \log n$	n	Yes	Merging
Quicksort	$n \log n$	$n \log n$	n^2	$\log n$	No	Partitioning
Selection sort	n^2	n^2	n^2	1	No	Selection

UPPGIFT

Använd **någon form av pseudokod** för
att förklara hur en vanlig morgonrutin kan se ut för dig,
från att du vaknar till att du befinner dig här.

Vilka steg går programmet igenom? Vilka kontrollstrukturer kan användas?

Försök att vara så tydlig som möjligt, men du behöver inte nödvändigtvis ta med exakt alla moment om det blir väldigt långt.

Extra fråga: Har du någon idé du skulle vilja testa eller förverkliga med hjälp av programmering?

Nu kan vi i någon form tänka och skriva i kod, och vi har även pratat övergripande om hur kod används.

Nästa lektion börjar vi med programmering på riktigt!

TAKOHEJ!