Programovací paradigmata

P. Jakš M. Popďakunik

31. října 2022

České vysoké učení technické, Fakulta jaderná a fyzikálně inženýrská

Definice

Paradigma = vzor, model

Programovací paradigma nám udávají způsoby jakým řešit problémy za pomoci programovacích jazyků.

Neexistuje jednotná definice které techniky považovat za programovací paradigma a které ne

Definice

Programovací jazyky mohou dané paradigma podporovat, slabě podporovat nebo nepodporovat.[1]

Říkáme že jazyk paradigma slabě podporuje pokud jeho použití podporuje ale není to zcela přirozené.

[1] Robert W. Floyd. "The Paradigms of Programming". In: *Commun. ACM* 22.8 (Aug. 1979), pp. 455–460. ISSN: 0001-0782. DOI: 10.1145/359138.359140. URL: https://doi.org/10.1145/359138.359140.

Imperativní programování

Program je napsaný jako posloupnost příkazů a určuje přesný způsob jak úlohu řešit.

Imperativní programování se dá rozdělit hlavně na tyto způsoby:

- Objektové programování
- Procedurální programování

Imperativní programování - Historie

První programovací jazyky byly strojové jazyky.

Jejich nástupcem jsou jazyky symbolických adres (assembler).

Oba dva druhy jazyků se skládají z velmi jednoduchých instrukcí a většinou se [2] považují za imperativní jazyk.

[2] Existují vyjímky: v moderní době jsou to hlavně FPGA (programovatelné hradlové pole) nebo historicky například tzv. lisp-machines

Imperativní programování - Ukázka Assembleru

```
SECTION .data           ;
    extern printf       ;Načte proceduru
    global main         ;
fmt:                    ;
    db "%d", 10, 0      ;
SECTION .text           ;
main:                   ;Označuje začátek programu
    mov     eax, 14     ;Uloží 14 do registru eax
    mov     ebx, 10     ;Uloží 10 do registru eax
    add     eax, ebx    ;Uloží do eax součet eax a ebx
    push    eax         ;Pošle hodnotu eax na zásobník
    push    fmt         ;Pošle hodnotu fmt na zásobník
    call    printf      ;Zavolá proceduru printf
    mov     eax, 1      ;Uloží 1 do registru eax
    int     0x80        ;Přerušením ukončí program
```

Strukturované programování

Strukturované programování zakazuje používání příkazu goto. Místo toho mají mít řídící struktury jeden vstupní a výstupní bod[3].

[3] Toto není vždy pravda, například příkaz break a obsluha vyjímek

Procedurální programování

Toto paradigma vzniklo spolu se vznikem vyšších programovacích jazyků.

Základem paradigmatu je rozdělení programů do procedur/funkcí za cílem získání větší modularity.

Procedurální programovaní hodně zavisí na použití bloků a scope.

Mezi hlavní představitele tohoto paradigmatu se řadí:
C, Pascal, Fortran, COBOL

Objektově orientované programování

Dá se považovat za nástupce procedurálního programování.

V objektovém programování pracujeme s objekty, což jsou entity které obsahují data (atributy) a kód (metody)

Pro objekty platí následující:

- Abstrakce
- Zapouzdření
- Kompozice
- Dědičnost
- Polymorfismus

Objektově orientované programování

Programovací jazyky které umožňují objektové programování můžeme rozčlenit do 2 skupin:

- Čistě objektové jazyky
  - V těchto jazycích jsou všechny datové typy objekty a všechny operace se provadí za pomoci metod.
  - Například Smalltalk nebo Ruby
- Hybridní nebo Objektově orientované jazyky
  - V těchto jazycích většinou například primitivní datové typy nejsou objekty
  - Sem řadíme například C++, Rust, Java atd.

Objektově orientované programování - ukázka Ruby[4]

```
def pow(a,b)
    power=1
    for i in 1..b
        power=power*a
    end
    return power
end
puts "Enter Base:"
base=gets.chomp.to_i
puts "Enter exponent:"
expo=gets.chomp.to_i
puts "The power is #{pow(base,expo)}"
```

[4] https://www.includehelp.com/ruby/print-power-of-a-number.aspx

OOP modely

Existují 2 modely objektového programování:

1. Model využívající třídy
2. Model využívající prototypy

OOP - Model využívající třídy

V tomto modelu uživatel definuje třídy které poté instancuje.

Třídu můžeme považovat za návod jak třídu instancovat.

Tento model je používaný např v C++, Javě.

OOP - Model využívající prototypy

V tomto modelu existují pouze objekty.

Objektům přiřazujeme metody a atributy a můžeme od nich odvozovat další objekty. Rodičovský objekt od kterého odvozujeme nazýváme prototyp.

Jazyky, které tento model implementují jsou téměř vždy interpretované a používají dynamický typový systém.

Hlavním představitelem tohoto modelu je JavaScript, nebo popřípadě Lua.

OOP - Model využívající prototypy

Jak funguje dědění?

Existují 2 způsoby jak jazyky mohou dědění implementovat

1. Delegace
   - Odvozený objekt obsahuje tzv. delegační ukazatel (delegation pointer) na předka.
   - Pokud zavoláme metodu, objekt se ji nejdříve pokusí najít ve svých metodách, pokud ji neobsahuje tak se metodu pokusí najít postupně ve svých předcích.
2. Zřetězení (Concatenation)
   - Zřídka používaný přístup. Používá ho například jazyk Kevo
   - Objekty neobsahují delegační ukazatel, místo toho je prototyp zkopírován.
   - Hlavním rozdílem je, že změny v předcích se automaticky nepropagují do potomků

Objektově orientované programování - ukázka JavaScript [5]

```
let animal = {
  eats: true,
  walk() {alert("Animal walk");}};
let rabbit = {
  jumps: true,
  __proto__: animal};
let longEar = {
  earLength: 10,
  __proto__: rabbit};
longEar.walk(); // prints "Animal walk"
alert(longEar.jumps); // prints "true"
```

[5] https://javascript.info/prototype-inheritance

Deklarativní programování

- Definice se různí
  - např.: Deklarativní - když není imperativní
- Imperativní programování je založeno na *Turingových strojích*
  - Matematický koncept
  - Jednojádrový procesor, jehož registry nabývají konečný počet stavů = celkový stav
  - Operuje na konečném počtu slov = hodnot v buňkách paměti = instrukcích
- Deklarativní programování je založeno jiném modelu výpočtů

Funkcionální programování

- Založeno na *Lambda kalkulu*
  - Lambda kalkulus např. nemá datové typy či rekurzi
  - Ekvivalentní s Turingovými stroji
- Pro připomenutí: *Haskell, Elm*

Logické programování

- Založeno na formální logice
- Logický program je sada logických vět v jazyce dané domény a následné dotazy na tato pravidla
- Programovací jazyky:
  - *Prolog*
  - *ASP* - answer set programming
  - *Datalog*

Jazykově orientované programování (Language-oriented programming)

V tomto paradigmatu jsou jednotlivé jazyky položené na stejnou úroveň jako objekty nebo jiné komponenty.

Toto paradigma říká, že namísto toho aby programátor problémy řešil v obecných programovacích jazycích, tak programátor vytvoří takzvané doménově specifické jazyky (Domain-specific language) a problémy řeší za pomoci nich.

Doménově specifické jazyky

DSL jsou jazyky které jsou určené k řešení jednoho konkrétního problému.

Tyto jazyky většinou nejsou turingovsky kompletní. Ale vývojem se mohou turingovsky kompletní stát (Například Perl).

Doménově specifické jazyky

Jedním ze speciálních případů DSL jsou markup jazyky (HTML, LaTeX, Markdown)

Doménově specifické jazyky - Shader languagess

Jazyky jako GLSL (OpenGL Shading Language) určené pro psaní shaderů pro grafické karty.

Například GLSL používá upravenou syntaxi jazyka C. Příklad[6]:

```
#ifdef GL_ES
precision mediump float;
#endif

uniform float u_time;

void main() {
^^Ivec2 st = gl_FragCoord.xy/u_resolution;
^^Igl_FragColor = vec4(st.x,st.y,abs(sin(u_time)),1
}
```

[6] https://thebookofshaders.com/

Doménově specifické jazyky - query languages

Jazyky určené pro práci s databázemi, nejznámějším příkladem je SQL.

Příklad:

```
SELECT * FROM Customers
WHERE Country='Czech_Republic';
```

Děkujeme za pozornost