УДК 004.43: 042.4

О ПАРАДИГМЕ ПАРАЛЛЕЛЬНОГО ПРОГРАММИРОВАНИЯ

Городняя Лидия Васильевна
К.ф.-м.н., доцент, старший научный сотрудник, e-mail: lidvas@gmail.com
Новосибирский национальный исследовательский
государственный университет,
630090, г. Новосибирск, ул. Пирогова, д. 2.
Институт систем информатики им. А.П. Ершова
Сибирского отделения Российской академии наук,
630090, г. Новосибирск, проспект Академика Лаврентьева, 6.

Аннотация. Доклад посвящен проблеме формирования самостоятельной парадигмы параллельного программирования, вызванной расширением и развитием системы базовых понятий, необходимых для рациональной разработки информационных систем управления процессами на современной аппаратуре.
Ключевые слова: парадигма параллельного программирования, языки и системы программирования, методы определения компьютерных языков, параллельные алгоритмы, учебное программирование

Введение. Парадигмы в программировании характеризуются стилем мышления при решении программистских задач, системой используемых и вводимых понятий и особенностями их практичной реализации. Стиль мышления и система понятий для параллельного программирования (ПП) уже сложились в процессе эволюции языков программирования (ЯП), но типовая их поддержка в системах программирования (СП) ещё не сформировалась [5]. Переход к параллельным алгоритмам (ПА) связан с пересмотром содержания многих понятий и введением новых терминов, отражающих разного рода явления и эффекты, не имевшие особого значения для обычных последовательных алгоритмов. Ведущую роль в результативности такого перехода, бесспорно, играет содержание образовательных программ подготовки специалистов в области информационных технологий (ИТ) [1, 6, 9].

1. Интеллектуальный вызов. Разнообразие моделей параллельных вычислений (ПВ) и расширение спектра доступной архитектуры следует рассматривать как вызов разработчикам языков и систем программирования (ЯСП), способным решать проблему создания методов компиляции многопоточных программ для многопроцессорных конфигураций [15]. ЯП должен допускать представление любых моделей параллелизма, проявляемого на уровне решаемой задачи или реализуемого с помощью реальной архитектуры, причем такое представление требует лаконичных форм и конструктивных построений, гарантирующих сохранение свойств программ при их реорганизации. Не менее важна расширяемость СП по мере развития средств и методов параллельных вычислений, темп которого превышает скорость осознания специалистами их возможностей [8].

Парадигма параллельного программирования (ППП) занимает нишу, связанную с реализацией программ выполнения вычислений на многопроцессорных системах для организации высокопроизводительных вычислений. Эта ниша обременена резким

повышением трудоемкости отладки программ, вызванной комбинаторикой выполнения фрагментов асинхронных процессов. Полноценное решение проблем ПП требует создания более специализированного инструментария, некоторые механизмы реализации которого могут быть изучены в форме экспериментальной разработки учебного языка ПП [6, 9].

Переход к параллельным алгоритмам (ПА) влечёт пересмотр содержания многих понятий и введение новых терминов, отражающих разного рода явления и эффекты, не имевшие особого значения для обычных ПА. Резкое расширение пространства решений задач меняет подходы к реализации решений, использующих параллелизм, и в некоторой мере сказывается собственно на постановке задач и планировании жизненного цикла программ решения задач, ориентированных на использование параллелизма.

Рассматривая задачу формализации языков параллельного программирования (ЯПП) как путь к решению проблемы адаптации программ к различным особенностям используемых многопроцессорных комплексов и многоядерных процессоров, приходится признать, что решение этой проблемы требует разработки новых методов компиляции программ и развития средств и методов ясного описания операционной семантики ЯПП, возможно дополненной описанием денотационной и аксиоматической семантики [7].

Для практичного решения проблемы разработки ЯПП и подходов к их реализации необходимо рассматривать вопросы системной поддержки ППП как базовой, отражающей прагматические различия в условиях реализации и применения изобразительных средств, используемых в жизненном цикле программ. В этом плане средства языков сверхвысокого уровня позволяют представлять регулярные, эффективно реализуемые структуры данных, гарантирующие высокую производительность вычислений и надежность процесса разработки программ, включая подготовку программ для многопроцессорных конфигураций.

2. Пространство парадигм программирования. В настоящее время по существу различимы более двух десятков парадигм программирования. Многие ЯП относят к пяти-восьми парадигмам. Часть упоминаемых в разных источниках парадигм можно характеризовать как технологии, стили или методики, отражающие поиск путей снижения трудоёмкости программирования и повышения надёжности программ на базе доступных СП [3]. Аспектно-ориентированное программирование поддерживается как макрорасширение ООП. Структурное программирование фактически сводится к ряду рекомендаций по стилю представления императивно-процедурных программ. Мета-программирование представляет собой технику компиляции программ в комплекте с типовыми элементами данных. Недетерминированное программирование иногда рассматривают как частный случай параллелизма [8]. При определении парадигм обычно выделяются следующие характеристики ЯП:

1) программируемые решения представляются в императивно-процедурной или в декларативной форме;
2) обрабатываемые элементы данных позиционируются как адресуемые блоки памяти или независимо размещаемые значения;
3) программа может быть защищена от изменений в процессе её выполнения или допускать программируемые модификации по ходу получения результатов
4) вычисления;
5) программа может быть целостной или собираться из типовых компонентов и шаблонов;

6) представленные в программе функции могут быть частичными, типизированными, обрабатывающими значения заданного домена или универсальными, дающими разумную реакцию на любой элемент данных;
7) управление вычислениями выполняется последовательно или параллельно;
8) порядок действий может быть определённым или недетерминированным;
9) вычисления могут быть «энергичными» или «ленивыми»;
10) области видимости имён могут быть глобальными или локализованными по иерархии конструкций с возможностью восстановления контекста;
11) распределение и повторное использование памяти может быть действием в программе или выполняться автоматически СП;
12) инициирование памяти первоначально размещаемыми значениями может требовать программируемых действий или выполняться в СП по умолчанию;
13) домены значений могут быть независимыми или допускать пересечения;
14) результат выполнения программы может быть рассредоточен по ряду переменных или сконцентрирован в специальном регистре;
15) контроль правильности может выполняться статически – при анализе текста программы или динамически – при выполнении кода программы.

В практике признают основными парадигмы императивно-процедурного, функционального, логического и объектно-ориентированного программирования, суммарно покрывающие это поляризованное пространство, поддерживающее практичные механизмы снижения трудоёмкости полного жизненного цикла программ, с тенденцией продвижения к ПП, встраиваемому в контекст привычных парадигм. Си и Фортран предлагают следующий выбор предпочтений:

1) программируемые решения представляются в императивно-процедурной, дополненной описаниями типов данных, включая функции/процедуры;
2) обрабатываемые элементы данных позиционируются как непрерывно адресуемые блоки памяти, а независимо размещаемые значения допускаются в основном при передаче параметров;
3) программа обычно защищена от изменений в процессе её выполнения, но полезность программируемых модификаций получила право на жизнь в языке C#;
4) программа целостна, хотя на уровне компиляции собирается из типовых шаблонов;
5) представленные в программе функции обычно являются частичными, типизированными, поскольку универсальность часто приводит к противоречиям с системой типового контроля, упрощающей работу компилятора;
6) управление вычислениями выполняется последовательно, а параллельное управление проникает в программу через специальные библиотечные модули и доработку программы «вручную»;
7) порядок действий заранее определён, если нужна недетерминированность, то она моделируется;
8) «энергичные» вычисления понятнее, хотя «ленивые» признаются более результативными;
9) области видимости имён преимущественно являются глобальными или отчасти локализованными по иерархии модулей, вызовов процедур или классов объектов с возможностью восстановления контекста;

10) распределение и повторное использование памяти выполняется действием в программе, но в языках Java и C# появилась возможность автоматической «сборки мусора»;
11) инициирование памяти первоначально размещаемыми значениями может требовать программируемых действий, хотя инструментальные оболочки делают это по умолчанию;
12) домены значений обычно независимы, роль пересечений выполняют преобразования типов данных;
13) результат выполнения программы рассредоточен по ряду переменных;
14) контроль правильности может выполняться статически при анализе компилируемой программы, что обосновывается экономией памяти и времени исполнения для динамического контроля.

3. Системы программирования. При измерении производительности суперкомпьютеров и в экспериментах с распараллеливанием программ активно используются задачи научных расчётов, преимущественно реализующих алгоритмы векторной обработки данных. Практические задачи современного ПП обычно выглядят как приведение больших ранее отлаженных программ на самых живучих СП для языков C или Fortran к форме, дающей выигрыш от распараллеливания с помощью штатных средств, включаемых в доступные СП.

В целом такая работа сводится к следующим видам работы:

- Разметка участков программы, допускающих автоматическое распараллеливание.
- Анализ участков и причин, препятствующих распараллеливанию программы.
- Выбор участков программы, допускающих их техническое приведение к форме, пригодной для распараллеливания.
- Изобретение рецептов полуручного преобразования текста программы с целью расширения возможностей распараллеливания.
- Приведение текста программы к предельно распараллеливаемой форме.
- Прогон распараллеленной версии программы для оценки выигрыша от параллелизма.
- Частичное перепрограммирование и отладка фрагментов программы для исключения или смягчения эффектов, препятствующих достижению нужных характеристик производительности.
- Установление частичной функциональной эквивалентности исходной программы и её результирующей версии.

Обычно компилятор поддерживает оптимизацию, обеспечивающую устранение неиспользуемого кода, чистку циклов, слияние общих подвыражений, перенос участков повторяемости для обеспечения однородности распараллеливаемых ветвей, раскрутку или разбиение цикла, втягивание константных вычислений, уменьшение силы операций, удаление копий агрегатных конструкций и др. Рассматривается зависимость ускорения вычислений от числа процессоров и объема общей и распределенной памяти. Выполняется систематическая замена рекурсии на циклы. Предпочитаются однородное пространство процессоров, общая память, быстрые обмены, соседство, гарантирующие улучшение производительности систем для высокопроизводительных вычислений. Заметно влияние дисциплины работы с памятью на характеристики параллельных процессов. Используется защищенная и размазанная память. Различны решения, принятые в разных языках

программирования, по работе с многоуровневой и разнородной памятью (доступ, побочный эффект, реплики, дубли и копии). Обработка транзакций становится одной из типовых семантик работы с памятью в ЯП.

Особый круг проблем связан с навыками учёта особенностей многоуровневой памяти в многопроцессорных системах. Обычное последовательное программирование такие проблемы может не замечать, полагаясь на решения компилятора, располагающего статической информацией о типах используемых данных и способного при необходимости выполнить оптимизирующие преобразования программы.

Следует отметить, что использование ЯПП в качестве языка представления исходной программы не гарантирует её приспособленность к удачному распараллеливанию. При анализе пригодности программы к распараллеливанию анализируются потенциальные зоны риска, требующие дополнительных испытаний и отладки [4].

4. Параллельные алгоритмы. Прежде всего, следует прояснить следующие вопросы, связанные с постановками практических задач:

- Насколько изменится трудоёмкость жизненного цикла программы решения задачи с помощью параллельного алгоритма?
- В какой мере при постановке задачи следует учитывать модель параллелизма?
- Как обосновать и измерить выигрыш от разработки параллельного алгоритма?
- Насколько изменяется постановка задачи при переходе к параллельным алгоритмам?
- Что даёт парадигма параллельного программирования на уровне разработки параллельного алгоритма?
- Какими средствами представляются разрабатываемые параллельные алгоритмы решения задачи на этапе, предшествующем разработке программы?

За полвека традиционного последовательного программирования отлажено колоссальное количество программ, аккумулированных в СП и стандартные библиотеки. Изменение постановок задач, уже имеющих готовые отлаженные программные решения, ради учёта допустимого параллелизма чревато повторным программированием и, что гораздо более трудоёмко, повторной отладкой. Основной аргумент за разработку ПА – целесообразность учета естественного параллелизма на уровне постановки задачи, утрачиваемого при решении задачи посредством обычных алгоритмов. Число ЯПП, удобных для реализации ПА, год от года растёт, хотя и их применение решает не все проблемы организации ПВ [14]. Так, отмечая простоту записи параллельной композиции, можно сложность её отладки даже для несложных программ оставить в тени [13].

Нередко ПА может быть реализован по частям на множестве различных устройств с последующим объединением полученных результатов и получением целевого результата. Возникают чисто практические вопросы:

- Каким образом в определении алгоритма выделены части, выполняемые отдельными устройствами?
- Обязана ли реализация алгоритма использовать в точности представленный в его определении набор устройств?
- Можно ли последовательный алгоритм рассматривать как параллельный, исполняемый на одном устройстве?

Следующая обойма вопросов касается категории «время» и связана с проблемами

синхронизации:

- Могут ли части параллельного алгоритма обладать своим независимым или централизованным отсчетом времени?
- Может ли синхронизация частей алгоритма противоречить его информационным связям и логике управления?
- Можно ли синхронизацию частей алгоритма рассматривать как частный случай асинхронности?
- Особые сложности параллелизма вызывают вопросы доступа к памяти:
- Каким образом взаимодействующие части параллельного алгоритма обмениваются данными?
- Могут ли части параллельного алгоритма изменять состояние общей памяти и памяти других частей?
- Может ли часть параллельного алгоритма воспрепятствовать использованию своей памяти другими частями?

Кроме того, части алгоритма могут быть определены в разных моделях вычислений и над разными структурами данных. Оценка результата разработки параллельного алгоритма, кроме оценки сложности вычислений и объёма данных, осложнена целесообразностью оценивать выполнение разного рода трудно формализуемых критериев, часть которых, однако, поддаётся современным средствам верификации на моделях [7].

5. Долгоживущие языки программирования. Новые и долгоживущие ЯП, как правило, имеют мультипарадигмальный характер [14]. Параллельное программирование использует средства, характерные для разных парадигм [7]. Это определяет возможность трансформационного подхода к накоплению правильности программных решений при разработке и модернизации параллельных программ на разных ЯП в рамках общей СП. Следует особо отметить не столько сложность собственно ПП, но трудоёмкость отладки программ для разных многопроцессорных конфигураций, необходимость разработки методов верифицирующей компиляции и оптимизации программных компонентов, средств масштабируемой макрогенерации кода и автоматизируемых трансформаций программ с удостоверением сохранения свойств при их комплексации из ранее отлаженных компонентов, приспособленных к многократному применению в разных условиях [12].

Развитие ЯСП в настоящее время ориентировано на решение задач на основе общих библиотечных модулей, обеспечивающих эффективную организацию процессов, или подъязыков, допускающих многопоточное программирование. Существуют сотни функциональных языков программирования, ориентированных на разные классы задач ПП, дающих эффективную отладку программ. Это не исключает реальную практику ручного распараллеливания ранее отлаженных обычных программ, приведения их к виду, удобному для применения производственных систем поддержки параллельных вычислений. Значительная часть таких работ носит технический характер и заключается в систематической реорганизации структур данных, изменении статуса переменных и включении в программу аннотаций, сообщающих компилятору об информационно-логических взаимосвязях. Существенным ограничением результата ручного распараллеливания является не только опасность повторной отладки алгоритма, но и его избыточная зависимость от характеристик целевой архитектуры [7].

6. Учебные языки и системы программирования. Авторы считают, что пришло время изучать информатику и программирование, начиная с мира параллелизма [6]. Теперь трудно не заметить, что: выполнение «одинаковых» действий обладает разной длительностью; длительность реагирования информационной системы может зависеть от текущей ситуации; собственно выполнимость действий зависит от не всегда заранее известных условий; системы можно и нужно настраивать; ряд систем могут работать одновременно и влиять на работу друг друга. На практике видно, что существуют кэши, протоколы, верификаторы, резервное копирование, транзакции, «гонки» данных, «смертельное объятие» и многое другое. Известно об успехах любительской астрономии, перспективах GRID-технологий и «облачных» вычислений. Активизация, вербализация и формализация таких знаний образует основу для быстрого изучения средств и методов параллельного программирования, начало которому авторы пытается дать в экспериментальном курсе «Парадигма параллельного программирования» [2, 6, 9].

Интересно отметить появление новых архитектур, обладающих полным набором команд с условным исполнением. В этом процессе расширяется пространство решений сложных задач, модернизируются методы развития информационных систем (ИС) на основе компьютерных сетей и многопроцессорных комплексов. Полезно рассмотреть перспективы развития парадигм программирования, обусловленных изменением условий эксплуатации современных информационных систем, особенно связанных с повсеместным распространением сетевых технологий, меняющих критерии оценки качества программ и методы обеспечения надежности и производительности программирования. Две основные линии такого развития – разработка распределенных ИС и компонентное программирование.

Варьирование правил функционирования сетей допускает как асинхронную, так и синхронную организацию срабатывания действий, включая дозирование нагрузки и специализацию процессоров и распределение действий по потокам выполнения. Использование иерархических, многоуровневых, структурированных и расширяемых сетей обеспечивает моделирование практически любых, накопленных в ЯП техник программирования и представления структур данных.

Сколь ни сложен мир параллелизма, программистам предстоит его понять, освоить и создать его полноценную поддержку с помощью ЯСП!

При оценке образовательного значения парадигм появилась тенденция выделять функциональное, параллельное и императивно-процедурное программирование в качестве принципиальных, а логическое и объектно-ориентированное рассматривать как дополнительные, изучаемые в виде расширения принципиальных парадигм [16]. Не исключено, что такое выделение обусловлено зависимостью преподавания логического программирования и ООП от владения развивающейся областью знаний или приложений, усложняющих учебный процесс.

Заключение. Мультипарадигмальность долго живущих ЯП и тенденция XXI-го века по созданию новых мультипарадигмальных ЯП говорят о созревании единой ПП, объединяющей выверенные в практике механизмы программирования. Тем более обосновано формирование общей ППП и создание учебных ЯСП, поддерживающих раннее обучение программированию как параллельному программированию, что и предсказаывал Кеннет Айверсон, автор языка APL [4].

Рассматривая задачу обучения ПП как путь к решению проблемы адаптации программ

к различным особенностям используемых многопроцессорных комплексов и многоядерных процессоров, мы видим, что решение этой проблемы потребует развития не только средств и методов ясного описания семантики ЯСП, но и методики преподавания ПП. Тридцать лет назад проект академика А.П. Ершова по обучению школьников информатике слегка обидел профессионалов, убеждённых, что программирование – это занятие для людей с высшим образованием. Возможно, что идее опережающего освоения параллелизма не легко будет найти признание, но ее реализация возможна в контексте современных дистанционных технологий и массового распространения мобильных устройств.

СПИСОК ЛИТЕРАТУРЫ

1. Андреева Т.А., Ануреев И.С., Бодин Е.В., Городняя Л.В., Марчук А.Г., Мурзин Ф.А., Шилов Н.В. Образовательное значение классификации компьютерных языков // Прикладная информатика. 2009. №6 (24). С. 18–28.
2. Андреева Т.А., Городняя Л.В. Преподавание парадигм программирования //XXV Ежегодная международная конференция-выставка «Информационные технологии в образовании» (ИТО-2015).
3. Бек К. Экстремальное программирование. Питер. 2002. 224 с.
4. Бурдонов И.Б., Косачев А.С., Кулямин В.В. Теория соответствия для систем с блокировками и разрушениями. М. Физматлит. 2008. 412 с.
5. Воеводин В.В. Параллельные вычисления / В.В. Воеводин, Вл.В. Воеводин. СПб. БХВ-Петербург. 2002. 608 с.
6. Городняя Л.В. Образовательные проблемы параллельного программирования //XXV Ежегодная международная конференция-выставка «Информационные технологии в образовании» (ИТО-2015).
7. Городняя Л.В. О проблеме автоматизации параллельного программирования // В сборнике Международной суперкомпьютерной конференции «Научный сервис в сети Интернет: многообразие суперкомпьютерных миров». http://agora.guru.ru/abrau2014 Проверено: 8 июня 2015.
8. Городняя Л.В. Парадигма программирования: курс лекций. Новосибирск. РИЦ НГУ. 2015. 206 с.
9. 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.
10. Магариу Н.А. Язык программирования АПЛ. М.: Радио и связь. 96 с.
11. Палмер С.Р., Фелсинг. Дж.М. Практическое руководство по функционально ориентированной разработке ПО. М. Вильямс. 2002. 304 с.
12. Степанов Г.Г. Пути обеспечения переносимости программ и опыт использования системы СИГМА // Трансляция и преобразование программ. Новосибирск. ВЦ СО АН СССР. 1984. 9 с.
13. Хоар Ч. Взаимодействующие последовательные процессы. М. Мир. 1989. 264 с.
14. Хорстман К. Scala для нетерпеливых. ДМК пресс. 2013. 408 с. 300 экз. ISBN 978-5-94074-920-2, 978-0-321-77409-5.
15. Knoop J. Compiler Construction / 20th International Conference, CC 2011Held as Part of

the Joint European Conferences on Theory and Practice of Software, ETAPS 2011 Saarbrьcken, Germany, March 26 ––April 3, 2011 // Lecture Notes in Computer Science. Springer V2011. V. 6601. 330 p.

16. Peter Van Roy. The principal programming paradigms (2008). https://www.info.ucl.ac.be/~pvr/paradigmsDIAGRAMeng108.pdf Проверено: 8 июня 2015.

UDK 004.43: 042.4

ON THE PARALLEL PROGRAMMING PARADIGM

Lidia V. Gorodnyaya

A.P. Ershov Institute of Informatics Systems, Siberian Branch of the Russian Academy of Sciences 6, Acad. Lavrentjev pr., Novosibirsk 630090, Russia

e-mail: gorod@iis.nsk.su

Federal State Autonomous Institution of Higher Education «Novosibirsk National Research State University» 630090, Novosibirsk-90, 2 Pirogova Str.

Annotation. The article concerns the actual problem of the parallel programming paradigms. Importance of this topic comes from the steep increase in the number of new-generation programming languages oriented at application and development of modern information technologies.

Keywords: programming languages and systems, programming paradigms, computer languages definition methods, parallel programming; educational programming

References

1. Andreeva T.A., Anureev I.S., Bodin E.B., Gorodnyaya L.V., Marchuk A.G., Murzin F.A., Shilov N.V. Obrazovatel'noe znachenie klassifikatsii komputernyh jazykov [The educational value of the computer languages classification] // Prikladnaja informatika. = Applied Informatics. №6 (24). 2009. Pp. 18–28. (in Russian)
2. Andreeva T.A., Gorodnyaya L.V. Prepodavanie paradigm programmirovanija. [Education of programming paradigm] //XXV Ezhegodnaja mezhdunarodnaja konferencia-vystavka «Informatsionnye tehnologii v obrazovanii » (ITO-2015) = XXV The annual international conference-exhibition "Information Technologies in Education" (ITO-2015) (in Russian)
3. Beck K. Extremaljnoe programmirovanie [Extreme Programming Explained]. Piter. 2002. 224 p. (in Russian)
4. Burdonov I.B., Kosachev A.S., Kuljamin V.V. Teoria cootvetstvia dlja system s blokirovkami I razrushenijami [The compliance theory for systems with blocking and destruction] M. Fizmatlit. 2008. 412 p. (in Russian)
5. Voevodin V.V. Paralleljnye vychislenija [Parallel computing]. SPb. BHV-Peterburg. 2002. 608 p. (in Russian)
6. Gorodnyaya L.V. Obrazovateljnye problem parallelnogo programmirovanija. [Educational problems of parallel programming)] // XXV Ezhegodnaja mezhdunarodnaja konferencia-vystavka «Informatsionnye tehnologii v obrazovanii » (ITO-2015) = XXV The annual

international conference-exhibition "Information Technologies in Education" (ITO-2015) (in Russian)

7. Gorodnyaya L.V. O probleme avtomatizatsii parallelnogo programmirovanija. [About the problem of parallel programming automation] // V sbornike Mezhdunarodnoj superkompjuternoj konferentsii «Nauchnyi servis v seti Internet: mnogoobrazie superkompjuternyh mirov». http://agora.guru.ru/abrau2014 Provereno: 8.06.2015. (in Russian)
8. Gorodnyaya L.V. Paradigma programmirovanija: kurs lectsij [The programming paradigm: lectures] Novosibirsk. RIC NGU. 2015. 206 p. (in Russian)
9. Gorodnyaya L.V. Paradigmy parallelnogo programmirovanija v universitetskih obrazovateljnyh programmah i specializatsii [The parallel programming paradigms in university education programs and specialization] //Vserossijskaja nauchnaja konferentsii «Nauchnyi servis v seti Internet: reshenije boljshih zadach = Allrussian Science Conference "Scientific service in the Internet: solving of the large tasks". Novorossijsk – Moskva. 2008. Pp. 180-184. (in Russian)
10. Magariu N.A. Jazyk programmirovanija APL [The programming language APL]. M. Radio i svjazj =M. Radio and Communication. 96 p. (in Russian)
11. Palmer S.R., Felsing Dzh.M. Prakticheskoe rukovodstvo po funktsionaljno orientirovannoj razrabotke PO [A Practical Guide to Feature-Driven Development]. - M.: Viljams, 2002. - 304 p. (in Russian)
12. Stepanov G.G. Puti obespechenija perenosimosti program I opyt ispoljzovanija sistemy SIGMA [The ways of tolerability programs and the experience of using the system SIGMA] // Transljatsija i preobrazovanie programm = Translation and transformation program. Novosibirsk. VC SO AN SSSR. 1984. 9 p. (in Russian)
13. Hoar Ch. Vzaimodejstvujuzhie posledovateljnye process [Communicating sequential processes] M. Mir. 1989. 264 p. (in Russian)
14. Horstman K. Scala dlja neterpelivyh [Scala for the Impatient]. DMK press. 2013. 408 p. – 300 экз. – ISBN 978-5-94074-920-2, 978-0-321-77409-5. (in Russian)
15. Knoop J. Compiler Construction / 20th International Conference, CC 2011Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2011 Saarbrьcken, Germany, March 26 ––April 3, 2011 // Lecture Notes in Computer Science. Springer V2011. V. 6601. 330 p.
16. Peter Van Roy. The principal programming paradigms (2008). https://www.info.ucl.ac.be/~pvr/paradigmsDIAGRAMeng108.pdf Provereno: 8.06.2015.