## Nejjednodušší program

Počítače dokáží hodně, ale my umíme jen málo programovat. Přesto by to chtělo vidět, že i s tím málem něco spočteme.

Jako příklad vyřešíme kvadratickou rovnici

$$x^2 + x - a = 0$$

Nepoužijeme ale obvyklý vzorec, který najde obě správná řešení, ale využijeme ochoty počítače hodně počítat. Nejprve rovnici přepíšeme do tvaru

$$x = a - x^2.$$

A nyní opakujeme (iterujeme)

$$x_0 = 0$$
$$x_1 = a - x_0^2$$
$$x_2 = a - x_1^2$$
$$\ldots$$
$$x_{20} = a - x_{19}^2$$

Program bude pro kontrolu vypisovat tabulku

$$0 \quad x_0$$
$$1 \quad x_1$$
$$\ldots$$
$$20 \quad x_{20}$$

a my pak zkontrolujeme, zda se hodnoty neustálí. To by znamenalo, že máme řešení rovnice.

Vyzkoušeli jsme dva počítačové jazyky, Pascal

```
program Hello;
var k : integer;
    x : real;      // proměnné, se kterými pracujeme, musíme v Pascalu deklarovat
const a = 0.2;
begin
  k := 0;
  x := 0;                          // začínáme s číslem 0
  while k < 21 do begin            // dokud je podmínka splněna, opakuj
      writeln (k,'␣', x:14:7);     // výpis výsledku
      k := k+1;
      x := a-x*x;                  // náš vzorec pro řešení kvadratick é rovnice
  end;
end.
```

a Python.

```
# Tento program zkouší řešit kvadratickou rovnici

a = 0.2
x = 0                            # začínáme s číslem 0
k = 0

while k<21:                      # dokud je podmínka splněna, opakuj
    print(k,x)                   # výpis výsldeku
    x = a-x*x                    # náš vzorec pro řešení kvadratické rovnice
    k = k+1
```

Výstup obou je podobný

```
0 0
1 0.2
2 0.16
3 0.1744
4 0.16958464
...
18 0.17082039288624637
19 0.17082039337418845
20 0.17082039320748754
```

Student snadno v obou jazycích pozná následující prvky, z nichž je program sestaven: komentáře, čísla, identifikátory, aritmetické výrazy, přiřazovací příkazy, příkazy cyklu **while**, podmínky opakování cyklu. V jazyce Pascal máme navíc deklarace proměnných včetně typu dat, která proměnné obsahují.

### Cyklus for

Je nepohodlné a náchylné k chybě starat se na třech místech o řízení cyklu (inicializace proměnné k, testováni na konec cyklu a zvyšování hodnoty k).

Proto máme cyklus for v Pascalu

```
...
for k:=0 to 20 do begin          //  opakuj pro k =0,1,2,...,20
    writeln (k,'␣', x:14:7);     //  výpis výsledku
    x := a-x*x;                  //  náš vzorec pro řešení  kvadratick é  rovnice
end;
...
```

i v Pythonu (zde pozor na horní mez cyklu)

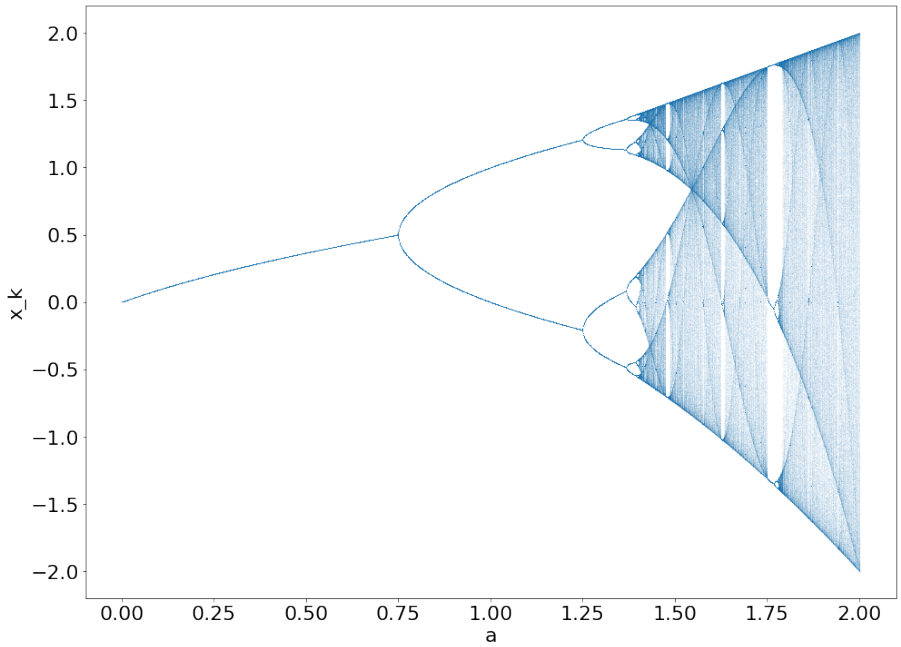
```
for k in range(0,21):            # opakuj pro k=0,1,2,...,20
    print(k,x)                   # výpis výsldeku
    x = a-x*x                    # náš vzorec pro řešení kvadratické rovnice
```

### Výstup programu

Data, která program vypsal mohou býto použita dále. Pokud tedy výstup našeho programu uložíme do souboru, můžeme jeho obsah např. převést do grafické podoby. Tyto kroky si ještě procvičíme podrobněji. Na cvičení jsme výstup s použitím gnuplotu [2] vykrelili jako lomenou čáru. A viděli, že její pdoba závisí ho dondotě $a$.

### Zajímavost

Ne pro všechny hodnoty $a$ náš postup řešení kvadratické rovnice funguje. Pro větší hodnoty přestane posloupnost konvergovat k řešení rovnice a místo začně nabývat mnoha různých hodnot. Závislost na parametru $a$ je dost zajímavá [3,4].




Obrázek 1. Na vodorovné ose je hodnota parametru $a$. Na vertikální pak pro mnoho různých $a$ hodnoty členů posloupnosti $x_{1000}, ..., x_{3000}$. Pokud posloupnost konverguje, slijí se všechny body do jednoho, pokud osciluje mezi dvěma hodnotami slije se do dvou. Překvapivě je ale pro některá $a$ možných hodnot $x_k$ velmi hodně.

### Odkazy


[1] `https://www.onlinegdb.com/` – Jedno z mnoha prostředí pro spouštění programů skrze webový prohlížeč.
[2] `http://gnuplot.respawned.com/` – Gnuplot ve vašem webovém prohlížeči.
[3] `https://en.wikipedia.org/wiki/Logistic_map` – Výklad chování
[4] `https://colab.research.google.com/drive/158LFtrBY0nHUHrkVSyeP69wnPzwqumXJ` – Kód vytvářející obrázek výše. (Kdybyste se k problému chtěli vrátit, až se naučíme kreslit obrázky.)