DOI 10.31891/2307-5732-2021-297-3-216-221
УДК 004.43

Т. М. КОРОТУН
Заклад вищої освіти "Міжнародний науково-технічний університет імені академіка Юрія Бугая"

ПАРАДИГМИ МОВ ВИСОКОГО РІВНЯ

Програмування на мовах високого рівня пристосоване до представлення розширюваної ієрархії понять, що відбиває природу розуміння людиною вирішуваних завдань і організації процесів їх рішення. Програмування на мовах високого рівня пристосоване до завдання структур даних, що відбивають природу вирішуваних завдань. Використовується розширювана ієрархія зон видимості структур даних і процедур їх обробки, підпорядкована структурно-логічній моделі управління, що допускає збіжність процесу відлагодження програм. Підготовка програм на базі мов надвисокого рівня націлена на представлення регулярних, ефективно таких, що реалізовуються структур даних, при обробці яких можливі перетворення представлення даних і програм, використання подібностей і доказових побудов, що гарантують високу продуктивність обчислень і надійність процесу розробки програм, включаючи підготовку програм для багатопроцесорних конфігурацій.

Як правило, мови паралельного програмування включають засоби, характерні для різних парадигм. Це визначає можливість трансформаційного підходу до накопичення правильності програмних рішень при розробці і модернізації паралельних програм на різних мовах у рамках загальної системи програмування

Ключові слова: мова високого рівня, парадигма, підмова програмування.

TETYANA MYKHAYLIVNA KOROTUN
Academician Yuriy Bugay International Scientific and Technical University

PARADIGMS OF HIGH LEVEL LANGUAGES

High-level programming is adapted to the presentation of an extensible hierarchy of concepts, which reflects the nature of human understanding of the tasks to be solved and the organization of the processes of their solution. High-level programming is adapted to the task of data structures that reflect the nature of the tasks. An extensible hierarchy of visibility zones of data structures and procedures for their processing is used, subject to the structural-logical management model, which allows for convergence of the program debugging process. The preparation of programs based on ultra-high level languages is aimed at presenting regular, effectively implemented data structures, the processing of which can transform the representation of data and programs, the use of similarities and evidence, ensuring high computing performance and reliability of the program development process, including program preparation for multiprocessor configurations.

Typically, parallel programming languages include tools specific to different paradigms. This determines the possibility of a transformational approach to the accumulation of the correctness of software solutions in the development and modernization of parallel programs in different languages within the overall programming system

Keywords: high-level language, paradigm, programming premise.

Вступ

Завдяки мовам високого рівня (МВУ) програмування стало масовою професією. Програмування на МВУ пристосоване до представлення розширюваної ієрархії понять, що відбиває природу розуміння людиною вирішуваних завдань і організації процесів їх рішення. Перехід до МВУ дав можливість систематично укрупнювати конструкції при підготовці текстів програм. Для цього знадобилися складні структури даних, стереотипи техніки програмування, що локалізуються зоні видимості імен об'єктів і процедур їх обробки, підлеглі структурно-логічної моделі управління, що допускає збіжність покрокового процесу відлагодження програм.

Результативні графічні інтерфейси і компонентні технології, що підтримують перенесення відлагоджених результатів в різні системи. У центрі уваги - інтеграція з бібліотеками процедур, ефективна компіляція програм, контроль типів даних, відповідність стандартам сфери застосування програм і технологіям швидкої розробки зручно супроводжуваних програм. Ряд проблем ефективності вирішується включенням в МВУ низькорівневих засобів.

Практично зникає необхідність у блоксхемах, а методика самодокументування і реалізації довідкових підсистем пом'якшує роль документування. Програмі на МВУ зазвичай відповідає сімейство допустимих процесів, визначення якого представлене формальною семантикою мови. Система програмування (СП), що підтримує МВУ, як правило, породжує один з процесів цього сімейства. Таке звуження диктується не лише реалізаційною прагматикою МВУ, але і необхідністю відтворення процесів при відлагодженні програм.

Альтернативні підходи до обробки інформації, що склалися при створенні і застосуванні мов і систем програмування, прийнято називати парадигмами програмування (ПП). Незалежно від області додатка, ПП можуть відрізнятися рівнем абстрагування від можливостей апаратури, мірою вивченості постановок завдань, мірою організованості і рангом працездатності програм рішення завдань. При визначенні ПП зазвичай поляризуються наступні характеристики МП :

- програмовані рішення представляються в імперативно-процедурній або в декларативній формі;
- оброблювані елементи даних позиціонуються як блоки пам'яті, що адресуються, або незалежно розміщувані значення;
- програма може бути захищена від змін в процесі її виконання або допускати програмовані модифікації по ходу отримання результатів обчислення;

- програма може бути цілісною або збиратися з типових компонент і шаблонів;
- представлені в програмі функції можуть бути частковими, такими, що типізуються, оброблювальними значення заданого домену або універсальними, такими, що дають розумну реакцію на будь-який елемент даних;
- управління обчисленнями виконується послідовно або паралельно;
- порядок дій може бути визначеним або недетермінованим;
- обчислення можуть бути енергійними або ледачими;
- зони видимості імен можуть бути глобальними або локалізованими за ієрархією конструкцій з можливістю відновлення контексту;
- розподіл і повторне використання пам'яті може бути дією в програмі або виконуватися автоматично системою програмування (СП);
- ініціація пам'яті спочатку розміщуваними значеннями може вимагати програмованих дій або виконуватися в СП за умовчанням;
- домени значень можуть бути незалежними або допускати перетину;
- результат виконання програми може бути розосереджений по ряду змінних або сконцентрований в спеціальному регістрі;
- контроль правильності може виконуватися статично – при аналізі тексту програми або динамічно - при виконанні коду програми.

Вибір між так протиставленими характеристиками в тексті програми виражається синтаксично, нерідко за допомогою специфікаторів мови програмування (МП). Визначення операційної семантики МП зазвичай використовує поняття абстрактної машини, яка у разі багатопроцесорних конфігурацій природно стає конструкцією з абстрактних процесорів, - абстрактним комплексом, що дозволяє виділяти схемний рівень і спрощувати включення в схему розробки програм механізмів верифікації (подібність моделі або відповідність аксіомам), а на їх основі підключати перевірку програм на правдоподібність, логічний висновок властивостей, виконання індуктивних і дедуктивних побудов [4].

Крім того, межі між областями практичного прояву різних парадигм програмування можна виразити типовими схемами постановок завдань на програмування.

Стандартне імперативно-процедурне програмування: "Визначений алгоритм рішення актуальної задачі. Необхідно отримати програму реалізації алгоритму з практичними просторово-часові характеристики на доступному устаткуванні".

Функціональне програмування: "Відома предметна область. Слід вибрати символьне представлення даних для цієї області і відлагодити систему універсальних функцій, придатних для створення прототипів рішення актуальних завдань з цієї області".

Логічне програмування: "Дана колекція фактів і стосунків, що характеризує актуальне завдання. Потрібно привести цю колекцію до форми, що демонструє відповіді на практичні запити відносно цього завдання".

Об'єктно-орієнтоване програмування: "Доступна ієрархія класів об'єктів з працездатними методами рішення ряду завдань деякої сфери додатка ІТ. Вимагається без зайвих трудовитрат уточнити цю ієрархію, щоб пристосувати її до рішення нових затребуваних завдань цієї області, її розширення або їй подібною".

Багато завдань включають такі формулювання в якості підзадач, що приводить при створенні МП і розробці СП до підтримки різних парадигм одночасно. Так, наприклад, при цілеспрямованій розробці монопарадигматичної мови Haskell, що позиціонується як чисто функціональна мова, автори прийшли до концепції монад, що дозволяє притягати механізми інших парадигм. Потреба в підтримці парадигм, відсутніх в МП, що реалізовується, може бути забезпечена в СП за допомогою бібліотечних процедур, асемблерних вставок, макрогенераторів або організації виходу на рівень операційної системи.

Діалекти МП часто бувають реалізаційний рівнозначний, можливо семантично еквіваленти і рівнопотужні, але помітні по експлуатаційній прагматиці, лексиці і синтаксису :

– учбові концентри для ознайомлення з основними ідеями;

– практичні подмови для програмування рішень типових завдань;

– проблемно-орієнтовані варіації для розширення сфери застосування;

– повні мови для дослідження і вибору поліпшень.

При неформальній характеристиці стилю програмування відзначаються відмінності в акцентах при відповіді на наступні питання:

1. У чому полягає основний метод обробки програми при відлагодженні?
2. Що показує результативну активність програми?
3. Коли приймається рішення про продовження незавершѐнных обчислень?

У практиці визнають основними парадигми імперативно-процедурного, функціонального, логічного і об'єктно-орієнтованого програмування, підтримувальні механізми зниження трудомісткості повного життєвого циклу програм, з тенденцією просування до паралельного програмування. Нині по суті розрізняють більше двох десятків парадигм. Багато МП відносять до п'яти-восьми парадигм. Частину згадуваних в різних джерелах парадигм можна характеризувати як стилі або методики, що відбивають пошук шляхів зниження трудомісткості програмування і підвищення надійності програм на базі доступних

СП. Наприклад, аспектно-орієнтоване програмування підтримується як макророзширення ТОП. Структурне програмування фактично зводиться до ряду рекомендацій по стилю представлення імперативно-процедурних програм. Метапрограммирование є технікові компіляції програм в комплекті з типовими елементами даних. Зустрічаються прецеденти розгляду недетермінізму як основній реалізації паралелізму.

Таблиця 1

Список понятий МП, различаемых операционной семантикой, определяемыми для сравнения

Поняття	***Пояснення***
Атом / Скаляр / Літерал	Атоми і скаляри можуть бути різних категорій або типів, що розрізняються динамічно або декларативно
Структура даних	Можливі обмеження на характер елементів структури, їх число і динаміку їх змін.
Змінна	Може бути ініційована до обчислень, обмежена наказаним типом даних, заданим статично або що виводиться за програмою
Значення	Різні типи значень пов'язані з різними операціями їх обробки і синтаксичними позиціями в тексті програми, що допускають або вимагають їх входження
Вираження	Форма, результат якої може бути вичислений і використаний як параметр в інших формах
Дія / Операція	Вбудована команда або підпрограма, що розглядається як елементарна база при організації обчислення
Умова / Логіка	Концепція істинних значень може вимагати як спеціального типу даних, так і розглядатися як навантаження звичайних значень (0, NIL).
Функція	Фрагмент програми, що можливо параметризується, представляє укрупнену одиницю, використовувану нарівні з операціями при організації обчислень.
Аргумент	Фактичний параметр використовуваної функції.
Виклик функції	Форма, використовувана для виконання функції при заданих параметрах.
Визначення функції	Форма, що представляє фрагмент програми, призначений для використання як функції.
Ідентифікатор/Ім'я	Унікальна форма, що створюється як синонім багаторазово використовуваного елементу даних.

Слід особливо відмітити трудомісткість паралельного програмування, що призводить до тенденції включати його у будь-яку парадигму на правах узагальнення окремих конструкцій і тим самим відкладати на майбутнє необхідність розробки методів верифицирующей компіляції і оптимізації програмних компонентів, засобів масштабованої генерації коду і трансформацій програм, що автоматизуються, з посвідченням збереження властивостей при їх комплексації з раніше відлагоджених компонентів [8]. Парадигма паралельного програмування займає нішу, пов'язану з реалізацією програм виконання обчислень на багатопроцесорних системах для організації високопродуктивних обчислень [3]. Ця ніша обтяжена різким підвищенням складності відлагодження програм, викликаною комбінаторикою виконання фрагментів асинхронних процесів. Якщо перший бар'єр до успіху в паралельному програмуванні обумовлений послідовним стилем мислення, то другий бар'єр залежить від концептуально не здоланої трудомісткості відлагодження програм. Хоча основна трудомісткість життєвого циклу програм пов'язана саме з тестуванням і відлагодженням програм, цей напрям за півстоліттям професійного програмування прогресує переважно за рахунок «силових« характеристик устаткування, концептуально не виходячи за межі можливостей екстракодів відлагодження програм в домовну епоху. Доцільність методів верифікації програм, що вже досягли практичних характеристик, ще не отримала довіри наших практиків, можливо із-за кваліфікаційного цензу на створення формальних специфікацій [2].

Перехід до паралельних алгоритмів тягне перегляд змісту багатьох понять і введення нових термінів, що відбивають різного роду явища і ефекти, що не мали особливого значення для звичайних послідовних алгоритмів. Алгоритми стають паралельними, програми - багатопотоковими, виконання коду - багатопроцесорним, дія може почати виконуватися до завершення попереднього, засоби управління окрім логічних значень використовують сигнали і повідомлення, обробка структур даних може вимагати зміни порядку їх обробки і багато що інше. Розширення простору понять, що виникає в результаті, міняє підходи до реалізації рішень, що використовують паралелізм, і в деякій мірі позначається власне на постановці завдань і плануванні життєвого циклу програм рішення завдань, орієнтованих на використання паралелізму.

Різноманітність моделей паралельних обчислень і розширення спектру доступної архітектури слід розглядати як виклик розробникам мов і систем програмування (МСП), спрямованих на рішення проблем створення методів компіляції багатопотокових програм для багатопроцесорних конфігурацій. Мова повинна допускати представлення будь-яких моделей паралелізму, що проявляється на рівні вирішуваної задачі або реалізовується за допомогою реальної архітектури, причому таке представлення вимагає різнорівневих форм і конструктивних побудов, що гарантують збереження властивостей програм при їх реорганізації. Ефективна реалізація паралельних алгоритмів використовує дуже широкий спектр складно сумісних засобів від дій нижчого рівня, ніж в звичних МП, що управляють, до маніпулювання просторами рішень по обробці даних в пам'яті, типових для мов понад високий рівень. Не менш важлива розширюваність мови у міру розвитку засобів і методів паралельних обчислень, темп якого перевищує швидкість усвідомлення фахівцями їх можливостей.

Як правило, мови паралельного програмування включають засоби, характерні для різних парадигм. Це визначає можливість трансформаційного підходу до накопичення правильності програмних рішень при розробці і модернізації паралельних програм на різних мовах у рамках загальної системи програмування. Розвиток МСП нині орієнтований на рішення завдань на основі загальних бібліотечних модулів, що забезпечують ефективну організацію процесів, або підмов, що допускають багатопотокове програмування.

Це не виключає реальну практику ручного розпаралелювання раніше відлагоджених звичайних програм, приведення їх до виду, зручного для застосування виробничих систем підтримки паралельних обчислень. Значна частина таких робіт носить технічний характер і полягає в систематичній реорганізації структур даних, зміні статусу змінних і включенні в програму анотацій, що повідомляють компілятор про інформаційно-логічні взаємозв'язки. Істотним обмеженням результату ручного розпаралелювання є не лише небезпека повторної відлагодження алгоритму, але і його надмірна залежність від характеристик цільової архітектури.

Розгляд технічних особливостей МСП через парадигмальную характеристику може дати зручні форми їх визначення відносно ряду простих концептуальних мов, включаючи мови низького рівня, обробки даних, що накопили механізми, успадковані методами реалізації мов високого рівня. Складність і суперечність класифікації безлічі МСП, що нестримно розширюється, при порівняно невеликому наборі помітних ПП приводить до поняття «визначника» підтриманих в МСП парадигм [6], що містить наступні процедури:

- розкладання мови на фрагменти по рівнях з метою виділення базових засобів мови і його реалізаційного ядра - семантичний базис;
- декомпозиція семантичного базису мови на основні семантичні системи з мінімізацією їх складності і, можливо, їх опис відносно концептуальних мов, що підтримують відомі парадигми;
- визначення абстрактної машини мови і інтерпретатора, формально достатнього для побудови розширень, еквівалентних початковій мові – нормалізоване визначення;
- порівняння отриманого визначення з описами відомих парадигм і концептуальних мов;
- обгрунтування виводів відносно парадигм досліджуваної мови;
- визначення рівня мови і його ніші в життєвому циклі програм і діяльності програмістів (цілі і завдання), базових мов, використаних при його створенні і реалізації, як основи для рекомендацій по вибору і застосуванню МП і його СП.

Розглядаючи завдання формалізації мов паралельного програмування як шлях до вирішення проблеми адаптації програм до різних особливостей використовуваних багатопроцесорних комплексів і багатоядерних процесорів, ми бачимо, що вирішення цієї проблеми вимагає розробки нових методів компіляції програм з акцентом на тестування, верифікацію і відладку, а також, розвитку засобів і методів ясного опису семантики мов паралельного програмування, включаючи представлення програмованих перетворень тексту і коду програми з посвідченням їх коректність.

Прийшов час вивчати інформатику і програмування, починаючи зі світу паралелізму. Спостережливі люди помічали, що виконання «однакових« дій має різну тривалість, що тривалість реагування інформаційної системи може залежати від поточної ситуації, що власне здійснимість дій залежить від не завжди заздалегідь відомих умов, що системи можна і треба настроювати, що ряд систем може працювати одночасно і впливати на роботу один одного. Їм відомо, що існують кеші, протоколи, веріфікатори, резервне копіювання, транзакції, «перегони» даних, «смертельні обійми» і багато що інше. Вони начулися про успіхи любительської астрономії, перспективи GRID - технологій і «хмарних» обчислень. Активізація таких знань утворює основу для швидкого вивчення засобів і методів паралельного програмування, початок якому дає пропонований експериментальний курс [5].

Програмування на мовах високого рівня пристосоване до завдання структур даних, що відбивають природу вирішуваних завдань. Використовується розширювана ієрархія зон видимості структур даних і процедур їх обробки, підпорядкована структурно-логічній моделі управління, що допускає збіжність процесу відлагодження програм. Підготовка програм на базі мов надвисокого рівня націлена на представлення регулярних, ефективно таких, що реалізовуються структур даних, при обробці яких можливі перетворення представлення даних і програм, використання подібностей і доказових побудов, що гарантують високу продуктивність обчислень і надійність процесу розробки програм, включаючи підготовку програм для багатопроцесорних конфігурацій.

Особливий круг освітніх проблем пов'язаний з навичками обліку особливостей багаторівневої пам'яті у багатопроцесорних системах. Звичайне послідовне програмування такі проблеми може не помічати, покладаючись на рішення компілятора, що має в розпорядженні статичну інформацію про типи використовуваних даних і здатного при необхідності виконати програми, що оптимізують перетворення. На жаль, використання мов паралельного програмування як мови представлення програми не гарантує її пристосованість до вдалого розпаралелювання.

Висновки

При оцінці освітнього значення парадигм виділяють в якості базових функціональне, паралельне і імперативно-процедурне програмування, а логічне і об'єктно-орієнтоване розглядає як доповнення, що вивчається у вигляді розширення базових парадигм.

Помітимо, що найбільш успішні, довго такі, що живуть мови і системи програмування являються мультипарадигмальными. Об'єднання різних парадигм у рамках одного МП або їх підтримка при реалізації СП підвищує їх сферу застосування і сприяє продуктивності праці програмістів, пом'якшуючи надмірну різноманітність теорій і моделей, що склалися на різних фазах і етапах життєвого циклу програм. Багато авторів нових технологій програмування, таких як екстремальне програмування і функціонально орієнтоване проектування, виражають претензії до рекомендацій, що історично склалися, по стилю і техніці

програмування, що нібито зробило практику програмування безпорадної [1,7]. Тут бажано пам'ятати про високий темп розвитку ИТ, усвідомлення можливостей яких не устигає дозрівати. Мультипарадигмалльность довго МП, що живуть, і тенденція створення нових мультипарадигмальных МП говорить про дозрівання єдиної парадигми, що об'єднує вивірені в практиці механізми програмування [9]. Не виключено, що розвиток методів компіляції таких мов приведе до добротних засобів і методів підготовки паралельних програм [10].

Література

1. Бек, К. Экстремальное программирование – Питер, 2002

2. Бурдонов И.Б., Косачев А.С., Кулямин В.В. Теория соответствия для систем с блокировками и разрушениями − М.Ж Физматлит, 2008. − 412 с.

3. Воеводин В. В. Параллельные вычисления / В. В. Воеводин, Вл. В. Воеводин. – СПб.: БХВ-Петербург, 2002. – 608 с.

4. Городняя Л.В. О проблеме автоматизации параллельного программирования // В сборнике Международной суперкомпьютерной конференции «Научный сервис в сети Интернет: многообразие суперкомпьютерных миров » – URL: http://agora.guru.ru/abrau2014

5. Городняя Л.В. Парадигмы параллельного программирования в университетских образовательных программах и специализации // Всероссийская научная конференция "Научный сервис в сети Интернет: решение больших задач − Новороссийск-Москва, 2008. − с. 180-184. Л.В.

6. Городняя Л.В. Парадигмы программирования Часть 1 Сравнение парадигм программирования Препринт 172 URL: - http://www.iis.nsk.su/files/preprints/172.pdf

7. Палмер С.Р., Фелсинг. Дж.М. Практическое руководство по функционально ориентированной разработке ПО. − М.: Вильямс, 2002. − 299 с.

8. Степанов Г.Г. Пути обеспечения переносимости программ и опыт использования системы СИГМА // Трансляция и преобразование программ. – Новосибирск: ВЦ СО АН СССР, 1984. – 9 с.

9. Хорстман К. Scala для нетерпеливых. – ДМК пресс, 2013. – 408 с. – 300 экз. – ISBN 978-5-94074-920-2, 978-0-321-77409-5.

10. Knoop J. Compiler Construction / 20th International Conference, CC 2011, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2011 Saarbrücken, Germany, March 26 —April 3, 2011 // Lecture Notes in Computer Science. Springer V2011. V. 6601. 330 p.

11. http://sourceforge.net/projects/mozart-oz/ – Сайт с материалами по системе Mozart, поддерживающей учебный мультипарадигмальный язык программирования Oz.

12. Парадигмы и языки в обучении информатике и программированию / Е. И. Большакова, Н. В. Баева, Н. В. Груздева, И. В. Горячая // Материалы Междунар. науч.-практ. конф. «Перспективные инновации в науке, образовании, производстве и транспорте», Одесса, 2012. Т. 4, вып. 2. С. 77–82.

13. Watt D., Findlay W. Programming language design concepts. Great Britain: John Wiley & Sons Ltd, 2004. 473 p.

14. Van Roy P. Programming Paradigms for Dummies: What Every Programmer Should Know. 2009. URL: http://www.info.ucl.ac.be/~pvr/paradigms.html.

15. Себеста Р. У. Основные концепции языков программирования. М.: Вильямс, 2001. 672 с.

16. Normark K. Overview of the four main programming paradigms / Aalborg University. 2013. URL: http://people.cs.aau.dk/~normark/prog3-03/html/notes/theme-index.html.

17. Пантелеев М. Г., Родионов С. В. Модели и средства построения экспертных систем: учеб. пособие. СПб.: Изд-во СПбГЭТУ«ЛЭТИ», 2003. 68 с.

References

1. Bek, K. Jekstremal'noe programmirovanie – Piter, 2002

2. Burdonov I.B., Kosachev A.S., Kuljamin V.V. Teorija sootvetstvija dlja sistem s blokirovkami i razrushenijami − M.Zh Fizmatlit, 2008. − 412 s.

3. Voevodin V. V. Parallel'nye vychislenija / V. V. Voevodin, Vl. V. Voevodin. – SPb.: BHV-Peterburg, 2002. – 608 s.

4. Gorodnjaja L.V. O probleme avtomatizacii parallel'nogo programmirovanija // V sbornike Mezhdunarodnoj superkomp'juternoj konferencii «Nauchnyj servis v seti Internet: mnogoobrazie superkomp'juternyh mirov » – URL: http://agora.guru.ru/abrau2014

5. Gorodnjaja L.V. Paradigmy parallel'nogo programmirovanija v universitetskih obrazovatel'nyh programmah i specializacii // Vserossijskaja nauchnaja konferencija "Nauchnyj servis v seti Internet: reshenie bol'shih zadach − Novorossijsk-Moskva, 2008. − s. 180-184.

6. Gorodnjaja L.V. Paradigmy programmirovanija Chast' 1 Sravnenie paradigm programmirovanija Preprint 172 URL: - http://www.iis.nsk.su/files/preprints/172.pdf

7. Palmer S.R., Felsing. Dzh.M. Prakticheskoe rukovodstvo po funkcional'no orientirovannoj razrabotke PO. − M.: Vil'jams, 2002. − 299 s.

8. Stepanov G.G. Puti obespechenija perenosimosti programm i opyt ispol'zovanija sistemy SIGMA // Transljacija i preobrazovanie programm. – Novosibirsk: VC SO AN SSSR, 1984. – 9 s.

9. Horstman K. Scala dlja neterpelivyh. – DMK press, 2013. – 408 s. – 300 jekz. – ISBN 978-5-94074-920-2, 978-0-321-77409-5.

10. Knoop J. Compiler Construction / 20th International Conference, CC 2011, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2011 Saarbrücken, Germany, March 26 —April 3, 2011 // Lecture Notes in Computer Science. Springer V2011. V. 6601. 330 p.

11. http://sourceforge.net/projects/mozart-oz/ – Sajt s materialami po sisteme Mozart, podderzhivajushhej uchebnyj mul'tiparadigmal'nyj jazyk programmirovanija Oz.

12. Paradigmy i jazyki v obuchenii informatike i programmirovaniju / E. I. Bol'shakova, N. V. Baeva, N. V. Gruzdeva, I. V. Gorjachaja // Materialy Mezhdunar. nauch.-prakt. konf. «Perspektivnye innovacii v nauke, obrazovanii, proizvodstve i transporte», Odessa, 2012. T. 4, vyp. 2. S. 77–82.

13. Watt D., Findlay W. Programming language design concepts. Great Britain: John Wiley & Sons Ltd, 2004. 473 p.

14. Van Roy P. Programming Paradigms for Dummies: What Every Programmer Should Know. 2009. URL: http://www.info.ucl.ac.be/~pvr/paradigms.html.

15. Sebesta R. U. Osnovnye koncepcii jazykov programmirovanija. M.: Vil'jams, 2001. 672 s.

16. Normark K. Overview of the four main programming paradigms / Aalborg University. 2013. URL: http://people.cs.aau.dk/~normark/prog3-03/html/notes/theme-index.html.

17. Panteleev M. G., Rodionov S. V. Modeli i sredstva postroenija jekspertnyh sistem: ucheb. posobie. SPb.: Izd-vo SPbGJeTU«LJeTI», 2003. 68 s.

Рецензія/Peer review : 23.04.2021 р. Надрукована/Printed :30.06.2021 р.