# CVIČENÍ 1

**Téma:** Indukce a rekurze/rekurence v programování a v matematice
**Cíle:** Zvládnutí techniky důkazu jednoduchých tvrzení matematickou indukcí, induktivně definované struktury a jejich vlastnosti, rekurzivní návrh funkcí pracujících s těmito strukturami.

**Úvod do předmětu:**
Seznamte studenty:
1. s Vaší maličkostí – uveďte na sebe kontakt a konzultační hodiny
2. se střídáním proseminářů a cvičení
3. s bodováním (viz edux )

**Matematická indukce:**

1. silná indukce - má jiný indukční krok:

slabá: z předpokladu platnosti S(n) dokážeme platnost S(n+1)

silná indukce: z předpokladu platnosti S(0), S(1), ..., S(n) dokážeme platnost S(n+1)

Příklad na důkaz indukcí: Mějme následující induktivní definici funkce f : N $\rightarrow$ N:

$$f(0) = 0,\ f(1) = 1$$

$$\left.\begin{aligned} f(2n) &= f(n) \\ f(2n+1) &= f(n) + f(n+1) \end{aligned}\right\} \text{ pro } n = 1, 2, 3, \ldots$$

Pro všechna n $\geq$ 0 dokažte pro tuto funkci ***f(n)*** platnost tvrzení:

***V(n) = "f(n) je sudé, právě když je n dělitelno třemi"***

2. Zavedeme (neúplně) ADT ListInt induktivně takto:
   - prázdný seznam je ListInt (vytvoří se pomocí **`initL()`** a testuje pomocí predikátu **`emptyL(L)`**)
   - je-li x celé číslo a L je ListInt, pak přidáním prvku x před začátek seznamu S (výsledek operace **`cons(x,L)`**) je rovněž ListInt
   - nic jiného než to, co vzniklo použitím výše uvedených pravidel, není ListInt

   Máme dále k dispozici operace **`first(S)`** a **`last(S)`**, které vrací první, resp poslední znak neprázdného seznamu S, a operace **`butFirst(S)`** a **`butLast(S)`**, které vrací zbytek seznamu S po odebrání prvního, resp. posledního prvku. Definujte nyní induktivně následující operace se seznamy:
   - **`length(L)`** – délka seznamu L (počet jeho prvků)
   - **`max(L)`** – maximální hodnota prvku v (neprázdném!) seznamu L
   - **`ordered(L)`** – predikát testující, zda prvky seznamu L tvoří neklesající posloupnost

   Na základě těchto definic vytvořte odpovídající rekurzivní funkce.

3. Zavedeme (neúplně) induktivní formou ADT String takto:
   - prázdný řetěz je řetěz (vytvoří se pomocí **`initS()`** nebo **`""`** a testuje pomocí predikátu **`emptyS(S)`**)
   - je-li c libovolný znak a S řetěz, pak také výsledek připojení znaku c na začátek (výsledek operace **`addFirst(c,S)`**) nebo na konec (výsledek operace **`addLast(S,c)`**) je řetěz
   - nic jiného …

   Máme dále k dispozici operace **`first(S)`** a **`last(S)`**, které vrací první, resp poslední znak neprázdného řetězu S. Definujte nyní induktivně následující operace se řetězy:
   - **`concat(S,R)`** – spojí za sebe řetězy S a R
   - **`lesseqStr(S,R)`** – vrací **`true`**, je-li řetěz S lexikograficky menší nebo roven řetězu R

4. Silná indukce není nic jiného než slabá indukce pro tvrzení P(n) = S(0) & S(1) & ... & S(n)
   Procvičit dokazování jednoduchých vlastností na několika příkladech týkajících se stromů.
   a) Zavedeme **pravidelný strom stupně 2** (**PSS2**) následující „množinovou“ definicí jako:
   - samostatný uzel (list) je PSS2

- uzel (kořen) s podřízenou dvojicí PSS2 {L, R} je rovněž PSS2
- nic jiného než struktura vzniklá použitím výše uvedených pravidel není PSS2

b) Připomeneme názvosloví: **listy** stromu jsou uzly, které nemají žádné podřízené stromy, ostatní jsou **vnitřní uzly; hloubka uzlu** je jeho vzdálenost od kořene; **výška uzlu** je maximální vzdálenost od listů; **výška stromu** je rovna výšce jeho kořene

c) Vhodným použitím indukce dokážeme následující

- **Tvrzení 1**: PSS2 s n (≥0) vnitřními uzly má n+1 listů.
- **Tvrzení 2**: PSS2 s n (≥0) vnitřními uzly má 2.n hran.
- **Tvrzení 3**: Výška PSS2 s n (≥1) listy je nejvýše rovna n-1.
- **Tvrzení 4**: PSS2 o výšce n (≥0) má celkem nejvýše $2^n$ listů.

5. Předpokládejme, že PSS2 jsou implementovány obdobně jako binární stromy, každý uzel má ale nejen složky **`left`**, **`right`**, ale navíc i složky **`depth`**, **`height`**, **`count`**. Navrhněte rekurzivní algoritmy pro:

– výpočet hodnot složky **`height`** (výška) každého uzlu zadaného PSS2
– výpočet hodnot složky **`count`** (počet uzlů příslušného podstromu) každého uzlu zadaného PSS2
– výpočet hodnot složky **`depth`** (hloubka) každého uzlu zadaného PSS2

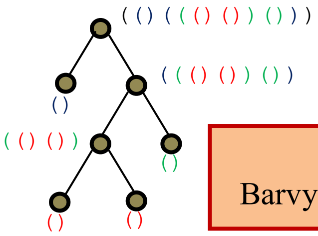
**Návod**: začněte rekurentními definicemi příslušných parametrů, které budou vycházet z induktivní definice PSS2, předpokládejte existenci predikátu **`Boolean list(T)`**

**Induktivní definice dalších typů stromů a jejich reprezentace**

6. Jak induktivně definovat
   – binární strom
   – n-ární strom
   – pravidelný strom stupně r
   – (obecný) kořenový strom
   – uspořádaný kořenový strom

   Pro každý typ stromu nakreslit charakteristický příklad, zjistit základní (elementární) alternativu a pravidlo konstrukce obecného případu

7. Jak bychom mohli vyjádřit strukturu nějakého stromu jistého typu pomocí řetězu znaků?




POZOR
Barvy jen rozlišují úroveň závorek!

Pravidla vyjádřit opět induktivním způsobem korespondujícím s příslušnou definicí daného typu stromu. Kromě struktury je třeba pamatovat i na identifikaci uzlu / uložení nějaké hodnoty.

8. Jak úplně jinak by se dala vyjádřit struktura stromu, který má např. očíslované uzly 1, 2, ..., n?