Paradygmaty programowania

Mirosław Głowacki

Wykład w językach C++, Haskell i PROLOG

Literatura

- R. Sebesta, *Concepts of Programming Languages*, Addison Wesley, 2005
- P. Van Roy, S. Haridi, *Concepts, Techniques, and Models of Computer Programming*, MIT Press, 2004
- J. Reynolds, *Theories of Programming Languages*, Cambridge University Press, 1998
- R. Sebesta, *Concepts of Programming Languages*, Addison Wesley, 2005
- B. Stroustrup, Język C++. Kompendium wiedzy, Wyd. IV, Helion 2014
- B. Meyer, Programowanie zorientowane obiektowo, Helion, Gliwice 2005
- J. Grębosz, Symfonia C++, Oficyna Kallimach, Kraków 2006
- R. Bird, *Introduction to Functional Programming using Haskell*, Prentice Hall, 1988
- M. Lipovaca, *Learn You a Haskell for Great Good!*, William Pollock, San Francisco 2011
- F. Kluźniak, S. Szpakowicz, *Prolog*, Wydawnictwa Naukowo-Techniczne, 1983
- U. Nilsson, J. Małuszyński, *Logic, Programming and Prolog*, John Wiley & Sons, 1995

Paradygmaty programowania

Mirosław Głowacki
Wydz. Inżynierii Metali i Informatyki Przemysłowej AGH

Spis treści

Paradygmaty

- Powinniśmy zapewne zacząć od wyjaśnienia, o czym będzie mowa w niniejszym wykładzie.
- Istnieje pojęcie paradygmaty programowania i na początek warto przyjrzeć się znaczeniu słowa „paradygmat”, często nadużywanemu przez filozofów, lingwistów i informatyków
- Otóż, jak podaje *Słownik języka polskiego PWN*, paradygmat to:

  przyjęty sposób widzenia rzeczywistości w danej dziedzinie, doktrynie itp.

  lub

  zespół form fleksyjnych (deklinacyjnych lub koniugacyjnych), właściwy danemu typowi wyrazów; wzorzec, model deklinacyjny lub koniugacyjny.

Paradygmaty programowania

- Jak te definicje mają się do programowania?
- Trudno orzec; sięgnijmy jeszcze do greckich korzeni słowa. Greckie παράδειγμα oznacza wzorzec bądź przykład.
- Czyżby chodziło więc o typowy, wzorcowy sposób pisania programów?
- Niezupełnie chodzi raczej o zbiór mechanizmów, jakich programista używa, pisząc program, i o to, jak ów program jest następnie wykonywany przez komputer.

Paradygmaty programowania

- Zatem paradygmat programowania to

  ogół oczekiwań programisty wobec języka programowania i komputera, na którym będzie działał program.
- Przyjrzyjmy się czterem zasadniczym przykładom.
- Przykłady te, obejmujące najbardziej powszechne paradygmaty programowania.

Przykład pierwszy programowanie imperatywne

- Programowanie imperatywne to najbardziej pierwotny sposób programowania, w którym program postrzegany jest jako ciąg poleceń dla komputera
- Ściślej, obliczenia rozumiemy tu jako sekwencję poleceń zmieniających krok po kroku stan maszyny, aż do uzyskania oczekiwanego wyniku.
- Stan maszyny należy z kolei rozumieć jako zawartość całej pamięci oraz rejestrów i znaczników procesora.
- Ten sposób patrzenia na programy związany jest ściśle z budową sprzętu komputerowego o architekturze von Neumanna

Przykład pierwszy programowanie imperatywne

- Poszczególne instrukcje (w kodzie maszynowym) to właśnie polecenia zmieniające ów globalny stan.
- Języki wysokiego poziomu — takie jak Fortran, Algol, Pascal, Ada lub C — posługują się pewnymi abstrakcjami, ale wciąż odpowiadają paradygmatowi programowania imperatywnego.
- Przykładowo, instrukcje podstawienia działają na danych pobranych z pamięci i umieszczają wynik w tejże pamięci.
- Abstrakcją komórek pamięci są zmienne.

Przykład pierwszy programowanie imperatywne

- Przykładowy program w języku imperatywnym (Pascal)

```
program silnia;
 var i, n, s: integer;
 begin read(n);
 s := 1;
 for i := 2 to n do
 s := s * i;
write (s)
end.
```

Przykład drugi programowanie funkcyjne

- Tutaj program to po prostu złożona funkcja (w sensie matematycznym), która otrzymawszy dane wejściowe wylicza pewien wynik
- Zasadniczą różnicą w stosunku do poprzednich paradygmatów jest brak stanu maszyny: nie ma zmiennych, a co za tym idzie nie ma żadnych efektów ubocznych.
- Nie ma też imperatywnych z natury, tradycyjnie rozumianych pętli (te wymagają np. zmiennych do sterowania ich przebiegiem).
- Konstruowanie programów to składanie funkcji, zazwyczaj z istotnym wykorzystaniem rekurencji.
- Charakterystyczne jest również definiowanie funkcji wyższego rzędu, czyli takich, dla których argumentami i których wynikami mogą być funkcje (a nie tylko „proste” dane jak liczby lub napisy).

Przykład drugi programowanie funkcyjne

- Przykładowy program (definicja funkcji) w pierwszym dostępnym języku funkcyjnym Haskell

```
factorial :: (Integral a) => a -> a
factorial 0 = 1
factorial n = n * factorial (n - 1)
```

Przykład drugi programowanie funkcyjne

- Obliczenie silni danej liczby: we współczesnym języku Haskell

```
quicksort :: (Ord a) => [a] -> [a]
quicksort [] = []
quicksort (x:xs) =
    let smallerSorted = quicksort [a | a <- xs, a <= x]
        biggerSorted = quicksort [a | a <- xs, a > x]
    in  smallerSorted ++ [x] ++ biggerSorted
```

Przykład trzeci programowanie w logice (programowanie logiczne)

- Na program składa się zbiór zależności (przesłanki) i pewne stwierdzenie (cel)
- Wykonanie programu to próba udowodnienia celu w oparciu o podane przesłanki.
- Obliczenia wykonywane są niejako „przy okazji” dowodzenia celu.
- Podobnie jak w programowaniu funkcyjnym, nie „wydajemy rozkazów”, a jedynie opisujemy, co wiemy i co chcemy uzyskać.

Przykład trzeci programowanie w logice (programowanie logiczne)

- Przykładowy program w języku logicznym PROLOG

```
ojciec(jan, jerzy).
ojciec(jerzy, janusz).
ojciec(jerzy, józef).
dziadek(X, Z) :- ojciec(X, Y), ojciec(Y, Z).
?- dziadek(X, janusz).
```

- `:-` oznacza „jeśli” lub „wtedy gdy”
- `?-` oznacza „czy”
- stałe: rozpoczynają się z małej litery, a zmienne z dużej

Przykład czwarty programowanie obiektowe

- W programowaniu obiektowym program to zbiór porozumiewających się ze sobą obiektów, czyli jednostek zawierających pewne dane i umiejących wykonywać na nich pewne operacje
- Dane składowe klasy obiektów stanowią o stanie obiektów będących ich instancjami, a funkcje składowe klasy umożliwiają zmianę tego stanu
- Ważną cechą jest tu powiązanie danych (czyli stanu) z operacjami na nich (czyli poleceniami) w całość, stanowiącą odrębną jednostkę — obiekt.

Przykład czwarty programowanie obiektowe

- Cechą nie mniej ważną jest mechanizm dziedziczenia, czyli możliwość definiowania nowych, bardziej złożonych obiektów, na bazie obiektów już istniejących.
- Zwolennicy programowania obiektowego uważają, że ten paradygmat dobrze odzwierciedla sposób, w jaki ludzie myślą o świecie
- Nawet jeśli pogląd ten uznamy za przejaw pewnej egzaltacji, to niewątpliwie programowanie obiektowe zdobyło ogromną popularność i wypada je uznać za paradygmat obecnie dominujący.

Przykład czwarty programowanie obiektowe

Przykładowy fragment programu w języku obiektowym (C++)

```
class macierz {
    static int nrmac;
    static int maxmac;
    int n, m;
    double* mac;
public:
    macierz(int = 1, int = 1, double = 0);
    macierz operator+(macierz);
    ~macierz();
    void piszmac();
    int sizeofmac(){return n*m;};
};
```

Paradygmaty a języki programowania

- Konkretny język programowania ucieleśnia jeden lub więcej paradygmatów
- Fortran, Pascal i C to języki pozwalające stosować paradygmat programowania imperatywnego. Mówi się wręcz, że są to języki imperatywne.
- Java bądź C# to z kolei języki obiektowe, w których typowe programowanie imperatywne zostało mocno ograniczone.
- Natomiast C++ jest językiem zarówno obiektowym, jak i imperatywnym.
- Do pewnego stopnia można zresztą uznać, że programowanie imperatywne to szczególny, wynaturzony przypadek programowania obiektowego, gdzie wszystko rozgrywa się wewnątrz jednego „superobiektu”.