УДК 681.3.06

Бойко Б. І.

СТРУКТУРА ІМЕНУВАНЬ У ПРОГРАМУВАННІ

Стаття присвячена семантиці структурованих імен у мовах програмування. Наведені приклади структурування даних та порівняння особливостей реалізації в різних системах програмування.

Слово «парадигма», маючи в інформатиці та програмуванні вузько професійний зміст, що зближує їх з лінгвістикою, водночас виступає і як характеристика взаємодії духовного та реального світу, і як інструмент опису фактів, подій, явищ і процесів, що, можливо, не існують одночасно, але поєднані певною спільністю понять,

Традиційна імперативно-процедурна парадигма програмування набула популярності кілька десятиліть тому у сфері вузькопрофесійної діяль-

ності фахівців з організації обчислювальних та інформаційних процесів. Останні десятиліття різко розширили межі застосування інформатики, впровадивши її навіть у сферу масового спілкування і дозвілля. Це змінює пріоритети у доборі засобів і методів обробки інформації та вимоги у процесі проектування та експлуатації інформаційних систем. Сформовані на початку епохи комп'ютерного програмування парадигми теоретичного та прикладного (процедурно-функціонального) програмування до цього часу мають найбільш стійкий характер [1]. У міру зростання складності розв'язуваних задач відбулося розшарування засобів і методів програмування залежно від загальності і глибини опрацювання технічних деталей організації процесів комп'ютерної обробки інформації. Локалізувалася значна кількість використовуваних мов програмування (МП).

Щоб означити екстенсіонал поняття «мова програмування» (тобто множину об'єктів, охоплюваних цим поняттям), достатньо явно перелічити ті конкретні мови, що їх усі упевнено вважають МП. Це Фортран, Паскаль, Бейсик, Лісп, Пролог, Форт, Перл, Java. Однак навряд чи можна не включати до цього списку мови машин Тюринга, алгоритмів Маркова. А стрімке розширення сфери застосування автоматизованих систем диктує необхідність долучати до нього ще й формалізовані фрагменти природних мов та інтерфейси! засоби візуальних середовищ. Таке екстенсіональне означення практично виключає можливість аналізу нових окремих випадків, що в означенні не перераховані. Воно не дає повного уявлення про весь спектр мов програмування, бо не спирається на істотні властивості цих мов,- тому таке означення повинне бути інтенсіональним.

Наведемо одне з можливих інтенсіональних означень МП. Мова програмування - це інструмент для планування поведінки автоматизованої системи (виконавця).

Разом з тим МП має виконувати ще й комунікативну функцію у спілкуванні людей чи людей із автоматизованими системами (виконавцями). Люди можуть обмінюватися думками не інакше як за допомогою мовних засобів, нехай це вимовлені чи написані слова й речення, сигнали чи інші обумовлені знаки; думки також можуть зберігатися, накопичуватися тільки за допомогою мовних засобів. Тому можна вважати, що мова програмування - це знакова система для складання плану дій користувача автоматизованої системи у процесі розв'язання ним задач (планування поведінки автоматизованих систем).

Мовою в широкому розумінні називають будь-яку знакову систему, що виконує функції формування, збереження й передачі інформації і виступає засобом спілкування між людьми. Під знаком розуміють будь-який (матеріальний) об'єкт (процес, явище), що слугує представником деякого іншого об'єкта (процесу, явища). Таким чином, знак щось «позначає», і залежно від ступеня близькості до того, що позначається, виділяють різні типи знакових систем.

Найбільш близькими до позначуваного є так звані знаки-індекси (природні знаки). Вони настільки близькі до нього, що є ніби його частиною. Природний знак свідчить про це ціле чи якісь частини цього цілого. Зв'язок природного знака з позначуваним об'єктом - причинно-наслідковий. Наприклад, тривала відсутність відповіді на запит до базованої на локальній мережі автоматизованої системи свідчить про значний обсяг оброблюваної сервером баз даних інформації.

Ще один тип знака - це знак-образ. Образ уже не є частиною об'єкта, що позначається, але відбиває його найсуттєвіші властивості і, отже, повинен нагадувати цей об'єкт. Знак-образ віддаляється від позначуваного об'єкта більше, ніж знак-індекс, але все-таки ще пов'язаний з тим, що позначається, хоча б тим, що повинен бути на нього схожий. Образні знакові системи поширені у підсистемах інтерфейсу «користувач — автоматизована система», візуальних середовищах програмування.

Систематизація знакових систем, що використовуються у програмуванні, є окремою проблемою, тому наступним типом знакових систем у нашому розгляді будуть власне мови. Знаком у мовах є слово. Слово віддалене від того, що воно позначає: звуки при промовлянні слова «рі» чи відповідні літери у письмовому викладі не мають ніякого видимого зв'язку з відношенням довжини кола до його діаметра. Більше того, в одній МП воно позначає константу, а в іншій - функцію без параметрів. Таким чином, слово як знак є довільним. Якщо і є якийсь зв'язок між словами й об'єктами, що позначаються цими словами, то він визначається або правилами формалізованої мови, або користувачем автоматизованої системи.

Комплексне вивчення мов поряд із філософією, логікою та інформатикою здійснюється й семіотикою. Семіотика аналізує мову у кількох аспектах, з-поміж яких для МП найрозвиненішими є:

- синтаксичний;
- семантичний;
- прагматичний.

Синтактика - це розділ семіотики, що вивчає структуру мови (синтаксис) - способи утворення, перетворення й зв'язки між знаками. Абстрагуючись від усіх інших факторів, вона досліджує зв'язки між знаками деякої мови, установлює правила побудови складених знаків (наприклад, речень) із простіших знаків (наприклад, окремих слів чи простіших речень).

Семантика займається проблемою інтерпретації, тобто аналізом відношень між знаками та об'єктами, що позначаються ними, між словами та відповідними поняттями, а також вивчає відношення між значеннями простих знаків і значеннями складних знаків, складених із простих, наприклад відношення між значенням слів і зна-

ченням речень, побудованих із цих слів (для МП-програм).

У відомому семіотичному трикутнику [2] ім'я (номінація) позначає, іменує об'єкт (денотат) і виражає поняття про об'єкт (десигнат). Десигнат (сигніфікат, інтенсіонал) відповідає смислу чи значенню знака у нашій свідомості. Денотат (референт, екстенсіонал) позначає той об'єкт чи систему об'єктів, що відповідає даному імені (знаку, слову, словосполученню). Іншими словами, знак позначає денотат (знак називають також позначенням чи іменем, а денотат - тим, що позначається, чи значенням). Так, при передачі повідомлення саме повідомлення є знаком, а його зміст - денотатом.

Одним із прикладів знакової системи може бути система числових констант конкретної МП. Правила, що визначають перелік допустимих цифр та їх допустиме розташування, утворюють синтаксис, правила обчислення позначеного числа - семантику. При цьому запис числа є знаком, а саме число - денотатом.

Використовувані нами МП (у вузькому розумінні) є складнішими знаковими системами. У загальному випадку в МП знаки - це елементи програм (у тому числі й завершені програми), а денотати - елементи і ознаки поведінки виконавця (характеристики його поведінки), зокрема дані, операції управління, їх структура, їх зв'язки й атрибути. Наприклад, знаку «begin» в одному контексті (елементу програми мовою Паскаль) як денотат може відповідати такий елемент поведінки, як вхід у блок, а в іншому - змінна дійсного типу, у третьому - масив властивостей.

Далі розглянемо деякі способи структурування даних у програмуванні за допомогою формальних імен та методи оперування цими іменами та іменованими даними.

Процес розробки будь-якої складної програми супроводжується паралельним структуруванням даних та відповідних процедур і функцій для роботи з ними. Суттєвим є те, що сама структура даних є не менш важливою, ніж власне множина даних, бо також несе в собі семантичне значиму інформацію. Якщо врахувати той факт, що основні методи структурування даних пов'язані з іменуваннями об'єктів розгляду [3], то природною є теза про відповідність структур іменувань структуруванню оброблюваних даних.

У багатьох випадках спостерігається дисбаланс між використовуваними структурами даних і методами роботи з цими структурами. Наприклад, SQL явно представляє лише плоскі таблиці (досить бідні самі по собі структури даних) і дуже розвинену мову запитів до даних [4]. Є приклади інструментальних систем, які, навпаки, пропонують дуже розвинені засоби опису структур даних, але не включають взагалі ніяких методів роботи з цими структурами. Так, об'єктні СУБД у принципі пропонують потенційно багаті структури даних у сукупності з методами, але ці методи, як правило, призначені для роботи з наявними даними, а не для їх структурування.

Оброблювані будь-якою програмою дані утворюють деяку множину. Для того щоб вести їх обробку, розробнику програми потрібно якимось чином іменувати елементи цієї множини (чи її підмножин). У результаті таких дій утвориться вже дві множини: вихідна і множина імен, між якими існує деяке відображення. Основою спрощень, необхідних у процесі складної розробки, є абстрагування, бо дуже часто набагато простіше працювати не з усією вихідною множиною, а з деяким іншим образом її підмножини. Важливим є лише збереження деяких інваріантних сутностей. У математиці такий процес застосовується повсюдно. Як правило, замість вихідної множини береться її образ, з яким у деякому сенсі «зручніше» працювати в певній конкретній ситуації. Прикладом при обробці масивів числової інформації може слугувати перетворення матриці загального вигляду до деякого спеціального виду - діагонального, жорданової форми тощо. У випадку діагоналізації матриці замість n^2 параметрів (значень усіх елементів матриці) істотними для оперування залишаються тільки η власних чисел та власні вектори матриці.

Ранні мови програмування надавали розробнику практично ідентичні стандартні засоби структурування даних, що найчастіше зводились до утворення масивів однорідної інформації та записів, що містять різнорідну інформацію. Апарат динамічних структур із використанням вказівників та файлів різних типів (а в подальшому й таблиць баз даних) у більшості випадків не дає зручного та компактного способу іменування.

Так, розроблена в інструментальному середовищі Delphi [5] програма може містити послідовність операторів

```
Query1.Close;
Query1.SQL.Clear;
Query1.SQL.Add('Select Area from Country where
Name = ''Argentina''');
Query1.Open;
```

Результатом її виконання з точки зору проектувальника програми є єдине число із відомим значенням (за умови існування в таблиці відповідної інформації). Разом з тим існують приклади, що демонструють можливість адекватнішого іменування об'єктів розгляду. Так, Delphi за допомогою складеного імені *Table1.Fields[i]* дає доступ до значення у позначеному індексом *i* стовпчику поточного рядка таблиці, за допомогою *Table1.FieldByName('Area'). AsFloat* – до числового значення поля *Area,* а об'єктна модель HTML-сторінки дає змогу за допомогою конструкції *document.all["Table1"].rows* іменувати усі рядки таблиці.

Наведені у прикладах конструкції не є іменами в класичному трактуванні МП, хоча й повсюдно застосовуються в програмуванні. У перспективних МП (особливо орієнтованих на користувача-непро-

фесіонала) повинні бути розвинені засоби створення фактично довільних імен структур даних з деяких атомарних типів. Саме ім'я повинне бути «змінним» та структурованим, тобто містити у своєму складі як літерали, так і інші імена, у тому числі неявні. Як найпростіші приклади можна навести об'єктні посилання.

Як тільки допускається структурованість імені, неминуче повинен з'явитися набір операцій, що дає змогу працювати з окремими атомами структурованого імені. Якщо деяким іменам відповідають значення, то виникає відображення множини значень на множину імен, а оскільки це відображення може бути означене розробником програми для його ж (та читачів чи виконавців) зручності, то слід вкладати в структуру *імені структуру даних. Більш того,* частину даних варто розмістити в множині імен.

Наступним кроком є розгляд множини імен у якості множини значень, і у свою чергу поіменування і її. Така конструкція є частиною неявного іменування і застосовується у вигляді вказівників, посилань тощо. Саме це й дозволяє по-різному структурувати ті самі дані залежно від розв'язуваної задачі. Механізми ж, застосовувані в різних МП для реалізації цього «перейменування», різняться між собою і не є достатньо загальними.

Таким чином, з аналізу практики розробки програм та наведених прикладів стає зрозумілим, що на рівні абстракції, характерному для масового програмування, систематичне використання іменувань для структурування інформації забезпечується інтегрованим використанням засобів МП, а на вищих *рівнях абстракції проблема має вирішуватись* на єдиному концептуальному підґрунті.

1. *Редько В. Н.* Программология: прошлое, настоящее, будущее // Вестник Международного Соломонова университета,- 1999.- №1.-С. 23-59.
2. *Frege G.* Uber Sinn und Bedeutung // Zeitschrift ftlr Philosophic und philosophische Kritik- 1982-Bd. 100- S. 25-50.
3. *Редько В. Н.* Основания программологии // Кибернетика и системный анализ.-2000-№ 1.-С. 3-27.
4. *Буй Д. Б., Поляков С. А.* Композиційна семантика SQL-подібних мов: табличні структури даних, композиції, приклади // Вісник Київського університету. Серія фіз.-мат. наук.- 1999 .-№1.-С 130-140.
5. *МатчоДж., Фолкнер Р. Ф.* DELPHI.- М.: БИНОМ, 1995- 634 с.

B. L Boiko

THE NAMING STRUCTURE IN PROGRAMMING

The article is devoted to semantics of structural names in programming languages. Examples of the data structuring and comparison of the implementation peculiarity in different programming systems are presented.