Rekurze (viz Rekurze)

PLIN048 – Základy programování pro humanitní obory

Richard Holaj
FF MU

24. března 2017

Co nás dnes čeká?

- Vnořené funkce
  - Volání funkce uvnitř funkce
  - Princip vkládání a dosazování
  - Princip vkládání a dosazování
- Rekurze
  - Princip rekurze – přímá a nepřímá
  - Analýza rekurze
  - Použití rekurze
- Cvičení

Volání funkce uvnitř funkce

- funkce (podprogram) $\sim$ *program*$\rightarrow$ lze uvnitř volat funkce
- definice mimo funkci
- postupné vyhodnocování $\rightarrow$ zanořená volání

Princip vkládání a dosazování I

Kód

```
1  def add(x, y):
2      print(x)
3      print(x)
4      return x + y
5
6  def multiply(a, b):
7      res = 0
8      for i in range(b):
9          res = add(res, a)
10     return res
11
12 multiply(4, 2)
```

Interpretace

```
1  a = 4
2  b = 2
3  res = 0
4  for i in range(b):
5      x = res
6      y = a
7      print(x)
8      print(y)
9      add = x + y
10     res = add
```

Princip vkládání a dosazování II

Kód

```
1  def neg(z):
2      print(z)
3      return -z
4
5  def add(x, y):
6      print(x, y)
7      return x + y
8
9  def substract(a, b):
10     print(a, b)
11     return add(a, neg(b))
12
13 print(substract(10, 5))
```

Interpretace

```
1  a = 10
2  b = 5
3  print(a, b)
4  z = b
5  print(z)
6  neg = -z
7  x = a
8  y = neg
9  print(x, y)
10 add = x + y
11 substract = add
12 print(substract)
```

Princip rekurze – přímá a nepřímá

- funkce volající sama sebe
- pozor na nekonečné zanoření (koncové podmínky)
- $A \rightarrow A$ (přímá) x $A \rightarrow B \rightarrow \ldots \rightarrow A$ (nepřímá)

Analýza rekurze - tail recursion

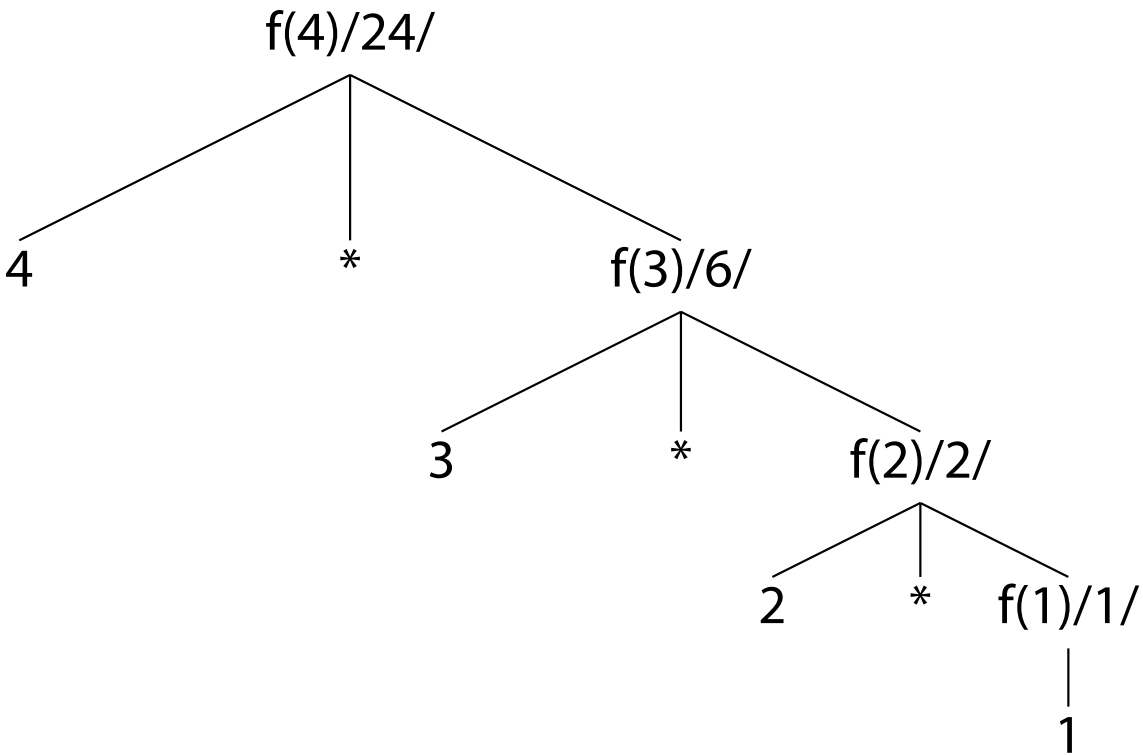
Kód

```
1  def f(x):
2      if x <= 1:
3          print("stop")
4          return 1
5      else:
6          print(x)
7          return x * f(x - 1)
8
9  def f_iter(n):
10     res = 1
11     for i in range(n, 0, -1):
12         res = res * i
13     return res
14
15 f(4)
```

Interpretace

```
1  x_4 = 4
2  print(x_4)
3  x_3 = 3
4  print(x_3)
5  x_2 = 2
6  print(x_2)
7  x_1 = 1
8  print("stop")
9  f_1 = 1
10 f_2 = x_2 * f_1
11 f_3 = x_3 * f_2
12 f_4 = x_4 * f_3
```

Analýza rekurze - tail recursion (reprezentace)



Analýza rekurze - single point

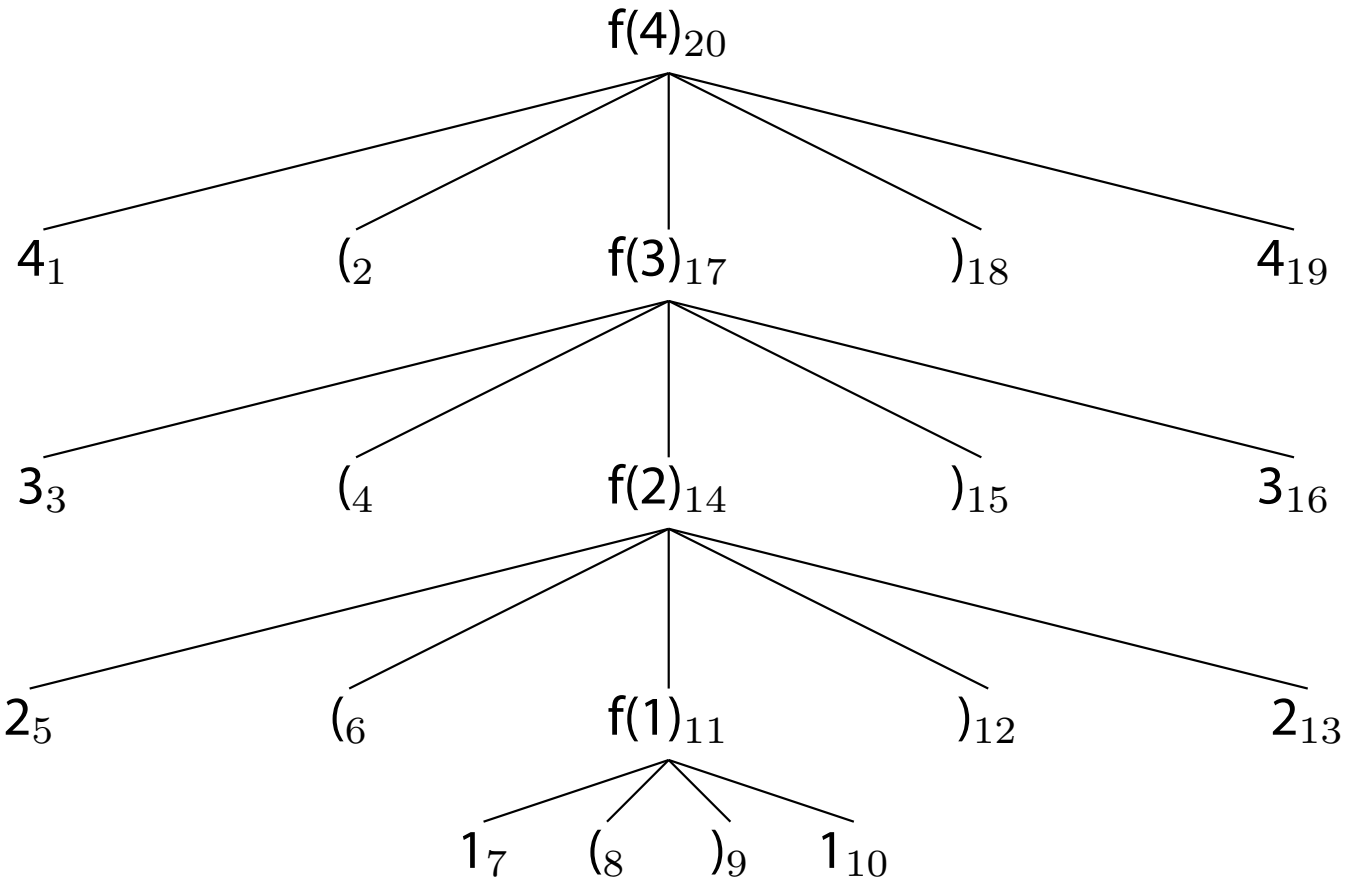
Kód

```
1  def brackets(n):
2      print(n)
3      print("(")
4      if n > 1:
5          brackets(n - 1)
6      print(")")
7      print(n)
8
9  brackets(3)
```

Interpretace

```
1  n_3 = 3
2  print(n_3)
3  print("(")
4  n_2 = 2
5  print(n_2)
6  print("(")
7  n_1 = 1
8  print(n_1)
9  print("(")
10 print(")")
11 print(n_1)
12 print(")")
13 print(n_2)
14 print(")")
15 print(n_3)
```

Analýza rekurze - single point (reprezentace)



Analýza rekurze - multi point

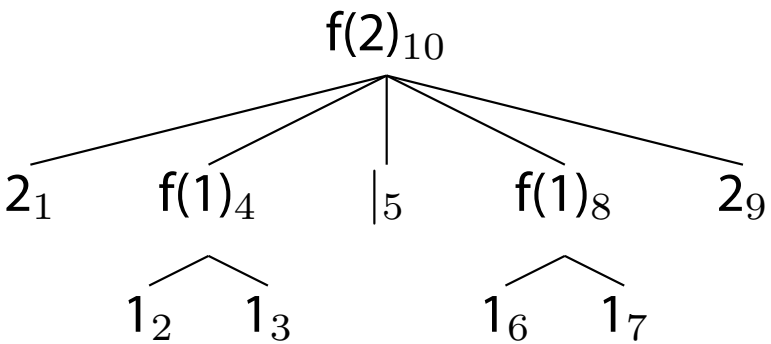
Kód

```
1 def brackets(n):
2     print(n)
3     if n > 1:
4         brackets(n - 1)
5         print("|")
6         brackets(n - 1)
7     print(n)
8
9 brackets(2)
```

Interpretace

```
1  n_2 = 2
2  print(n_2)
3  n_1_a = 1
4  print(n_1_a)
5  print(n_1_a)
6  print("|")
7  n_1_b = 1
8  print(n_1_b)
9  print(n_1_b)
10 print(n_2)
```

Analýza rekurze - multi point (reprezentace)



Analýza rekurze - expression multi point

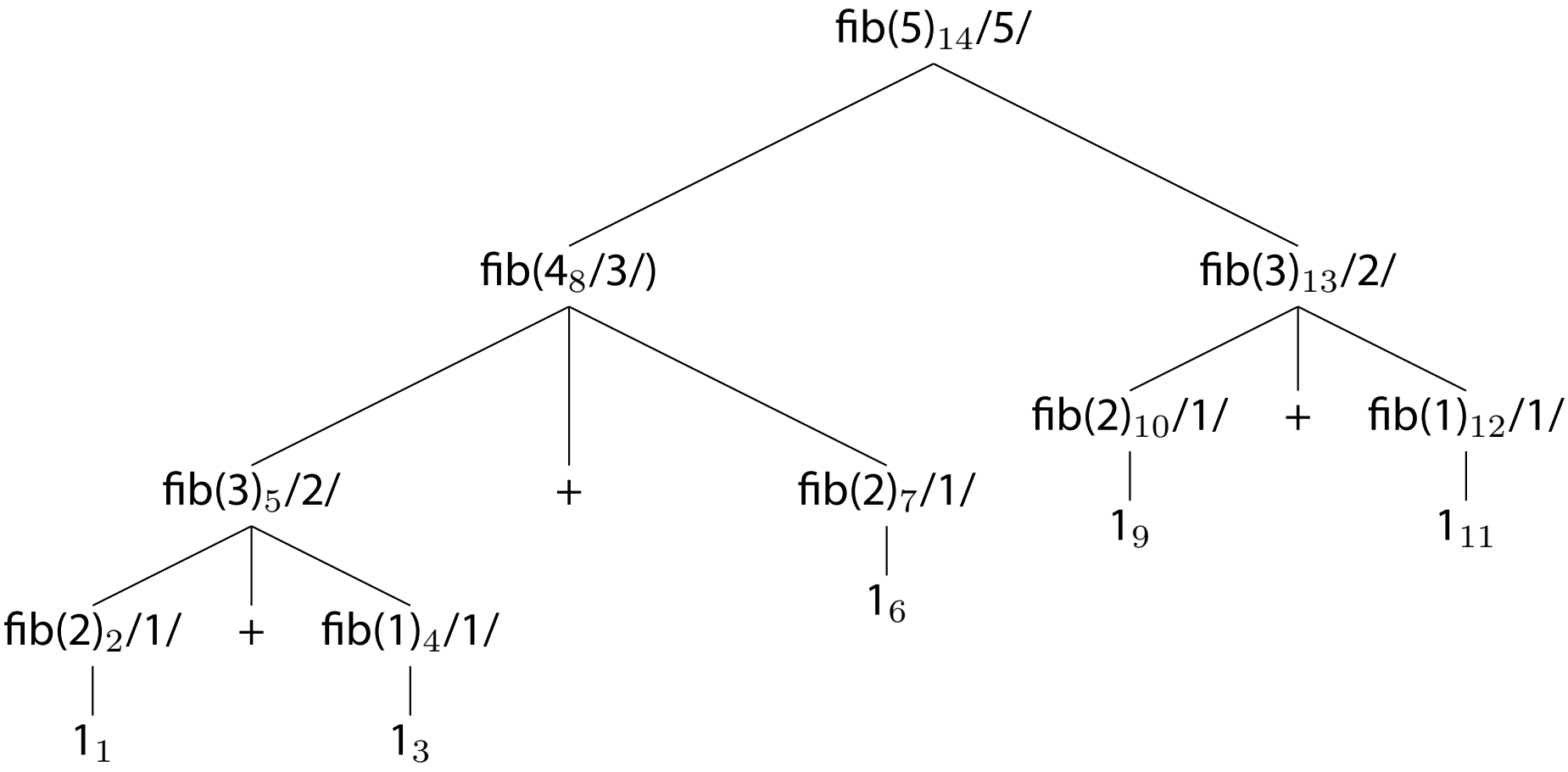
Kód

```
1 def f(n):
2   if n <= 2:
3     return 1
4   else:
5     return f(n-1) + f(n - 2)
6
7 fib(5)
```

Interpretace

```
1  n_4 = 4
2  n_3_a = 3
3  n_2_a = 2
4  f_2_a = 1
5  n_1_a = 1
6  f_1_a = 1
7  f_3_a = f_2_a + f_1_a
8  f_2_b = 1
9  f_4 = f_3_a + f_2_b
10 n_3_b = 3
11 n_2_c = 2
12 f_2_c = 1
13 n_1_b = 1
14 f_1_b = 1
15 f_3_b = f_2_c + f_1_b
16 f_2_d = 1
17 f_5 = f_4 + f_3_b
```

Analýza rekurze - expression multi point (reprezentace)



Princip návratových hodnot

- fraktály
- divide & conquer
- procedurální generování
- devil's due

Cvičení

Napište program na procházení a výpis libovolně zanořené kolekce, využijte rekurzi.

Napište program, který vypíše n-tý člen tribonacciho posloupnosti. První tři prvky jsou 0, 1, 1 a každý další člen je součtem předchozích tří. Použijte rekurzi.

Napište program, který projde seznam pomocí rekurze.