МУЛЬТИПАРАДИГМЕННЕ ПРОГРАМУВАННЯ

Лекція 1

ВВЕДЕННЯ ДО МУЛЬТИПАРАДИГМЕННОГО ПРОГРАМУВАННЯ

Мультипарадигменное программирование — программирование с одновременным использованием множества парадигм.

Основные подходы к организации мультипарадигменного программирования:

- создание нового языка программирования,
- расширение существующего языка программирования,
- встраиваемые интерпретаторы,
- расширяемые интерпретаторы,
- трансляция из одного языка в другой,
- сборка модулей, написанных на разных языках программирования,
- библиотечное расширение существующего языка программирования.

Мультипарадигмальна мова програмування — як правило, мова програмування, розроблена спеціально як інструмент мультипарадигмального програмування, тобто образотворчі можливості якого спочатку передбачалося успадкувати від декількох, найчастіше неспоріднених мов.

Іноді термін **мультипарадигмальна мова програмування** визначають як «мову, який підтримує більше ніж одну парадигму програмування. Таке визначення є недостатньо точним, бо саме поняття парадигми програмування різні автори визначають по-різному. Наприклад, якщо вважати парадигмами програмування рекурсію, структурне програмування і присвоювання, то виявиться, що під це визначення підійдуть чи мало не всі існуючі мови програмування, за винятком деяких особливих випадків (наприклад, мови Haskell, де немає присвоювання в звичному вигляді).

Мета розробки мультипарадигмальних мов програмування складається, як правило, у тому, щоб дозволити програмістам використовувати кращий інструмент для роботи, визнаючи, що ніяка парадигма не вирішує всі проблеми найлегшим або найбільш ефективним способом.

Один з найбільш амбітних прикладів — Oz, який є логічною, функціональною, об'єктно-орієнтованою, мовою конкурентного (паралельного) програмування тощо. Oz розроблено за десять років, її мета — об'єднати поняття, які традиційно пов'язані з різними програмними парадигмами.

Як одну з найбільш успішних мультипарадигмальних мов програмування часто називають мову C++.

Мультипарадигмальні мови

Узагальнене програмування (англ. *generic programming*) — парадигма програмування, що полягає в такому описі даних і алгоритмів, який можна застосовувати до різних типів даних, не змінюючи сам опис. У тому чи іншому вигляді підтримується різними мовами програмування.

Можливості узагальненого програмування вперше з'явилися в 1970-х роках у мовах CLU та Ada, а потім у багатьох об'єктно-орієнтованих мовах, таких як C++, Java, D і мовах для платформи .NET.

Термін ***"Узагальнене програмування"*** вперше було введене Девідом Массером і Олександром Степановим (1989) , які описували парадигму програмування, яка заснована на тому, що типи даних і структури даних є абстрактними і не впливають на конкретну реалізацію алгоритмів, а загальні функції реалізовані з використанням узагальнених формалізованих типів.

Приклади мультипарадигмальных мов програмування, розділених за кількістю парадигм, що підтримуються:

Дві парадигми

- Функціональна, об'єктно-орієнтована:
  - Dylan.
- Функціональна, процедурна:
  - APL.
- Функціональна, логічна:
  - AFL;
  - Curry;
  - Mercury.

Dylan — динамічна об'єктно-орієнтована мова програмування, націлена на швидку розробку програм; розроблений насамперед зусиллями Apple.

При необхідності, пізніше можна оптимізувати програми введенням інформації про типи. Dylan підтримує множинну спадковість, поліморфізм і багато інших парадигм. Мова загального призначення, придатна як для прикладного, так і для системного програмування. Включає в себе збирання сміття, перевірки в ході виконання, відновлення після помилок і модульну систему.

Ім'я мови Dylan означає «**DY**namic **LAN**guage».

Ця мова народилася в Apple на початку 1990 р. ЇЇ розробники хотіли створити покращений гібрид з елегантного варіанту LISP — Scheme, системи ООП CLOS від потужного промислового варіанту LISP — Common Lisp та ідеями з Smalltalk — і все це з нормальною загальноприйнятою системою позначень алголо/паскале-подібного синтаксису. Незабаром після цього аналогічний проект був запущений в Університеті Карнегі-Меллон — над створенням компілятора Dylan працювала знаменита команда Карнегі-Меллон з реалізації CMU Common Lisp. Іншу, комерційну версію з повною IDE випустила компанія Harlequin.

APL (вимовляють «ей-пі-ель», названа за книгою *A Programming Language*) — інтерактивна[en] масиво-орієнтована[en] мова програмування та інтегроване середовище розробки, що доступні від низки розробників і для більшості комп'ютерних платформ. Вона ґрунтується на математичній нотації, винайденій Кеннетом Айверсоном і його колегами, що пропонує спеціальні засоби для проектування і розробки цифрових обчислювальних систем, як апаратного забезпечення, так і програм.

APL має поєднання унікальних і порівняно рідкісних функцій, які привертають увагу програмістів і роблять її плідною мовою програмування:

- Вона лаконічна, використовує символи, а не слова і застосовує функції до всіх масивів без використання явних циклів.
- Абстрактна, орієнтована на розв'язання задач, орієнтована на написання програм незалежних від архітектури комп'ютера або операційної системи.
- Має одне просте, послідовне і рекурсивне правило пріоритету: правий аргумент функції — це результат всього виразу праворуч від неї.
- Це полегшує розв'язання проблем на високому рівні абстракції.

Перше втілення того, що пізніше перетворилося на мову програмування APL, було опубліковане і формалізоване в *A Programming Language*, книзі, що описує нотацію винайдену 1957 року Кеннетом Е. Айверсоном в Гарвардському університеті. Айверсон розробив математичну нотацію для роботи з масивами, якої він навчав своїх учнів.

1960 року він почав працювати на IBM, і, працюючи з Адіном Фалкофом, створив APL на основі своєї нотації. Вона була використана всередині IBM для коротких дослідних звітів на комп'ютерних системах, таких як Burroughs B5000 і його стековому механізмі, коли стекові машини оцінювалися порівняно з регістровими машинами IBM з метою розробки майбутніх комп'ютерів.

Крім того, 1960 року Айверсон уже використовував свою нотацію в чернетках 6-ї глави, що називалася «Мова програмування» для книги, яку він писав з Фредом Бруксом, *Automatic Data Processing*, яка потім буде опублікована 1963 року.

1962 року відома перша спроба використати нотацію для стандартизації набору інструкцій для машин, які пізніше стали сімейством IBM System/360.

1963 року д-р Герберт Хеллерман, що працював в науково-дослідному інституті IBM Systems, реалізував частину позначень на комп'ютері IBM 1620, і він був використаний студентами в спеціальному курсі середньої школи для розрахунків трансцендентних функцій підсумовуванням рядів. Студенти випробували свій код в трансляторі доктора Хеллермана. Цю реалізацію частини позначень називають PAT (Personalized Array Translator).

1963 року Фалькоф, Айверсон, та Едвард Сассенгут, що на той час працювали на IBM, використали нотацію для формального опису архітектури і функціональності серії машин IBM System/360, що зрештою втілилося в статті, опублікованій в *IBM Systems Journal* 1964 року. Після публікації команда звернула свою увагу на втілення нотації в комп'ютерній системі. Одним з мотивів для цього фокусу на реалізації був інтерес з боку John L. Lawrence, який мав нові обов'язки в Science Research Associates, освітній компанії, купленій IBM 1964 року. Лоуренс попрохав Айверсона і його групу, щоб вони допомогли із використанням мови як інструменту для розробки та використання комп'ютерів в освіті.

Після того, як Lawrence M. Breed і Philip S. Abrams зі Стенфордського університету приєднались до команди IBM Research, вони продовжували свої попередні роботи з реалізації запрограмованих в FORTRAN IV частини нотацій, що було зроблено для IBM 7090 під управлінням операційної системи IBSYS. Ця робота була закінчена в кінці 1965 року і пізніше стала відома як IVSYS (Iverson System, система Айверсона).

Основи цієї реалізації були докладно описані Abrams в *Stanford University Technical Report, «An Interpreter for Iverson Notation»* in 1966.[18], Як і система PAT Геллермана раніше, ця реалізація не включала набір символів APL, а використовувала спеціальні зарезервовані слова англійською для функцій і операторів. Система була пізніше адаптована для системи з розділенням часу і, в листопаді 1966 року, була перепрограмована для комп'ютерів IBM/360 Model 50, що працювали в режимі розділення часу, і далі була використана всередині IBM.

Curry (Kappi) - вбудована мова проґрамування загального призначення. У Curry об'єднані дві парадигми декларативного проґрамування - функціональна і логічна. Більш того, в цій мові використані найбільш важливі операційні принципи подібних декларативних мов. Названа на честь американського ученого Гаскелла Каррі.

Мова Каррі поєднує в собі можливості функціонального програмування (вкладені вирази, функції вищого порядку, ліниві обчислення), логічного програмування (логічні змінні, часткові структури даних, вбудована система пошуку) і методів проґрамування для паралельних систем (паралельне обчислення виразів з синхронізацією). Більше того мова Каррі надає додаткові механізми в порівнянні з чистими мовами проґрамування (у порівнянні з функціональними мовами - пошук і обчислення за неповними даними, в порівнянні з логічними мовами - більш ефективний механізм обчислень завдяки детермінізму і викликом за необхідністю для функцій).

Mercury — язык функционально-логического программирования со строгой типизацией, призванный решить следующие две проблемы, которые возникают при использовании классического языка логического программирования Prolog:

1. проблема производительности. Современные реализации языков логического программирования по производительности уступают реализациям языков программирования императивного типа.
2. проблема отладки. Реализации языков логического программирования осуществляют меньше проверок во время компиляции, чем реализации языков программирования императивного типа. Это вынуждает программиста находить ошибки самому и без какой-либо существенной помощи со стороны отладчика.

Язык разработан в Мельбурнском университете. Первую версию выпустили Fergus Henderson, Thomas Conway и Zoltan Somogyi 8 апреля 1995 года.

Синтаксис Mercury частично унаследован от Пролога, система типов похожа на Haskell. Это чисто декларативный язык, разработчики полностью убрали из него все императивные возможности, что позволило улучшить встроенные в компилятор возможности оптимизации. Название Mercury дано в честь бога скорости Меркурия и отражает направленность на получение быстродействующих программ. Операции, при реализации которых обычно отказываются от чисто декларативного подхода, такие как ввод-вывод, выражаются в Mercury с помощью декларативных конструкций, используя линейные типы.

Три парадигми

- Функціональна, процедурна, об'єктно-орієнтована:
  - Perl; (з версії 5)
  - Python;
  - JavaScript;
  - Tcl;
  - PHP; (з версії PHP 5.3 частково підтримується функціональне програмування)
- узагальнена, процедурна, об'єктно-орієнтована:
  - C++;
  - D.

Чотири парадигми

- Функціональна, узагальнена, процедурна, об'єктно-орієнтована:
  - OCaml.
  - Common Lisp;
- Функціональна, процедурна, об'єктно-орієнтована, конкурентне:
  - Рубі.
- Об'єктно-орієнтована, узагальнена, процедурна, аспектно-орієнтована:
  - Java.

Шість парадигм

- Об'єктно-орієнтоване програмування, узагальнене програмування, процедурне програмування, функціональне програмування, подієво-орієнтоване програмування, рефлексивне програмування:
    - C#.

Сім парадигм

- логічна, програмування з обмеженнями, функціональна (як ледачі, так і «енергійні» обчислення), процедурна (імперативна), об'єктно-орієнтована, розподілена, паралельна
    - Oz