Міністерство освіти і науки України
Національний авіаційний університет
Навчально-науковий інститут комп'ютерних інформаційних технологій
Кафедра комп'ютеризованих систем управління

Конспект
з дисципліни «Функціональне і логічне програмування»

Виконав:
студент ННІКІТ
групи СП-325
Клокун В. Д.

Київ 2019

1. П'ятниця 8 лютого 2019 р.

В логічній програмі алгоритм розв'язання задачі не описується. Описується умова задачі, а саме вказується, які є об'єкти предметної області, які відношення між ними, а також описується мета, якої необхідно досягти. Основою опису є відношення між об'єктами.

Логічна програма складається з двох типів висловлювань:

1. Факти, які описують властивості об'єктів.
2. Правила, які описують відношення між об'єктами.

Використовуючи факти і правила, наведені в програмі, можна отримати або вивести з наявної інформації нові факти або правила. Крім цього, в логічній програмі вказується мета пошуку. Тобто логічна програма є базою даних про умову задачі. Розв'язання задачі полягає у пошуку способу задоволення заданої мети на основі наявних фактів і правил. Алгоритм цього пошуку реалізований в інтерпретаторі або компіляторі мови програмування. Найбільш поширеною мовою логічного програмування є Prolog, у якого є численні спадкоємці.

Функційне програмування. Єдина управляюча конструкція в мові функційного програмування — це виклик функції. У функційній мові задана деяка множина базових функцій, а решта функцій будується з них за допомогою композиції. Наприклад, ми можемо мати $\max(a, b)$, тоді $\max(a, b, c) = \max(a, \max(b, c))$. Найбільш відомі мови функційного програмування: Lisp, Haskell, Scheme, Standard ML.

Концептуальне програмування. Це автоматизований синтез програм на основі формальних специфікацій задачі. Специфікація задачі — це її формальний опис: блок-схема, словесний опис, словесний опис і так далі.

Імперативне і декларативне програмування.

2. Понеділок 11 лютого 2019 р.

2.1. Мова Prolog. Терм

Терм — це об'єкт даних, може бути константою змінною або структурою (складним термом). Константи в Пролозі поділяються на 2 види:

1. Константи-числа, які позначають числові значення.
2. Константи-атоми, які позначають елементарні об'єкти предметної області.

Числа бувають цілі і дійсні (integer, real).

Атом — будь-яка послідовність символів, взятих у подвійні лапки — «""». В більшості випадків атом іменує об'єкт предметної області. Наприклад, атом може позначати людину, тварину або предмет. Якщо за контекстом програми атом можна відрізнити від інших об'єктів, наприклад, від імен змінних, то лапки можна не використовувати. Але це накладає свої обмеження: не можно ставити пробіли, спеціальні символи. Всі інші типи даних в Пролозі складаються із сполучень констант і змінних.

2.2. Змінні

Змінна в Пролозі може набувати значення константи, тобто атому або числа. Імена змінних в програмі починаються з великої літери або символу підкреслення. Якщо ім'я об'єкта в програмі починається з великої літери, вважається, що це змінна. `X` — змінна, x — атом, незмінна величина.

В лапки можна включати спеціальні символи за допомогою «\». Існує спеціальна змінна, яка називається анонімною і позначається символом підкреслення «_». Вона успішно зіставляється з будь-яким значенням, але не пов'язується з ним, тобто отримати з неї будь-яке значення неможливо. Її використовують в програмах для заповнення місця. В тому разі, якщо це індекси в програмі допомогає вказати аргумент, але значення, з яким пов'язаний аргумент, неважливе для програміста.

2.3. Складні терми (структури)

Запис структур в програмах на Пролозі виглядає подібно до записів у мовах C або Pascal. В Пролозі кажуть, що структура складається з головного функтора та компонентів. Головним функтором називається ім'я функції (атом), а компонентами — те, що стоїть в дужках. Компоненти структури наводять в дужках і через кому: `dog(rex), father(ivan, anton)`.

Арність — кількість компонентів структури. `dog/1, father/2` або `father(i, o)` — позначення арності, де `i` — input, `o` — output.

В Пролозі використовується 2 види коментарів:

1. Блочні — `/* comment text */`.
2. Строчні — `% comment text`.

2.4. Розділи програми

Програма на Пролозі може включати такі розділи:

1. Domains — оголошуються і описуються домени — типи, визначені програмістом. Тут треба оголошувати структури.

```
domains
```

2. Predicates.

```
predicates
  dog(symbol)
  father(symbol, symbol)
```

3. Clauses. Тут розміщують факти і правила, що описують зміст програми.
4. Goal. Ціль програми.
5. Database — база знань, динамічні типи даних.

Програма на Пролозі — це сукупність тверджень або висловлювань (речень, клаузів), які за своєю суттю близькі до алгебри логіки. Сукупність тверджень утворює базу даних (базу знань) програми. Це статична база даних, тобто вона не змінюється під час роботи програми. Також може використовуватись динамічна база даних, твердження якої можуть доповнюватись або видалятись з неї під час роботи програми.

Твердження в розділі clauses бувають 2 типів:

1. Факти.
2. Правила.

Кожне твердження (факт або правило) закінчується крапкою. В ході виконання (або розгляду) програми інтерпретатор Прологу виконує доведення цілей, які входять у ці твердження.

Факт — це одиночна ціль, яка вважається за визначенням істинною. Він є описом заданих відомостей про об'єкти предметної області. Наприклад:

```
dog(rex).
father(ivan, petro).
```

Правила дозволяють вивести з наявних фактів деякий інший факт. Правила складаються із голови і хвоста. Наприклад:

```
animal(X) if dog(X).

animal(X) :- dog(X).
```

Голова правила істинна, якщо істинні всі цілі, вказані у хвості.

Змінні у Пролозі не оголошуються. Відповідно тип змінної в програмі на Пролозі ніяк не вказується. Більш того, одна і та ж змінна в різних реченнях може набувати значення різних типів. Але про тип змінної, використаної

в конкретному випадку, можна дізнатись із розділу `predicates`, де вказується тип компоненту структури.

В Пролозі головними сутностями вважаються не об'єкти предметної області, а відношення між ними. Об'єкти, які вступають в ці відношення, розглядаються як другорядні сутності, порівняно з самими відношеннями. З цього підходу випливають такі наслідки:

1. Типи аргументів приписують не до змінних, які представляють об'єкти, а до відношень, в яких ці об'єкти використовуються.
2. Замість секції оголошення змінних, яка входить до імперативної програми, в Пролог-програмі присутня секція оголошень відношень — `predicates`
3. Областю дії змінної є одне речення. В рамках одного речення одне і те ж ім'я змінної означає одну і ту ж змінну. Однак, в іншому твердженні змінна з тим же ім'ям (X) може мати інший сенс і навіть містити значення іншого типу.

Написання програми на Пролозі полягає в тому, що програміст описує об'єкти предметної області, описує відношення між ними і визначає правила, які описують умови, за яких ці відношення дійсні.

Співставлення. Це процес, на вхід якого подаємо 2 терми. Результатом співставлення є висновок, що ці 2 терми співставимі, тобто відповідні один одному або неспівставимі, тобто невідповідні. Якщо виявилось, що терми співставимі, то кажуть, що співставлення завершилось успіхом. Якщо неспівставимі, то кажуть, що процес співставлення завершилось невдачею — `fail`.

Правила співставлення:

1. Якщо 2 об'єкти — константи або конкретизовані змінні, то вони співставимі, якщо їх значення рівні.
2. Якщо один із об'єктів не конкретизована змінна, то співставлення можливе. При цьому якщо другий об'єкт — константа або конкретизована змінна, то першу неконкретизовану змінну конкретизує значення другого об'єкта. Якщо другий об'єкт теж неконкретизована змінна, то 2 змінні пов'язуються між собою, тимчасово ототожнюються. Після цього, якщо одна із них буде конкретизована певним значенням, то таку ж конкретизацію отримає друга пов'язана з нею змінна.
3. Якщо 2 об'єкти — структури, то вони співставимі, якщо вони, по-перше, мають однакові головні функтори, і, по-друге, всі їх компоненти співставимі.