Paradygmaty programowania

Jacek Michałowski, Piotr Latanowicz

15 kwietnia 2014

Zadanie 1

Rachunek predykatów I rzędu stanowi podstawę teoretyczną dla

a) [**Nie**] języka Scheme

b) [**Tak**] języka Prolog

c) [**Nie**] języka Smalltalk

Programowanie logiczne to odmiana programowania deklaratywnego, w której program podawany jest jako pewien zestaw zależności, a obliczenia są dowodem pewnego twierdzenia w oparciu o te zależności. Rachunek predykatów I rzędu jest realizowany w programowaniu logicznym.

Scheme - język funkcyjny
Prolog - język logiczny
Smalltalk - język obiektowy (dynamicznie typowany)

Zadanie 2

Zadanie 2

Do cech wyróżniających języków imperatywnych należą:

a) [**Tak**] istnienie zmiennych wielokrotnego przypisania

b) [**Nie**] wykorzystanie wyrażeń warunkowych

c) [**Tak**] wykorzystanie instrukcji iteracyjnych

Zadanie 2

Wspólne elementy dla języków imperatywnych:

- Instrukcje przypisania wykonują pewne zadanie na zlokalizowanych w pamięci danych i odkładają tam wynik działania na potrzeby późniejszych operacji.
- Instrukcje pętli umożliwiają wielokrotne wykonanie tego samego kodu - w zależności od potrzeb, "wielokrotność"może oznaczać pewną określoną z góry liczbę powtórzeń lub wykonywanie do czasu spełnienia pewnych warunków.
- Instrukcje warunkowe wykonują pewien blok kodu tylko wtedy, kiedy spełniony jest określony warunek. W przeciwnym razie blok ten jest pomijany podczas wykonywania.

Jednak wyrażenia warunkowe występują także w innych językach, stąd w b prawidłową odpowiedzią jest **Nie**.

Zadanie 3

Funkcje wyższego rzędu

a) [**Tak**] to funkcje, których argumentami i/lub wartością są funkcje

b) [**Nie**] to funkcje, które mogą działać na nie określonych z góry typach danych

c) [**Nie**] są typowym elementem wszystkich języków deklaratywnych

Funkcja wyższego rzędu - jest to funkcja, która zwraca lub przyjmuje jako argument inne funkcje. Powszechnie stosowane w językach funkcyjnych, jak SML, Ocaml, Haskell, jak również w Pythonie i Javascripcie. Żeby możliwe było tworzenie funkcji wyższego rzędu, funkcje muszą być typem pierwszoklasowym. Takie funkcje służą często do dostarczenia generycznych realizacji algorytmów (programista parametryzuje je własnymi funkcjami).

Zadanie 4

Zadanie 4

Język Java jest językiem

a) [**Nie**] dynamicznym

b) [**Tak**] imperatywnym

c) [**Tak**] statycznie typowanym

Zadanie 4

Język dynamiczny - język programowania wysokiego poziomu, który podczas działania programu wykonuje wiele operacji przeprowadzanych w innych językach na etapie kompilacji. Np. dodawanie nowego kodu, przez rozszerzanie obiektów i definicji, przez zmianę typów danych. Zachowania takie można emulować w niemal wszystkich językach programowania o wystarczającej złożoności, jednak języki dynamiczne mają wbudowane konstrukcje umożliwiające ich bezpośrednie wykorzystanie.

Java nie jest językiem dynamicznym (na przykład dlatego że jest kompilowany), jest językiem imperatywnym (patrz: pytanie 2), a także jest statycznie typowany (kontrola typów na etapie kompilacji).

Zadanie 5

Regułami nazywa się w programie prologowym klauzule postaci:

a) [**Nie**] $A_1 \wedge A_2 \wedge \ldots \wedge A_n \leftarrow B$

b) [**Tak**] $A \leftarrow B_1 \wedge B_2 \wedge \ldots \wedge B_n$

c) [**Nie**] $\leftarrow B_1 \wedge B_2 \wedge \ldots \wedge B_n$

W języku Prolog wszystkie klauzule muszą być klauzulami Horna, czyli w formie $A_1 \wedge A_2 \wedge A_3 \rightarrow B$, tylko implikacja jest zapisana w odwrotną stronę (od prawej do lewej). Jak widać w a to B implikuje koniunkcję A_1, A_2 itd. zatem nie jest to klauzula Horna. W b wszystko jest w porządku. W c nie ma nic z drugiej strony implikacji, więc również nie jest to klauzula Horna.

Zadanie 6

Wartość λ-wyrażenia $\lambda x.x + 1$ jest

a) [**Nie**] liczbą o 1 większa od x

b) [**Tak**] funkcją jednoargumentową

c) [**Nie**] funkcją rekurencyjną

Zadanie 6

W rachunku lambda każde wyrażenie określa funkcję jednoargumentową. Z kolei argumentem tej funkcji jest również funkcja jednoargumentowa, wartością funkcji jest znów funkcja jednoargumentowa. Funkcja jest definiowana anonimowo przez wyrażenie lambda, które opisuje, co funkcja robi ze swoim argumentem.

Rachunek lambda z pozoru nie ma żadnego mechanizmu, który umożliwiałby rekurencję – nie można odwoływać się w definicji do samej funkcji. Rekurencję można osiągnąć poprzez operator paradoksalny Y.

Czyli wartość nie może być liczbą o 1 większą od x (a), bo jest funkcją. Jest funkcją jednoargumentową (b), nie jest funkcją rekurencyjną (c), ponieważ w rachunku lambda nie można odwoływać się w definicji do samej funkcji.

Zadanie 7

Niech `f(x, y)` będzie w pewnym języku zdefiniowana jako

```
{ if y>0 then x + "iks" else y }
```

Aby móc stwierdzić, że wywołanie `f(1, 0)` nie spowoduje błędu na żadnym etapie, wystarczy wiedzieć, że

a) [**Nie**] jest to język z przeładowaniem operatorów

b) [**Nie**] jest to język oferujący funkcje polimorficzne

c) [**Tak**] jest to dynamicznie typowany język interpretowany

Przeładowanie operatora "+" nic nie da, jeżeli język wymaga, aby dodawać do siebie zmienne tego samego typu (a tutaj próbuje dodać liczbę do ciągu znaków). To czy język umożliwia stosowanie funkcji polimorficznych (czyli metod wirtualnych) nie ma tutaj żadnego znaczenia. Jeśli jest to język dynamicznie typowany (np. Perl) to nie będzie błędu.

Zadanie 7

Dziękujemy za uwagę