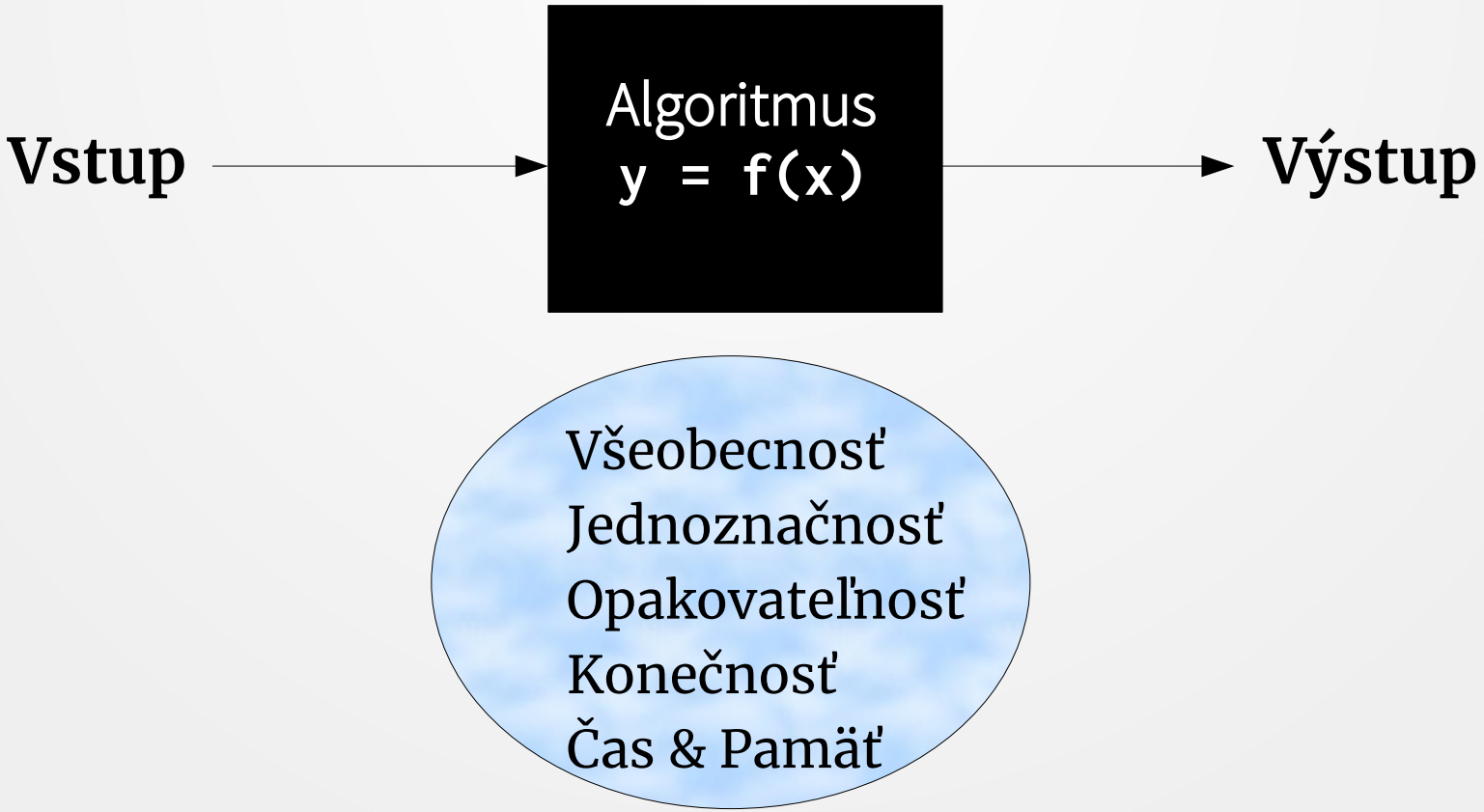
Algoritmické myslenie

Miroslav Hájek

Algoritmus

- Presný postup krokov pri riešení problému – pravidlá
- Tvorí nové z existujúceho



Riešenie problémov

1. **Porozumenie**

   „If you can't explain something in simple terms, you don't understand it."

   —*Richard Feynman*

2. **Plán**

   Ako z X vytvorím Y?

3. **Rozdeľuj a panuj**

   Pospájanie podproblémov

4. **Sebareflexia**

   Overenie fungovania v rôznych situáciach.

Prvopočiatky postupov

Antické Grécko (~300 p.n.l.)

- **Eratostenovo sito**
  - hľadanie všetkých prvočísel menších ako nejaké číslo

1.) Vytvor zoznam po sebe idúcich čísel 2 až n: (2, 3, 4, 5, ... , n)
2.) Prvé ešte nevyčiarknuté číslo nechaj a vyčiarkni všetky jeho násobky.
3.) Ak existuje, posuň sa na prvé nevyčiarknuté číslo, potom opakuj krok 2.
4.) Ak sa nedá už nič vyčiarknúť, zoznam obsahuje len prvočísla. Koniec.

- **Euklidov algoritmus**
  - výpočet najväčšieho spoločného deliteľa (NSD) dvoch čísel
  - NSD(a, b) je NSD(b, a mod b), kým b > 0

Dopad algebry - „Algorism“

- Muḥammad ibn Mūsā al-Khwārizmī (Perzia, 820 n.l.)
- ***Počítanie s hindsko – arabskou číselnou sústavou*** za použitia pravidiel
- **Výpočty koreňov rovníc** metódami doplňovania a vyvažovania

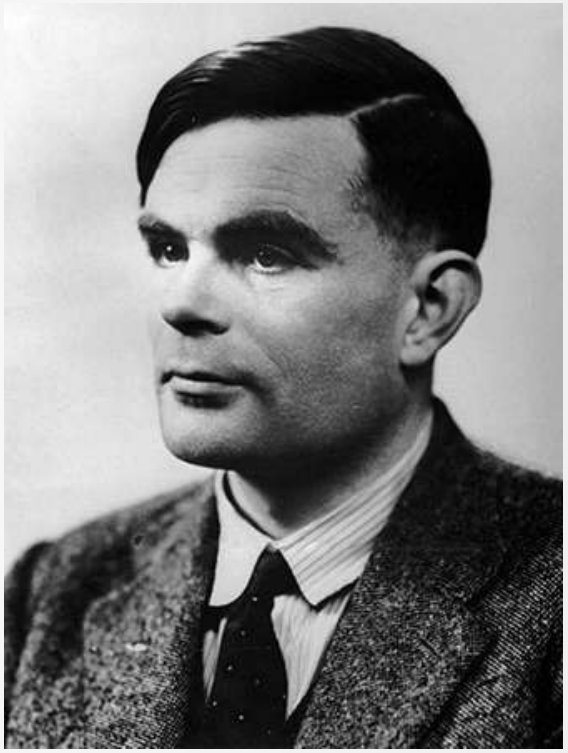
$$x^2 + 14 = x + 5$$
$$x^2 + 9 = x$$

Logika ako základ matematiky

- **David Hilbert** – *Entscheidungsproblem (1928) – rozhodovací problém*
- Algoritmus, ktorý rozhodne, či je dané tvrdenie dokázateľné z axiómov podľa pravidiel logiky.

- Dokázať, že všetky pravdivé matematické tvrdenia môžu byť dokázané – **úplnosť matematiky**
- Dokázať, že len pravdivé matematické tvrdenia môžu byť dokázané – **konzistenciu matematiky**
- Dokázať existenciu rozhodovacieho postupu na rozhodnutie o pravde alebo nepravde akejkoľvek matematickej teórie – **rozhodnosť matematiky**

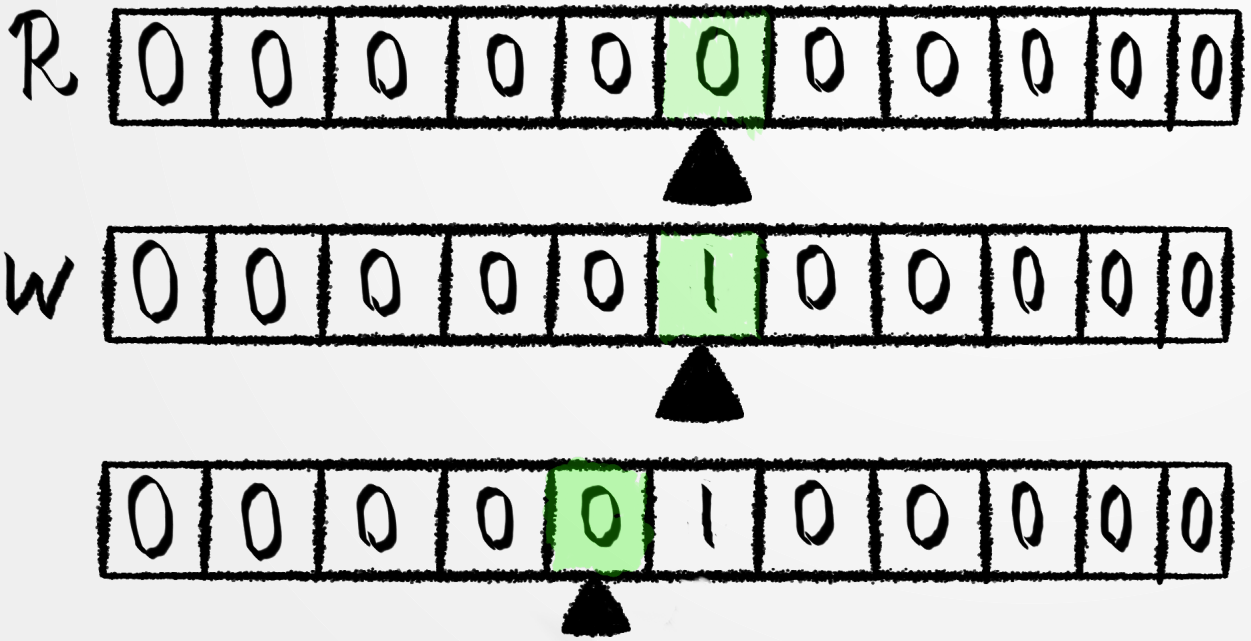
Úsvit výpočtovej techniky

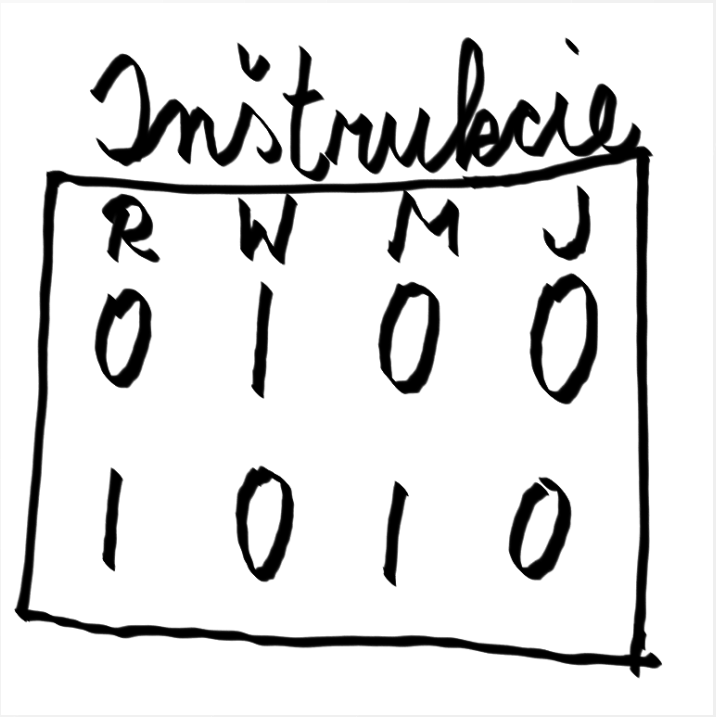


- **Kurt Gödel** - Gödelove vety o neúplnosti logických systémov
- **Alonzo Church** – λ-kalkul
- **Alan Turing** – univerzálny Turingov stroj (1936)
- O efektívnej vypočítateľnosti a vyvrátení rozhodovacieho problému

Turingov stroj

- Teoretický abstraktný model fungovania všetkých počítačov
- [Páska, Hlava] - [Stav, Tabuľka inštrukcií], Abeceda
- **3 - stavový Busy Beaver** – Čítaj, Zapíš, Posuň, Ďalší stav

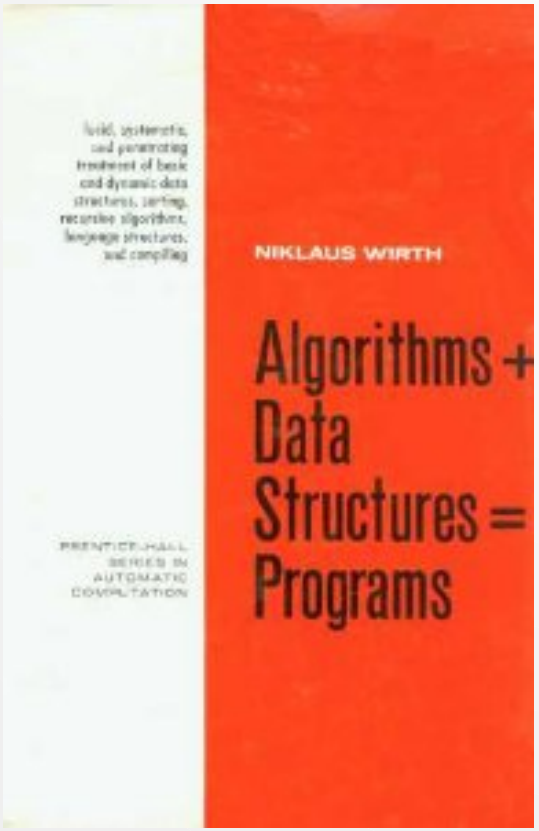


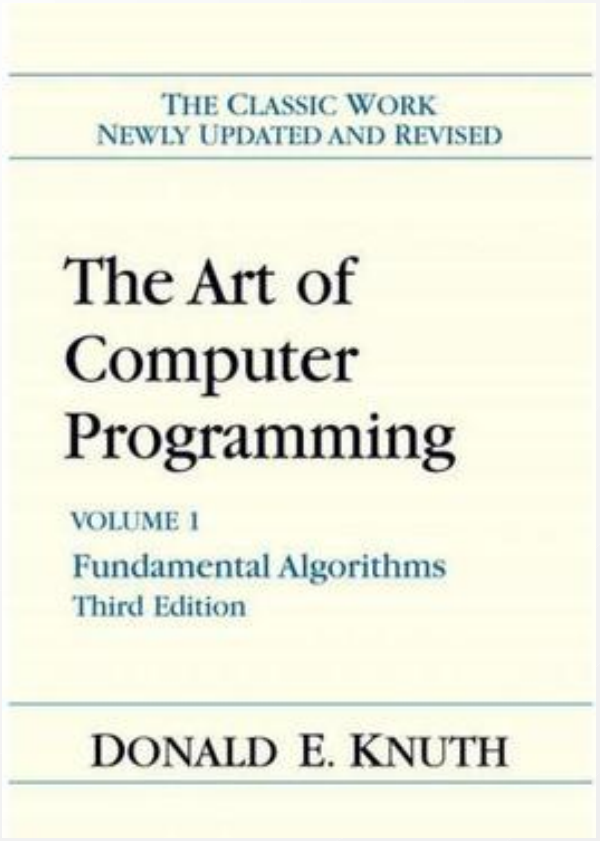


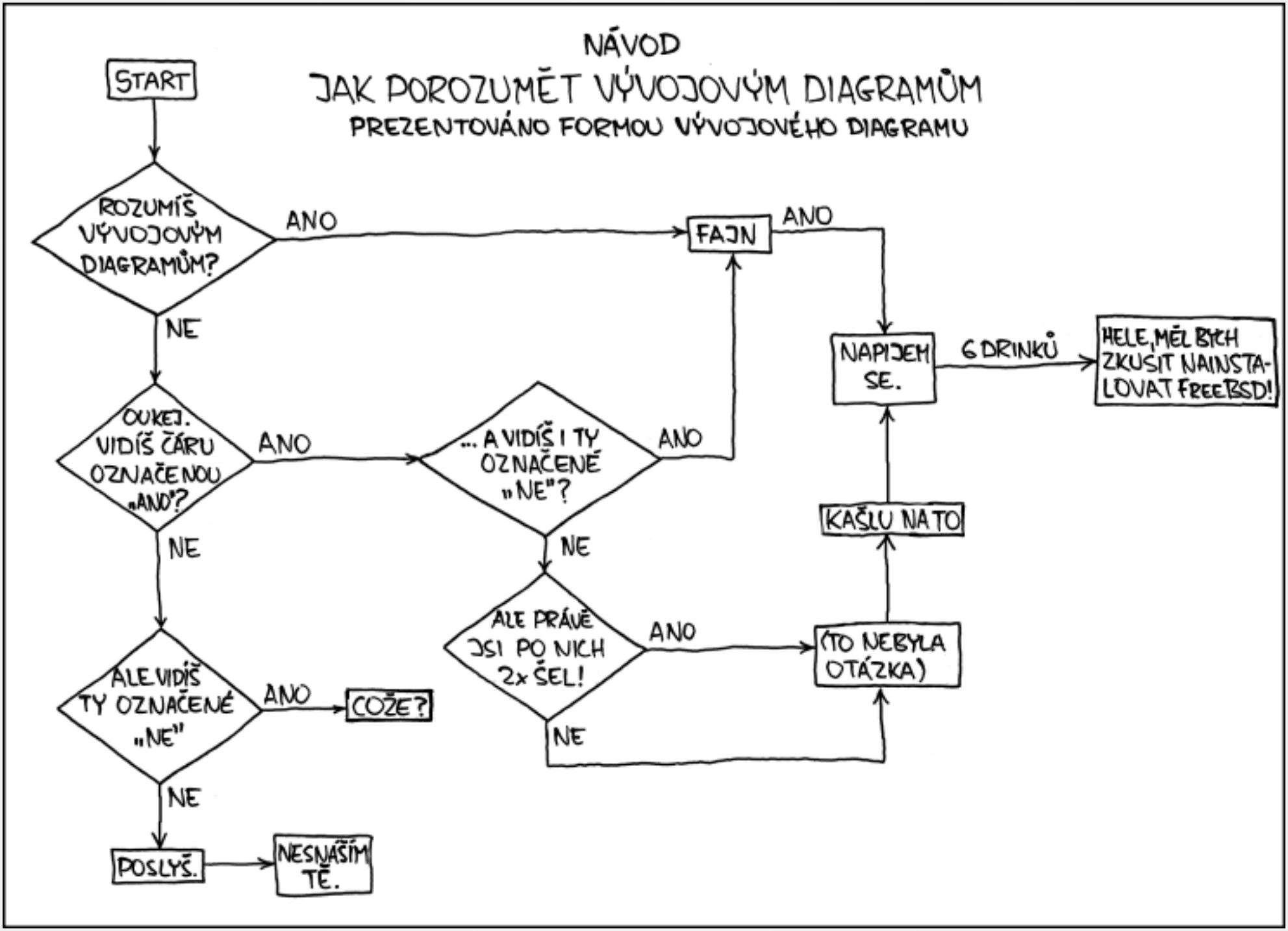
https://turingmachinesimulator.com/

Dáta & Algoritmy

- Neoddeliteľná dvojica – jedno závisí a ovplyvňuje druhé
- **Kuchyňa:** ingrediencie *(dáta)*, recept *(algoritmus)*
- **Majstrovanie:** materiál *(dáta)*, návod *(algoritmus)*

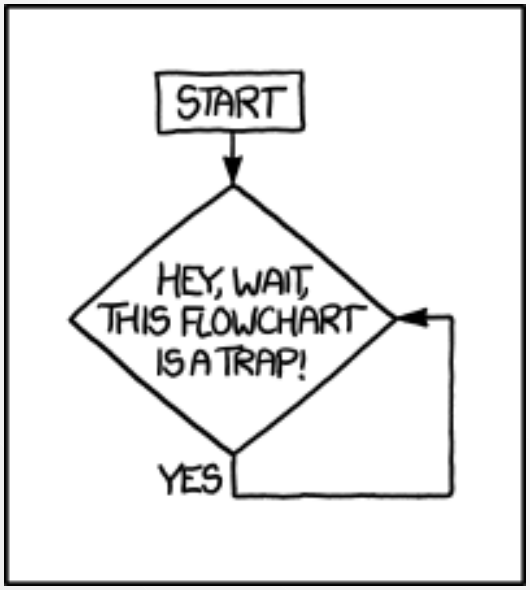


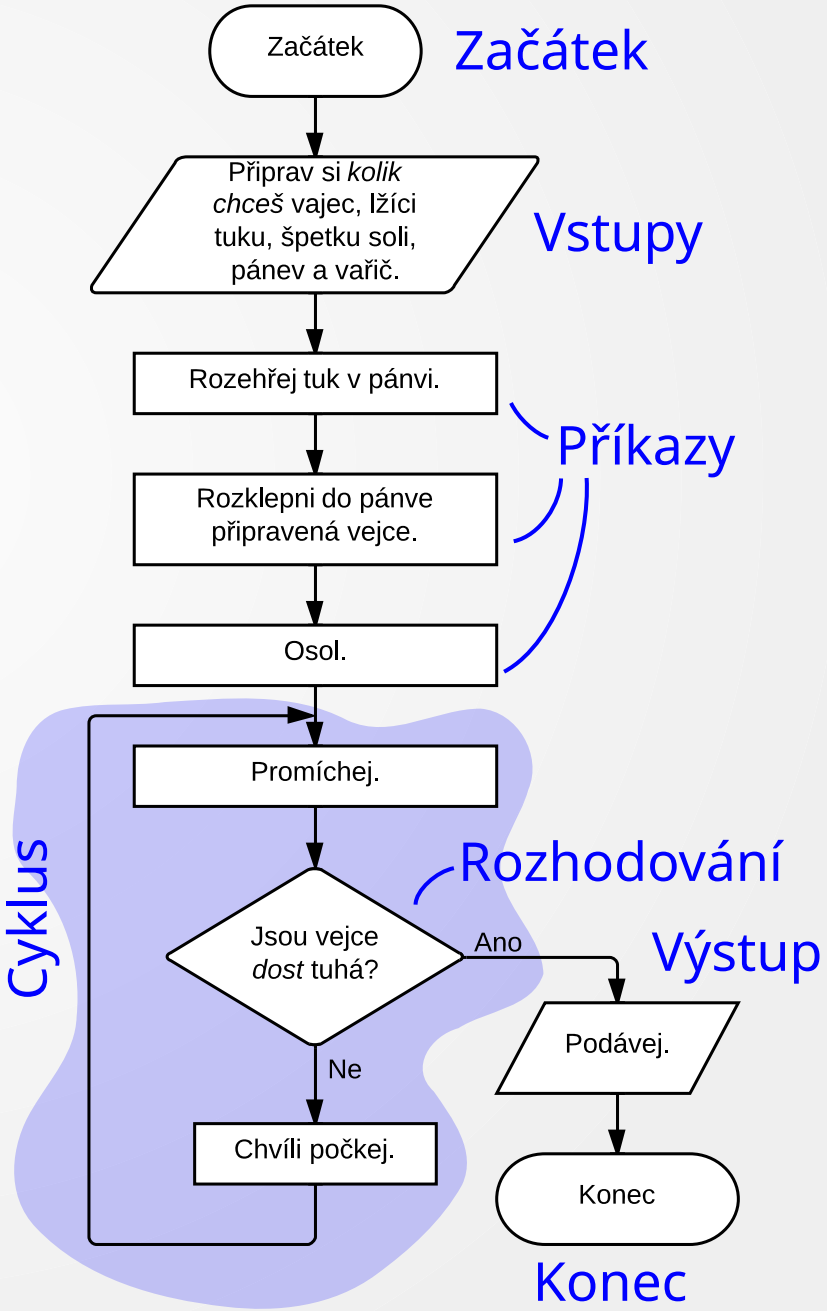



NÁVOD
JAK POROZUMĚT VÝVOJOVÝM DIAGRAMŮM
PREZENTOVÁNO FORMOU VÝVOJOVÉHO DIAGRAMU
START
ROZUMÍŠ VÝVOJOVÝM DIAGRAMŮM?
ANO
FAJN
ANO
NE
NAPIJEM SE.
6 DRINKŮ
HELE, MĚL BYCH ZKUSIT NAINSTALOVAT FREEBSD!
OUKEJ. VIDÍŠ ČÁRU OZNAČENOU „ANO"?
ANO
... A VIDÍŠ I TY OZNAČENÉ „NE"?
ANO
NE
KAŠLU NA TO
NE
ALE PRÁVĚ JSI PO NICH 2× ŠEL!
ANO
(TO NEBYLA OTÁZKA)
ALE VIDÍŠ TY OZNAČENÉ „NE"
ANO
COŽE?
NE
NE
POSLYŠ.
NESNÁŠÍM TĚ.

Varíme praženicu

- Vývojový diagram
  – spôsob zápisu algoritmu





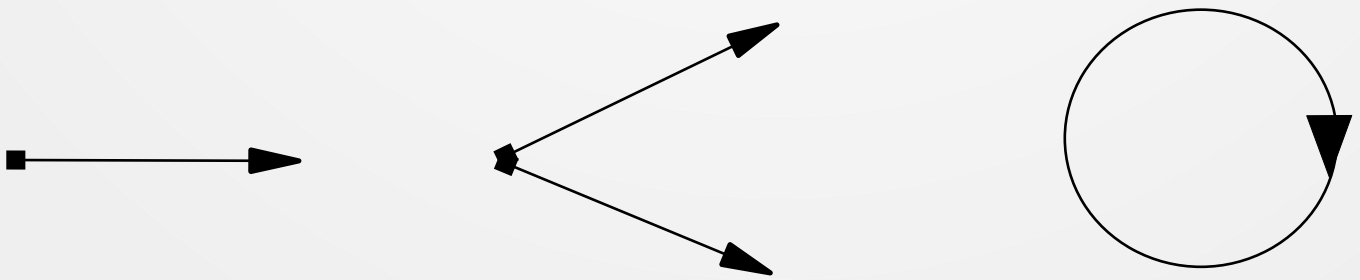
Prvky algoritmu

- Pri štrukturovanom programovaní:
- **3 pravidlá** na vytvorenie postupov zo základných pokynov
- *Cieľ: Nerobiť veci zbytočne viackrát!*

- **Sekvencia** – Prv rob toto, potom to.

- **Výber** (Podmienky) – AK je-pravda POTOM to INAK ono

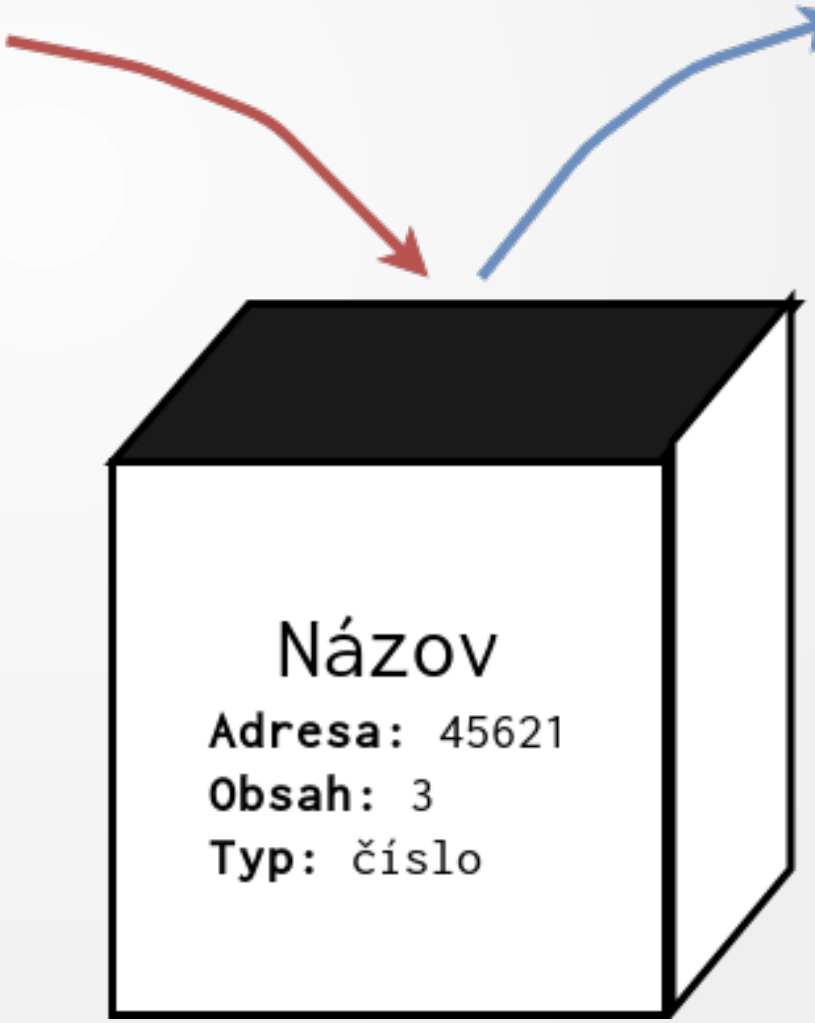
- **Opakovanie** (Cykly) – KÝM je-pravda ROB toto



Premenné

- Spôsob ukladania údajov v pamäti počítača - „Pomenovanie adresy“
- Operácie:
  - Zapíš (Vlož)
  - Čítaj (Vyber)

0	00000000
1	00000001
2	00000011
3	00000011
4	00000100
5	00000100
6	00000101
7	00000110
8	00000110
9	00000111
10	00001000
11	00001010
12	00001010



Dátový typ premenných

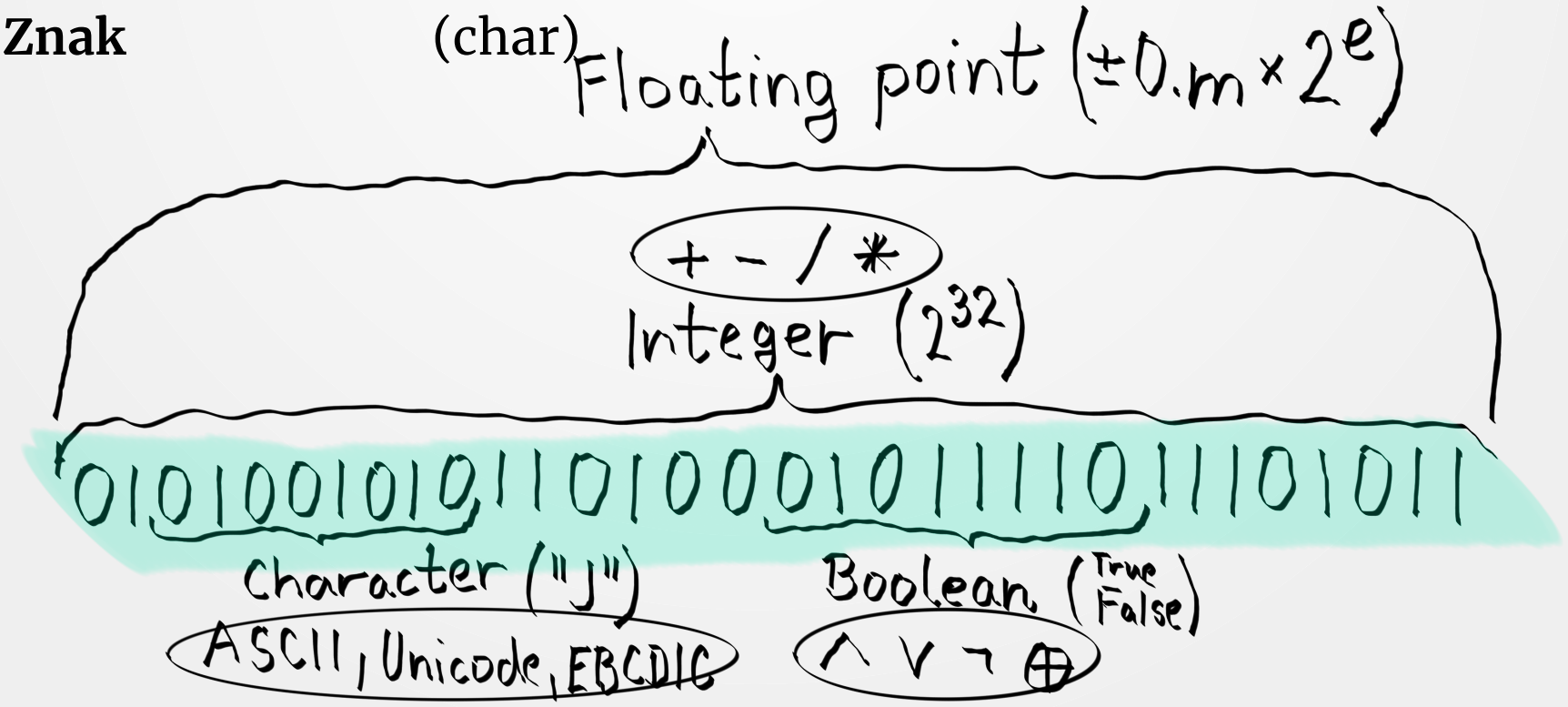
- Určuje ako máme porozumieť binárnej sekvencii v pamäti

Pravdivostný typ (bool)

Celé číslo (int)

Reálne číslo (real, float)

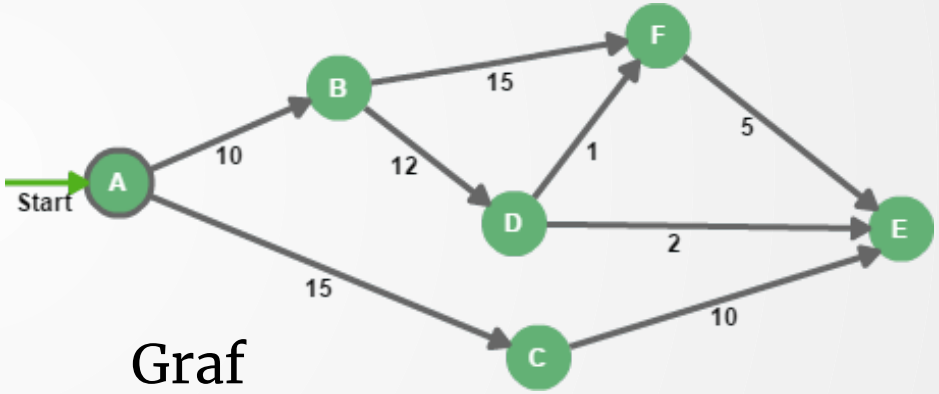
Znak (char)



Dátové štruktúry

- Zlučujú existujúce typy pod nový dátový typ, ktorý popisuje komplexnejší jav

„Volám sa TEXT“
Reťazec – pole znakov (str)



Graf

Type of Array

key	value
0	Car
1	Bicycle
2	Truck
3	Motorcycle
4	Bus

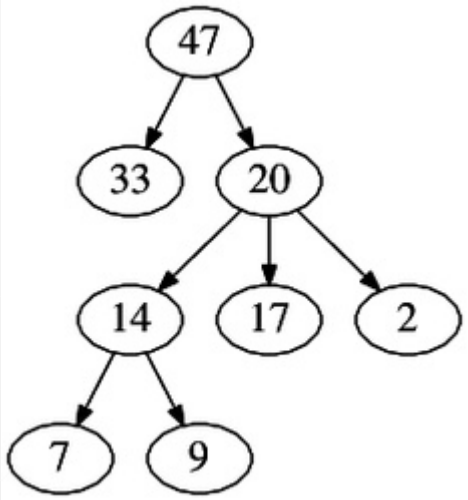
Indexed Array

key	value
id	786
pass	123
role	admin
active	1
level	1

Associative Array

Pole (Zoznam)

Asociatívne pole
(Hashovacia tabuľka)

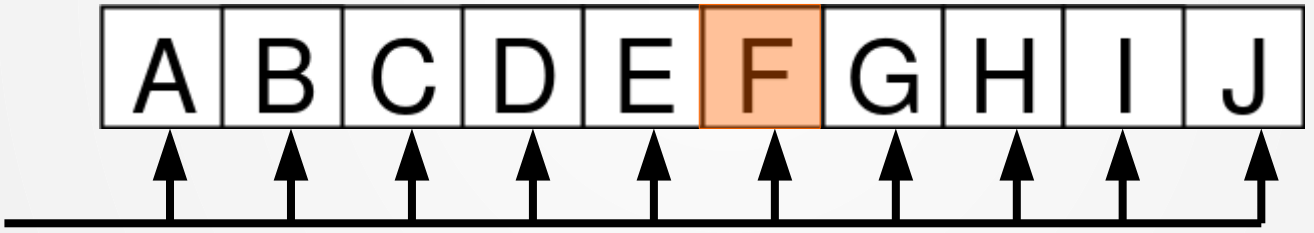


Strom

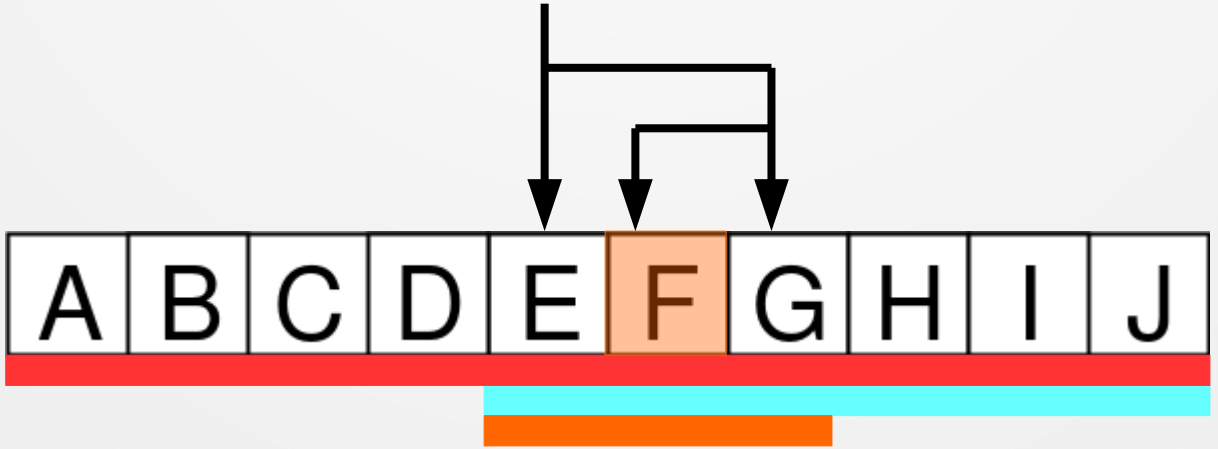
Hľadanie v slovníku

- Dôležitá vlastnosť dát: abecedné poradie -> vhodný algoritmus
- Spôsoby:

- **Lineárne** – O(n) - posupne



- **Binárne** - O(log n) – polovicu z polovice atď.



Desiatkové na binárne číslo

- **Vstup:** Kladné celé číslo *dec* v desiatkovej sústave
- **Výstup:** Reťazec *bin* predstavujúci binárny zápis čisla

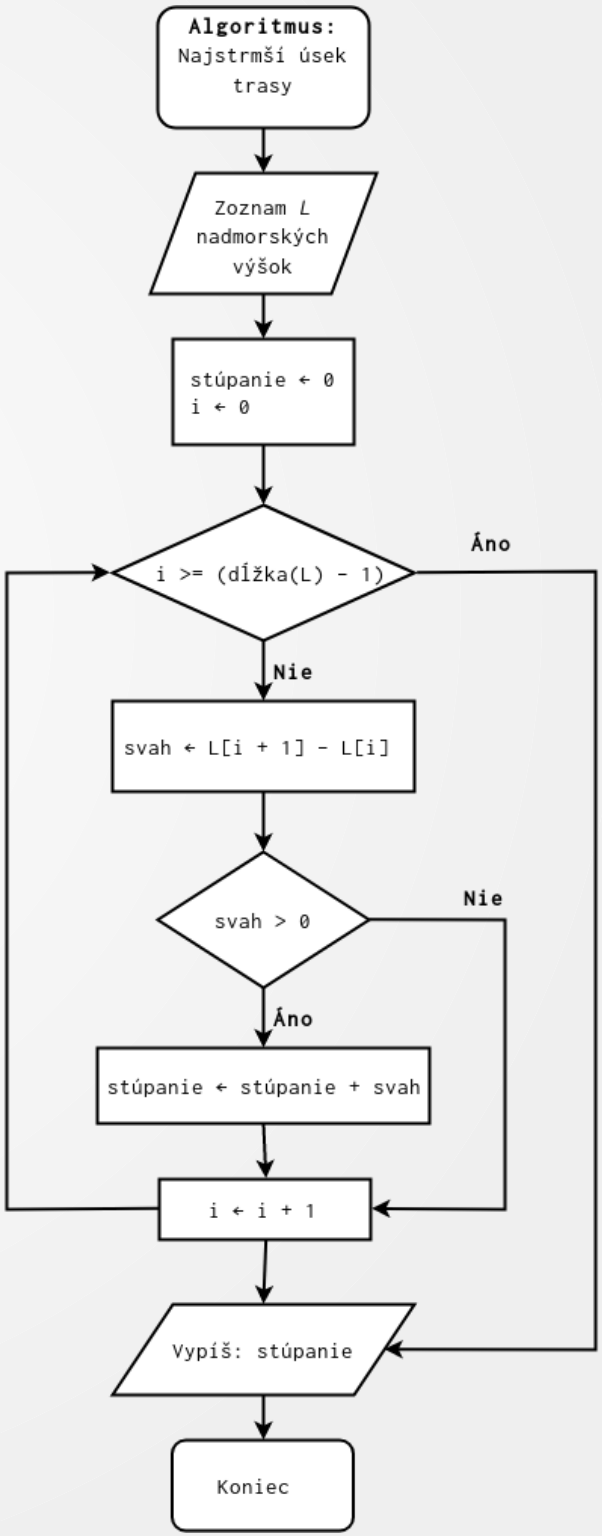
```
bin ← ""

kým dec > 0, rob:
    cifra ← znak(dec mod 2)
    bin ← bin + cifra
    dec ← dec div 2

výsledok bin
```

```
118(10) = 1110110(2)

118 : 2 =  59   zv. 0
59  : 2 =  29   zv. 1
29  : 2 =  14   zv. 1
14  : 2 =   7   zv. 0
7   : 2 =   3   zv. 1
3   : 2 =   1   zv. 1
1   : 2 =   0   zv. 1
```

Stúpanie na trase

- **Vstup:** Zoznam *L* nadmorských výšok na trase v rovnomerných vzdialenostiach
- **Výstup:** Číslo – súčet stúpaní na trase

```
stúpanie ← 0

pre i ← 0 po dĺžka(L) - 1 rob:
    svah ← L[i + 1] - L[i]
    ak svah > 0, potom:
        stúpanie ← stúpanie + svah

výsledok stúpanie
```



Najpoužívanejšie algoritmy

- Triedenie
- Fourierová transformácia (FFT) – časový na frekvenčný signál
- Dijkstrov algoritmus - navigácia, siete
- RSA, AES - šifrovanie
- Secure Hash Algorithm (SHA)
- Proporcionálny, Integračný a Derivačný regulátor - samokorekcia
- Generátor pseudonáhodných čísel - Mersenne twister
- Kompesia dát - zip, mp3, jpeg, h264
- Násobenie matíc - grafika, neurónové siete

https://mzucker.github.io/2016/09/20/noteshrink.html

https://medium.com/@_marcos_otero/the-real-10-algorithms-that-dominate-our-world-e95fa9f16c04

Vzťah: Stroj - Človek

*„Let us change our traditional attitude to the construction of programs: Instead of imagining that our main task is to **instruct a computer** what to do, let us concentrate rather on **explaining to human beings** what we want a computer to do.“*

— Donald Knuth

*„How do we convince people that in programming **simplicity and clarity** —in short: what mathematicians call **"elegance"**— are not a dispensable luxury, but a crucial matter that decides between success and failure?“*

— Edsger Dijksta

*„The psychological profiling [of a programmer] is mostly the ability to **shift levels of abstraction**, from low level to high level. To see something in the small and to see something in the large.“*

— Donald Knuth

Program(ovanie)

- strojovo čitateľný a vykonateľný popis uskutočňujúci algoritmus

```
int gcd(int a, int b)
{
    while (b > 0) {
        int t = b;
        b = a % b;
        a = t;
    }
    return a;
}
```

Prekladač

```
0: LDA [b]
1: JZ 9
2: STA [t]
3: LDA [a]
4: MOD [b]
5: STA [b]
6: LDA [t]
7: STA [a]
8: JMP 0
9: OUT [a]
10: HLT
11: /a
12: /b
13: /t
```

Adresa, Inštrukcia, Dáta

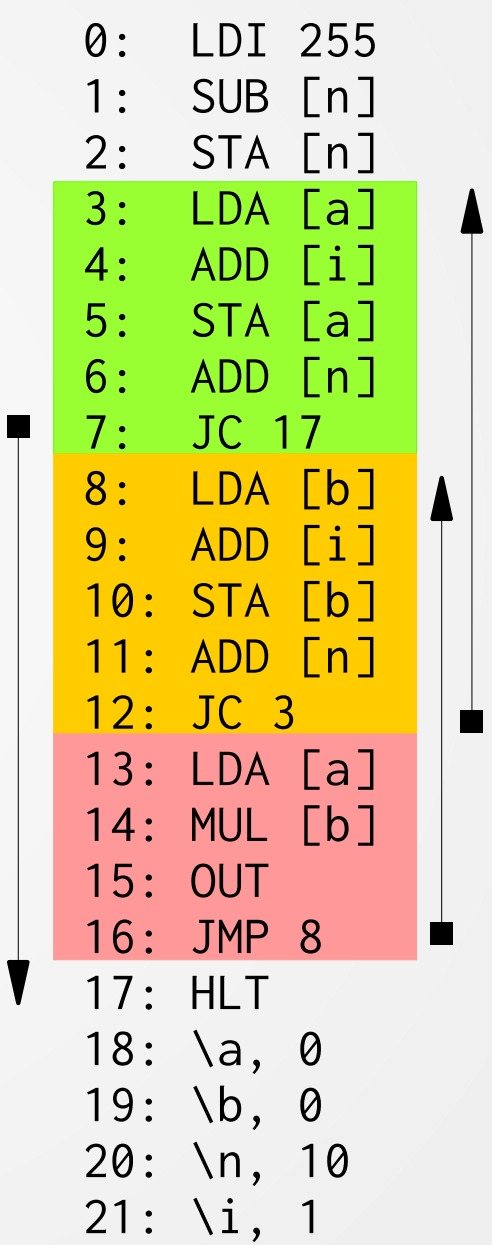
```
0000 0001 1101
0001 0011 1001
0010 0010 1101
0011 0001 1011
0100 0110 1100
0101 0011 1100
0110 0001 1101
0111 0011 1011
1000 0100 0000
1001 0101 1011
1010 1111 0000
1011 0000 0000
1100 0000 0000
1101 0000 0000
```

Počítač (Elekrické obvody)

Malá násobilka

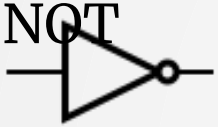
```
void nasobilka(int n)
{
    int a, b;

    for (a = 1; a <= n; a++) {
        for (b = 1; b <= n; b++) {
            printf("%d", a * b);
        }
    }
}
```

1	2	3	4	5	6	7	8	9	10
2	4	6	8	10	12	14	16	18	20
3	6	9	12	15	18	21	24	27	30
4	8	12	16	20	24	28	32	36	40
5	10	15	20	25	30	35	40	45	50
6	12	18	24	30	36	42	48	54	60
7	14	21	28	35	42	49	56	63	70
8	16	24	32	40	48	56	64	72	80
9	18	27	36	45	54	63	72	81	90
10	20	30	40	50	60	70	80	90	100



Logické hradlá

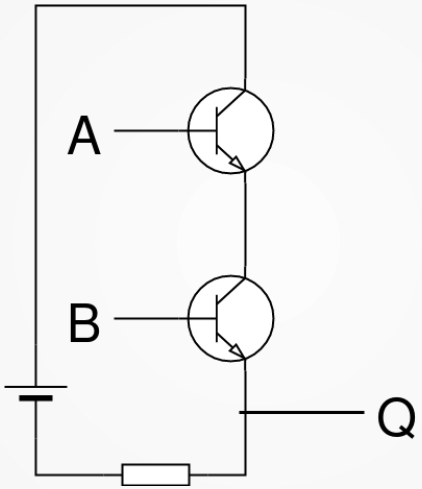
- NOT $\neg A$



- AND $A \wedge B$

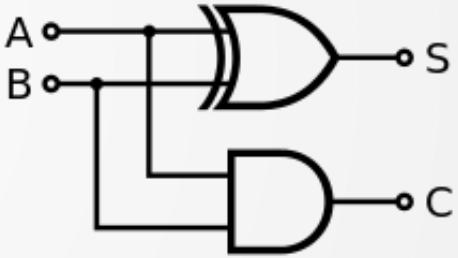
- OR $A \vee B$

- XOR $A \oplus B$

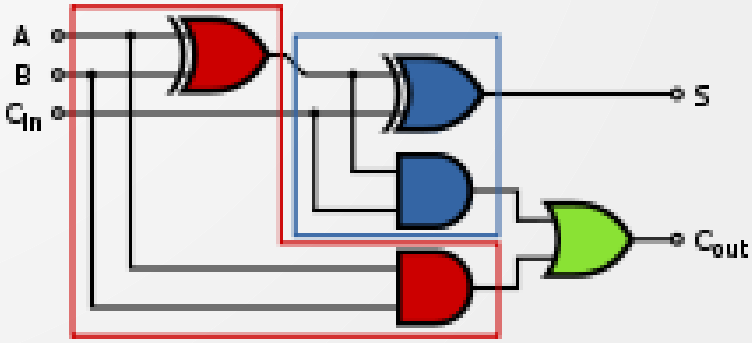


A	B	AND	OR	XOR
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Polovičná sčítačka



Úplná sčítačka (1 -bit)



$$S = A \oplus B \oplus C_{in}$$

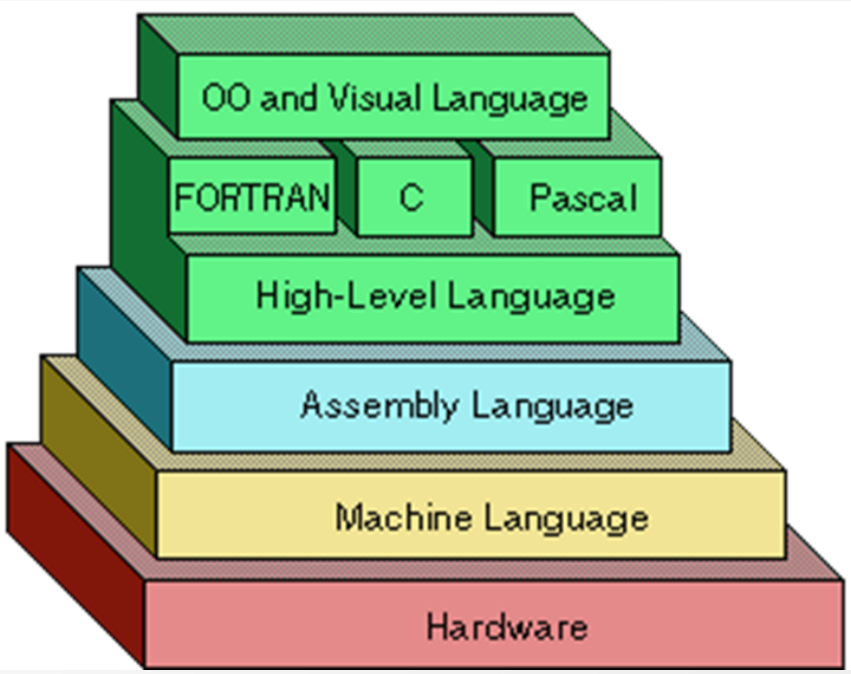
$$C_{out} = (A \wedge B) \vee ((A \oplus B) \wedge C_{in})$$

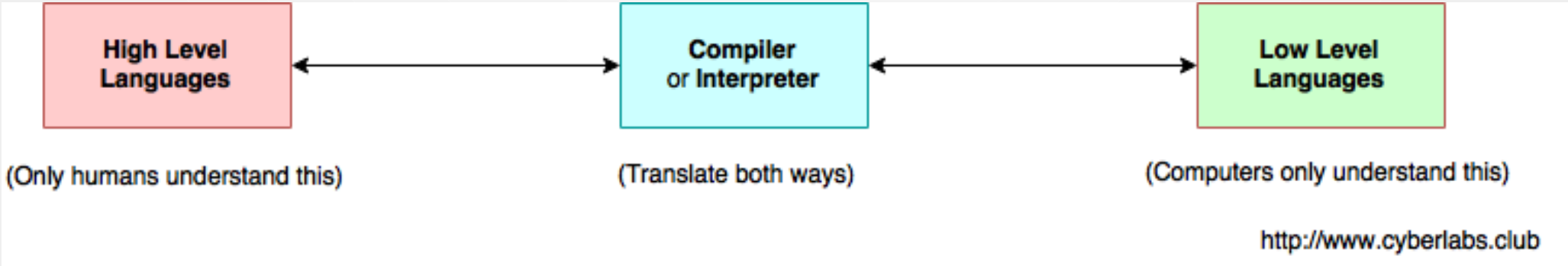
Pozn.: (1)Všetky sa dajú postaviť z NAND / NOR, (2)eater.net

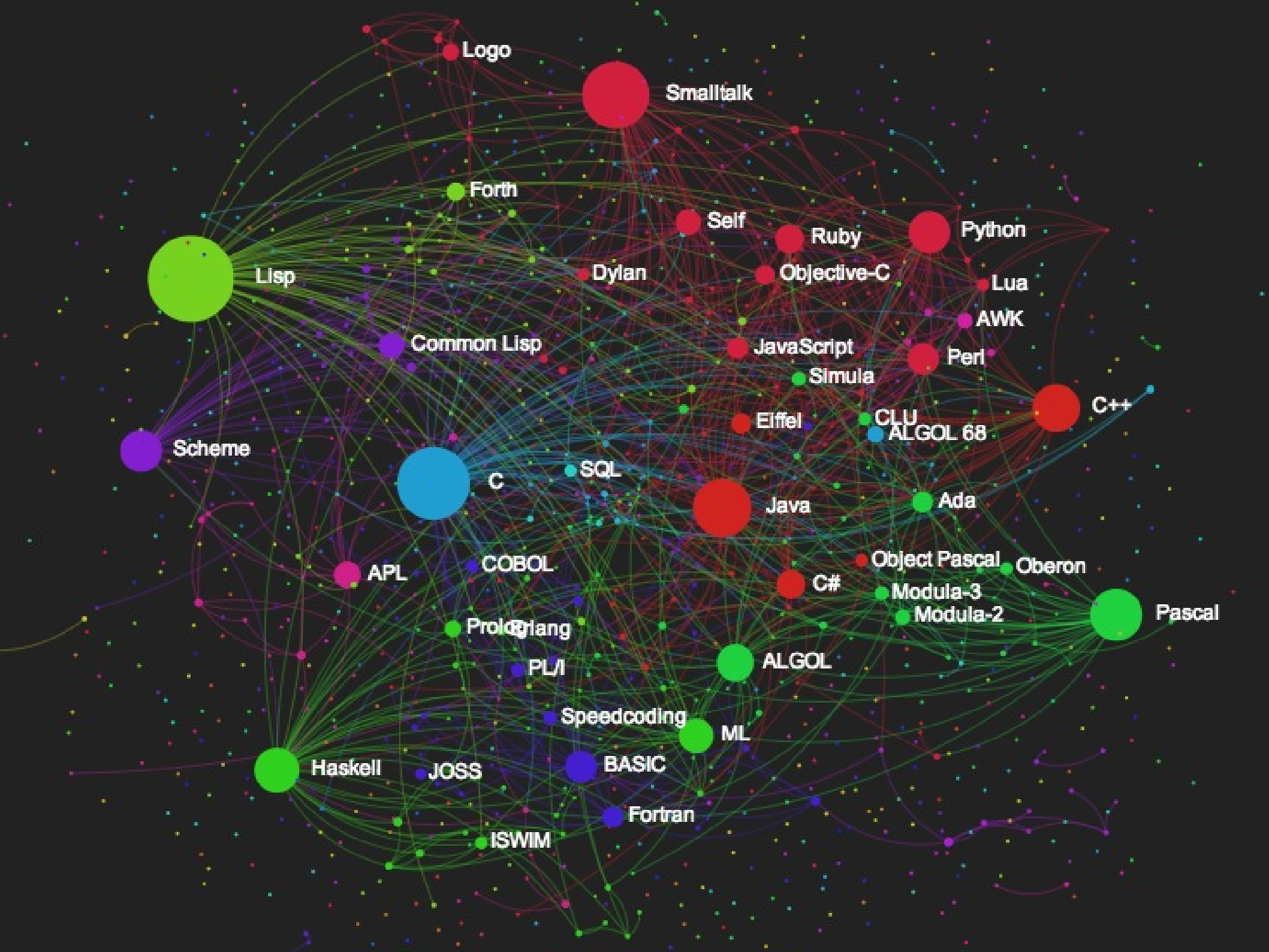
Krása programovacích jazykov

- **Abstrakcia** – virtuálny model stroja
- Hybrid medzi prirodzeným jazykom (angličtina) a matematickým algebraickým zápisom

https://rosettacode.org/






Logo
Smalltalk
Forth
Self
Ruby
Python
Lisp
Dylan
Objective-C
Lua
AWK
Common Lisp
JavaScript
Perl
Simula
C++
Eiffel
CLU
ALGOL 68
Scheme
C
SQL
Java
Ada
APL
COBOL
Object Pascal
Oberon
C#
Modula-3
Modula-2
Pascal
Prolog
Erlang
PL/I
ALGOL
Speedcoding
ML
Haskell
JOSS
BASIC
Fortran
ISWIM

WHICH PROGRAMMING LANGUAGE

SHOULD I LEARN FIRST?

WHAT IS PROGRAMMING?

Writing very specific instructions to a very dumb, yet obedient machine.

LANGUAGES

PYTHON · JAVA · C · PHP · C++ · JAVASCRIPT · C# · RUBY · OBJECTIVE-C

START

WHY DO YOU WANT TO LEARN PROGRAMMING?

- I don't know, just pick one for me
- Just for fun
- I'm interested
- Improve myself
- For my kids → Start with Scratch, then move on to...
- Get a job
- Make money
- I have a startup idea!

Have a brilliant idea/platform in mind? → YES

Which platform/field?

- Web
  - Front-end (web interface)
  - Back-end ("brain" behind a website)
  - I want to work for...
    - Corporate
      - What do you think about Microsoft?
        - I'm a fan!
        - Not Bad
        - Suck
    - Startup
- Mobile
  - Which OS?
    - iOS
    - Android
- 3D/Gaming
- Enterprise
- Web
  - Does your web app provides info in real-time, like twitter?
    - YES → JS
    - NO
- Doesn't matter, I just want $$$
- I want to work for big tech companies

Do you want to try something new with huge potential, but less mature?

- NOT SURE
- NO

Which one is your favourite toy?

- Lego
- Play-Doh
- I've an old & ugly toy, but i love it so much!

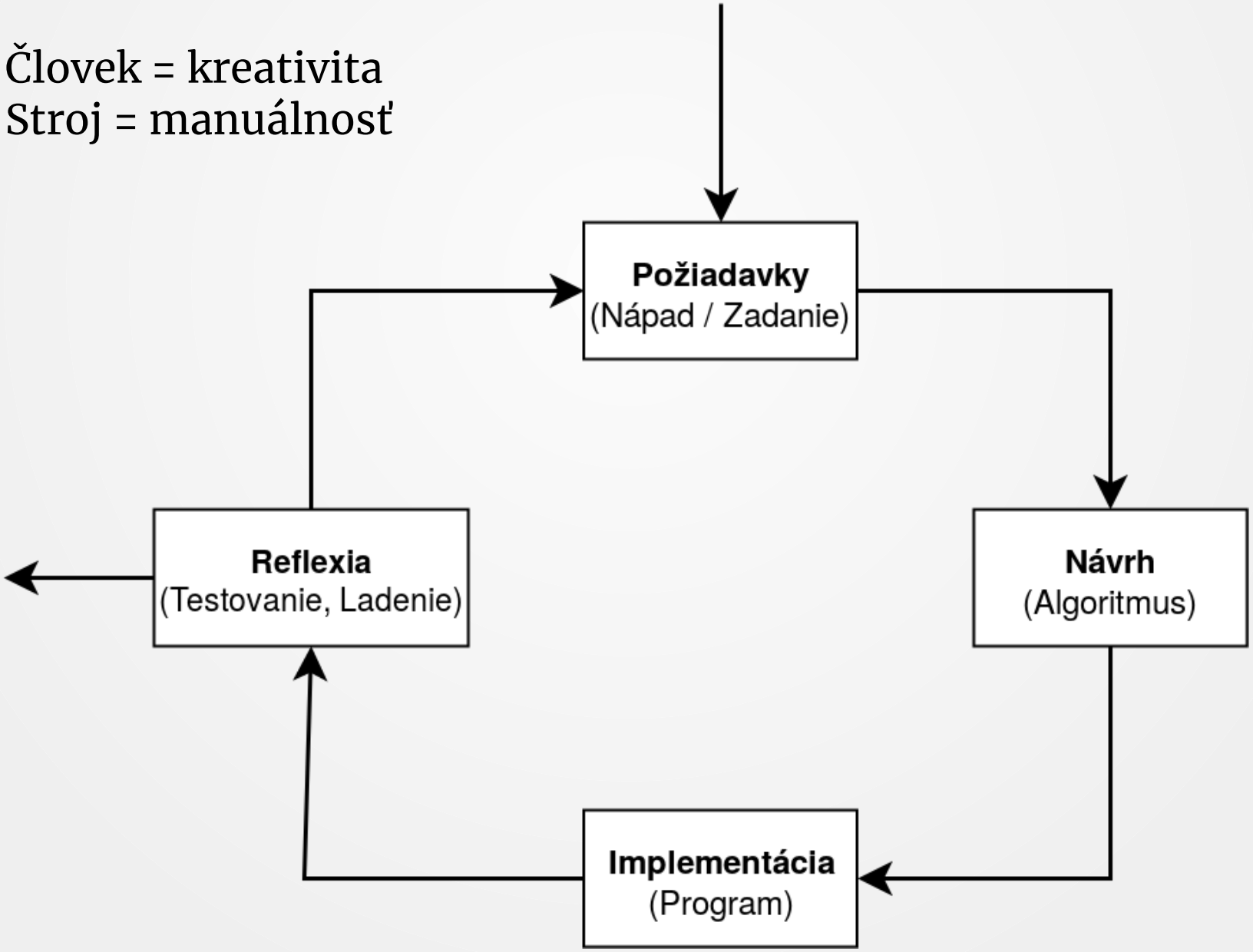
Nope. Just want to get started

I prefer to learn things...

- The easy way
- The best way
- The slightly harder way
  - Auto or Manual car?
    - Auto
    - Manual
- The really hard way (but easier to pick up other languages in the future)

Etapy vývoja softvéru

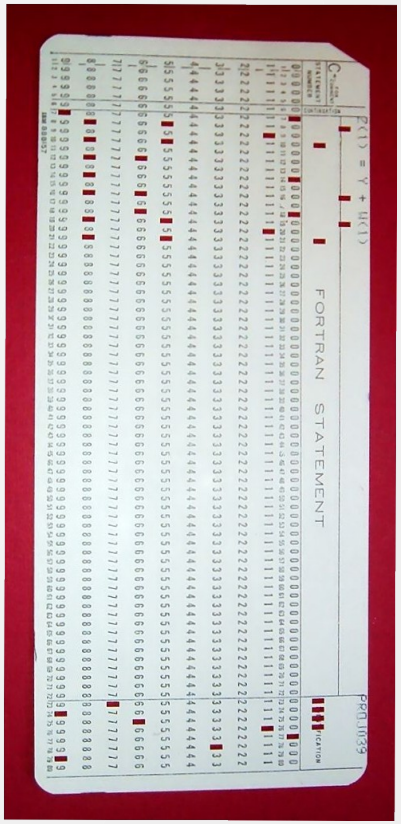
- Človek = kreativita
- Stroj = manuálnosť



FORTAN - Prvá generácia

„Formula Translation“: John Backus a IBM, 1952 - 1957

C ← FOR COMMENT / STATEMENT NUMBER (1–5)	CONTINUATION (6)	FORTRAN STATEMENT (7–72)	IDENTIFICATION (73–80)
C		PROGRAM FOR FINDING THE LARGEST VALUE	
C	X	ATTAINED BY A SET OF NUMBERS	
		BIGA = A(1)	
		DO 20 I = 2,N	
		IF (BIGA - A(I)) 10, 20, 20	
10		BIGA = A(I)	
20		CONTINUE	



Numerické riešenie obyčajných diferenciálnych rovníc

```
      READ  1, DELTAX
      PRINT  1, DELTAX
      XPRINT = 0.01
      X = 0.0
      Y = 0.0
3     Y = Y + DELTAX*(X*Y + 1.0)
      X = X + DELTAX
      IF (X - XPRINT) 3, 4, 4
4     PRINT  2, X, Y
      XPRINT = XPRINT + 0.01
      IF (X - 1.0) 3, 5, 5
5     STOP
```

$$\frac{dy}{dx} = xy + 1$$

$$\Delta y = \Delta x \cdot (x_0 y_0 + 1)$$

$$y_{i+1} = y_i + \Delta x(x_i y_i + 1)$$

$$x_{i+1} = x_i + \Delta x$$

ALGOL – Prvá generácia

"Algorithmic Language“ (ALGOL58, ALGOL60, ALGOL68)

```
procedure Absmax(a) Size:(n, m) Result:(y) Subscripts:(i, k);
    value n, m; array a; integer n, m, i, k; real y;
comment The absolute greatest element of the matrix a, of size n by m
    is transferred to y, and the subscripts of this element to i and k;
begin
    integer p, q;
    y := 0; i := k := 1;
    for p := 1 step 1 until n do
        for q := 1 step 1 until m do
            if abs(a[p, q]) > y then
                begin y := abs(a[p, q]);
                    i := p; k := q
                end
end Absmax
```

COBOL – Prvá generácia

„COmmon Business Oriented Language“ - Grace Hopper, 1959

```
IDENTIFICATION DIVISION.
PROGRAM-ID. RESEL-WORLD.
DATA DIVISION.
FILE SECTION.
WORKING-STORAGE SECTION.
*A record that describes a user
01  WS-USER.
     05  WS-FIRST-NAME   PIC a(10).
     05  WS-LAST-NAME  PIC a(10).
     05  WS-AGE        PIC 9(2).
01  WS-FULL-NAME       PIC a(20).
01  WS-CLOSE           PIC a(1).
01  WS-NEW-AGE         PIC 9(2).
01  WS-AGE-DELTA       PIC 9(2) VALUE 10.

PROCEDURE DIVISION.
*Run the code as performed paragraphs
     PERFORM GET-DATA
     PERFORM CALC-DATA
     PERFORM SHOW-DATA
     PERFORM FINISH-UP
     GOBACK.
```

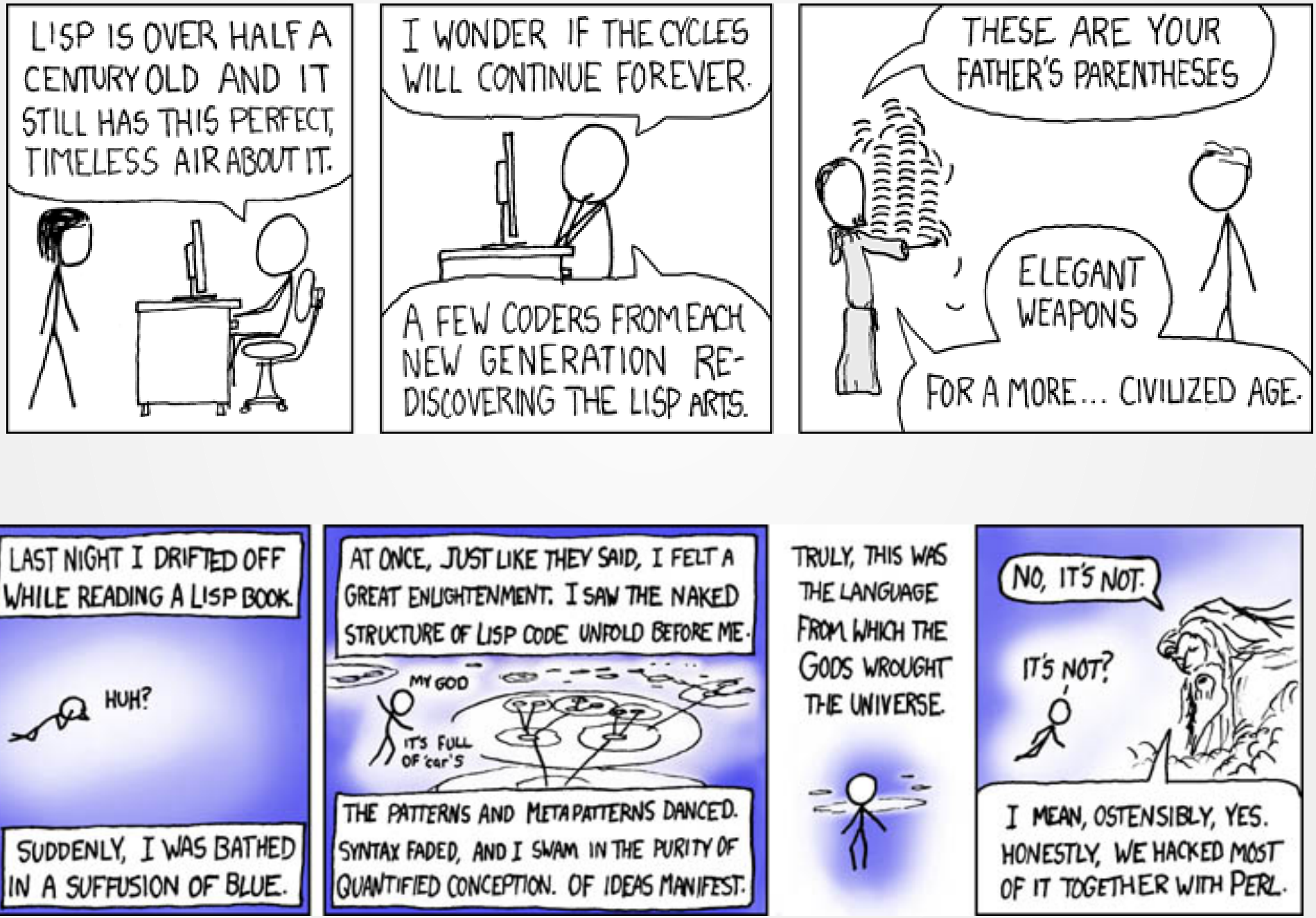
```
*A performed paragraph to get user input
GET-DATA.
     MOVE SPACE TO WS-USER WS-FULL-NAME
     DISPLAY "What is your first name?"
     ACCEPT WS-FIRST-NAME OF WS-USER
     DISPLAY "What is your last name?"
     ACCEPT WS-LAST-NAME OF WS-USER
     DISPLAY "What is your age?"
     ACCEPT WS-AGE OF WS-USER
     STRING WS-FIRST-NAME OF WS-USER DELIMITED BY SPACE
     SPACE DELIMITED BY SIZE
           WS-LAST-NAME OF WS-USER DELIMITED BY SPACE
           SPACE DELIMITED BY SIZE
           INTO WS-FULL-NAME
           ON OVERFLOW
           DISPLAY "SORRY, YOUR DATA WAS TRUNCATED"
     END-STRING.

*A performed paragraph for doing calculation
CALC-DATA.
* Sample addition statement
      ADD  WS-AGE-DELTA WS-AGE OF WS-USER TO WS-NEW-AGE.

*A performed paragraph to display output
SHOW-DATA.
      DISPLAY "Welcome " WS-FULL-NAME " In ten years you will be: "
       WS-NEW-AGE.

*A performed paragraph to end the program
FINISH-UP.
     DISPLAY "Strike any key to continue".
     ACCEPT WS-CLOSE
     DISPLAY "Good bye".
END PROGRAM RESEL-WORLD.
```

http://landoflisp.com/orc-battle.lisp

LISP - Prvá generácia

„LISt Processor“- John McCarthy, 1958 (MIT)

```
(defparameter *player-health* nil)
(defparameter *monsters* nil)
(defparameter *monster-num* 12)

(defun init-player () (setf *player-health* 30))

(defun monsters-dead () (every #'monster-dead *monsters*))

(defun player-dead () (<= *player-health* 0))

(defun game-loop ()
  (unless (or (player-dead) (monsters-dead))
    (show-player)
    (dotimes (k (1+ (truncate (/ (max 0 *player-agility*) 15))))
      (unless (monsters-dead)
        (show-monsters)
        (player-attack)))
    (fresh-line)
    (map 'list
         (lambda(m)
           (or (monster-dead m) (monster-attack m)))
         *monsters*)
    (game-loop)))
```



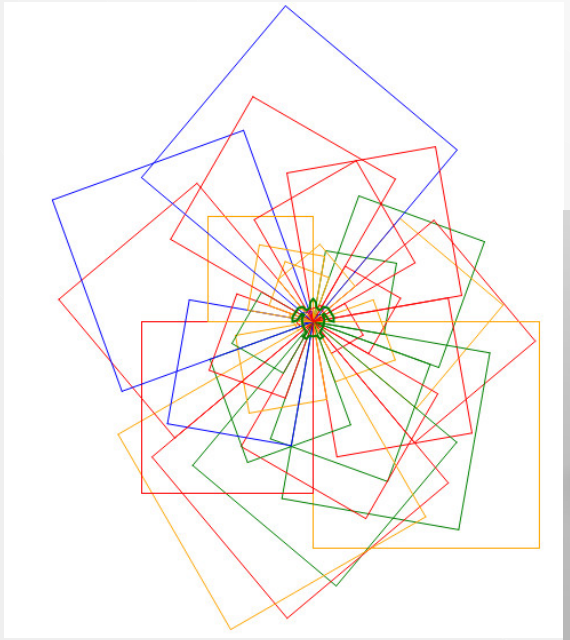
https://xkcd.com/ (224,297)

Jazyky v edukácií

Logo (1967)

- Korytnačia grafika

```
to square :length
  repeat 4 [forward :length right 90]
end
to randomcolor
  setcolor pick [red orange green blue]
end
clearscreen
repeat 36 [randomcolor square random 200
rt 10 ]
```



BASIC (1964) - ZX81

```
10 PRINT „NAPÍŠ TAJNÚ SPRÁVU“
20 INPUT $S
30 FOR J=1 TO LEN(S$)
40 LET X=CODE(S$(J TO J)
50 IF X < 38 OR X > 63 THEN LET N=X
55 IF X < 38 OR X > 63 THEN GOTO 100
60 IF INT(J/2) = J/2 THEN LET N=X+1
70 IF INT(J/2) <> J/2 THEN LET N=X-1
80 IF N < 38 THEN LET N=N+26
90 IF N > 63 THEN LET N=N-26
100 PRINT CHR$(N);
110 NEXT J
```

Súčasná prax – C, C++

```c
/* Počítanie slov a znakov na vstupe - K&R */

#include <stdio.h>

#define DNU     1   // v slove
#define VONKU   0   // mimo slova

int main(void)
{
    int z, pz, ps, stav;

    stav = DNU;
    ps = pz = 0;
    while ((z = getchar()) != EOF) {
        ++pz;
        if (z == ' ' || z == '\n' || z == '\t') {
            stav = VONKU;
        else if (stav == DNU) {
            stav = DNU;
            ++ps;
        }
    }
    printf(„Znaky: %d,  Slová: %d\n“, pz, ps);
}
```

Súčasná prax – Python

```
def narodeniny(mesiac):
    nar = []
    if mesiac < 1 or mesiac > 12:
        return nar

    ziaci = open('rodne-cisla.txt','r')

    for ziak in ziaci:
        meno, rodne = ziak.split(' ')
        m = int(rodne[2:4])
        if m > 50:
            m -= 50
        if m == mesiac:
            nar.append(meno)
    ziaci.close()
    return nar

print(narodeniny(5))
```

rodne-cisla.txt

```
Jano 18022003
Peter 09051996
Zuzka 10621998
Simon 17032001
Sara 24571997
Marika 16601999
Kamil 30072000
Katka 12612002
...
```

Súčasná prax – Domain Specific Languages

SQL (Databázy)

```
attendance                                                  student
+----+------+---------+-----------+------------+          +----+------+-------+
| id | date | arrival | departure | student_id |          | id | name | class |
+----+------+---------+-----------+------------+          +----+------+-------+
```

```
SELECT s.name, a.date, a.arrival, a.departure
FROM attendance AS a
JOIN student AS s ON s.id = a.student_id
WHERE s.name = 'Miroslav Hájek'
    AND (date BETWEEN '2018-05-01' AND '2018-05-30')
ORDER BY date;
```

Regex (Dynamické hľadanie v textoch)

```
^(([0-9]{5})|([0-9]{3}[ ]{0,1}[0-9]{2}))$
```

PSČ (95106, 834 09)

Ďakujem za pozornosť

„What I cannot create,
I do not understand.“

— Richard Feynmann

Zdroje

Text:

- https://medium.freecodecamp.org/how-to-think-like-a-programmer-lessons-in-problem-solving-d1d8bf1de7d2
- https://www.britannica.com/biography/Alan-Turing
- https://www.interviewcake.com/article/python/data-structures-coding-interview?course=dsa
- Jack Woehr. An interview with Donald Knuth. Dr. Dobb's Journal, pages 16-22 (April 1996)
- https://www.sitepoint.com/demystifying-regex-with-practical-examples/
- https://www.fortran.com/FortranForTheIBM704.pdf (str. 7)
- https://www.ibm.com/ibm/files/T507917N26894N49/us___en_us___ibm100___fortran___programmers_primer.pdf (str. 17)

Obrázky:

- Muḥammad ibn Mūsā al-Khwārizmī - The Bodleian Library, University of Oxford, CC BY 4.0, https://commons.wikimedia.org/w/index.php?curid=43327479
- Princeton University, Fair use, https://en.wikipedia.org/w/index.php?curid=6082269
- http://www.arithmeum.uni-bonn.de/en/events/285, https://en.wikipedia.org/w/index.php?curid=43426215
- https://popelka.ms.mff.cuni.cz/~lessner/mw/index.php/U%C4%8Debnice/Algoritmus/V%C3%BDvojov%C3%A9_diagramy
- https://vyuka.prokyber.cz/projects/ite/wiki/Diagram
- By https://www.amazon.com/Art-Computer-Programming-Fundamental-Algorithms/dp/0201896834/ref=pd_bbs_sr_2?ie=UTF8&s=books&qid=1219447664&sr=8-2, Fair use, https://en.wikipedia.org/w/index.php?curid=18987672
- By Source (WP:NFCC#4), Fair use, https://en.wikipedia.org/w/index.php?curid=39956853
- https://www.ksp.sk/kucharka/
- http://webdevzoom.com/understanding-array-in-php/
- https://exploring-data.com/vis/programming-languages-influence-network/
- https://codeburst.io/what-programming-language-should-i-learn-f3f164ca376c
- https://thebittheories.com/levels-of-programming-languages-b6a38a68c0f2
- By Arnold Reinhold - I took this picture of an artifact in my possession. The card was created in the late 1960s or early 1970s and has no copyright notice., CC BY-SA 2.5, https://commons.wikimedia.org/w/index.php?curid=775153
- http://archive.computerhistory.org/resources/text/algol/ACM_Algol_bulletin/1064048/frontmatter.pdf
- http://www.calormen.com/jslogo
- https://www.root.cz/clanky/ceskoslovenske-osmibitove-pocitace-2-ndash-pmd-85/