УДК 519.681

В.Я. Иосенкин
Приднестровский государственный университет им. Т.Г. Шевченко,
г. Тирасполь, Молдова

Контекстно-ориентированное программирование

Рассматривается технология программирования, основанная на контекстной интерпретации лексем. Приводится обоснование актуальности и определены цели, задачи и объект исследования. Исследуются проблемы современных языковых средств и показывается необходимость применения изложенных принципов контекстной технологии, на основе которых разработана и на примерах представлена система контекстного программирования Esse. Делаются выводы о преимуществах и недостатках применения технологии и показывается эффективность применения технологии для представления знаний.

Введение

Узким местом современных технологий программирования остается объективная трудоемкость, интеллектуальная и технологическая сложность процесса программирования [1], [2].

Недостатком существующих форм представления кода является наличие семантического разрыва между языком программирования высокого уровня и математической моделью предметной области и между математической моделью предметной области и ее описанием на естественных языках. Семантическим разрывом называется феномен, определяемый как мера различия принципов, лежащих в основе различных уровней языковых и аппаратных средств вычислительных систем. Что это означает? Когда мы используем языки высокого уровня, у нас есть абстракции – типы данных (знак, целое, массив и т.д.), для каждого типа данных определяются операции, которые над ними применимы. Но в памяти компьютера эти объекты представляются одинаково, как последовательность байт, и процессор работает с некоторым набором операций, которые у него есть, сильно отличающимся от тех, что в языках программирования. И совершенно другими абстракциями (баланс, проводка, вентиль, шаблон, двигатель внутреннего сгорания) приходится оперировать при постановке задачи. То есть имеется смысловой, или семантический, разрыв между теми средствами, которые используются для представления каких-то решаемых задач, и теми средствами, что мы вынуждены использовать при реализации этих представлений.

Выделяют несколько основных видов семантических разрывов, из которых наиболее актуальными и затрагивающими всех пользователей являются разрывы:

– между математической моделью и программными языковыми средствами (устраняются прикладным программным обеспечением с помощью объектно-ориентированного, аспектно-ориентированного программирования [3], но они снимают только часть проблем, не затрагивая серьезно семантику и синтаксис языков программирования);
– между математической моделью и ее описанием на естественном языке (языки-посредники для записи алгоритмов не решают проблему, а только расщеп-

ляют ее на меньшие, поскольку необходимо оперировать еще и понятиями языка-посредника).

Их последствиями являются:

1) высокая стоимость программных средств, так как необходимо держать постановщика задачи, который преобразует то, чем нам требуется управлять, в математическую модель, ряд специалистов, которые эту математическую модель (абстракцию) смогут интерпретировать в терминах и операциях конкретной вычислительной системы, программистов;
2) сложность постановки задач;
3) низкая эффективность и надежность программных средств;
4) большой объем программного кода.

А единственным эффективным методом сокращения этих видов семантического разрыва видится приближение языков программирования к структуре и синтаксису естественных языков. Ограничением выразительных качеств известных языков программирования является бесконтекстность правил вывода, что не согласуется с явлением контекстной интерпретации слов в естественных языках и как следствие – уменьшает надежность программирования, устанавливает серьезный барьер между языком программирования и естественным языком.

Целью проводимого исследования является анализ языковых средств информационных систем и разработка технологии программирования для сокращения семантического разрыва между моделью предметной области и используемыми для ее описания формальными языками.

В соответствии с поставленной целью были определены следующие задачи исследования.

1. Анализ проблемы семантического разрыва в современных информационных системах.
2. Разработка модели контекстных языковых средств и методов их синтаксического анализа и компиляции.
3. Разработка технологии (а впоследствии методологии) контекстного программирования.
4. Реализация и исследование компилятора контекстного языка программирования с использованием разработанной технологии.

Объектом исследования являются формальные языки и методы их грамматического разбора и компиляции, а предметом исследования – механизм представления, передачи, обработки и использования знаний в современных информационных системах.

В результате исследования разработана оригинальная технология контекстного программирования [4], [5] и экспериментальным путем показана эффективность ее применения для представления знаний [6], [7].

Исследование современного состояния языковых средств

Существующие технологии объектно-ориентированного программирования реализуются в рамках языков программирования, порождаемых узким классом контекстно-свободных (КС) формальных грамматик. Произвольное понятие предметной области, описываемое КС-языком (стековая переменная, функция, вызов функции, объявление переменной, арифметическое выражение и т.д.) [8], представляется как

последовательность терминальных и нетерминальных знаков (или как последовательность других понятий и слов), т.е. все понятия задаются абсолютно. Получается, что если есть понятие «вызов функции», то оно и остается понятием вызова функции независимо от его местоположения в тексте программы и относительно других понятий языка. Но в естественных языках все не так: различные понятия имеют различный смысл и даже разные определения в зависимости от контекста языка. Таким образом, выразительные свойства контекстно-свободных языков ограничены бесконтекстным использованием определенных понятий, где в качестве понятия используется нетерминальный знак языка.

Основной проблемой программирования на естественном языке является языковая неоднозначность, что связано с контекстной интерпретацией слов в естественных языках [9], решение проблемы – построение языка программирования на основе контекстной интерпретации языковых сущностей.

Для этого предлагается использовать не КС-языки, а контекстные, в которых каждое понятие зависит как от левого контекста, так и от правого, как и в естественных языках, и только в этом контексте оно заменяется на некое другое понятие. Каждое понятие может описываться несколькими контекстами, и в другом контексте оно может означать совсем не то, что в другом. Эти языки задаются контекстно-зависимой грамматикой. Например: при анализе текста встретилось слово «коса», понять смысл которого без начального и конечного текста невозможно (девичья коса, железная коса, речная коса).

Контекстный язык разрешим, то есть можно определить принадлежность строки языку [10, с. 22-23]. Но в них имеется одна проблема, связанная с большой трудоемкостью переборного алгоритма. Пока алгоритма эффективного разбора такого языка не существует. Но можно попытаться изменить или несколько упростить саму грамматику: брать только правый или только левый контекст. Так, рассмотренная ниже система контекстного программирования реализует левоконтекстную грамматику.

Принципы контекстной технологии программирования

В рамках объектно-ориентированной компонентной технологии предлагается использование контекстных языковых средств.

При полной поддержке принципов объектно-ориентированной методологии сформулируем принципы технологии программирования, где главным является принцип контекстной интерпретации слов:

1) основные синтаксические единицы языка (вместо структур и классов): понятия (определение словаря) nation и предложения (словарной статьи) sentence, вместе они составляют сущности языка essence;
2) четкое разделение программы на декларативную (структура сущностей – понятий и предложений), процедурную (определяет, как интерпретировать понятие в заданном контексте или предложении) и ситуационную (императивную) – собственно исполняемая часть;
3) все элементы языка (ключевые слова, операции, встроенные типы данных, управляющие конструкции) определяются по мере необходимости средствами языка. Для обеспечения наибольшей гибкости и универсальности необходим отказ от каких-либо ключевых слов, базовых операций, никаких

встроенных типов данных – ни сложных, ни простых, никаких управляющих конструкций. Все эти элементы определяются по мере необходимости средствами языка. Необходимо только правило определения элементов языковых конструкций;

4) использование разработанной формально-языковой модели представления знаний. Формально-языковая модель адаптирована для формализации знаний на естественных языках и описания модели предметной области с использованием программных средств, равных им по выразительной мощности. В ней используется похожий на используемый в продукционной модели механизм перебора правил [11], но учитывается также приоритет правил, а в правилах учитывается порядок следования фактов и есть дополнительный элемент – контекст. По форме определения понятий (иерархия понятий) много общего с семантическими сетями [12], но вместо отношений используется полное или частичное наследование и имеется опосредованная связь через предложения, дающая большую гибкость. Имеется только отдаленная аналогия структуры понятий со структурой фрейма [11], [13]. С формальными логическими моделями, основанными на классическом исчислении предикатов первого порядка, можно заметить сходство в определении операций (этим сходство ограничивается), где в роли предикатных символов выступает результирующий контекст, но для предлагаемой модели это только частный случай, когда результирующий контекст состоит из одного понятия;
5) важнейший принцип – принцип контекстной интерпретации слов: новые понятия определяются через ранее известные понятия путем указания тех конструкций предложения, в которых эти понятия могут быть использованы; более того, каждое такое предложение, содержащее описание этого понятия в контексте его применения, сопровождается его интерпретацией в ранее определенных конструкциях – предложениях.

Все эти свойства языковых средств, взятые по отдельности, имеются и в других современных системах программирования в той или иной степени. Новый взгляд на процесс программирования определяется тем, что эти черты рассматриваются в совокупности, как взаимосвязанные, и реализуются в комплексе, что вполне естественно, поскольку все указанные свойства дополняют друг друга.

Использование контекстной интерпретации слов встречается в современных языках программирования, но, как правило, эпизодически и непоследовательно. Например, в языке Perl каждая операция и каждый терм вычисляются в определенном контексте, который определяет поведение операции и интерпретацию возвращаемого ею значения. В нем существует два основных контекста – скалярный и списковый, а также специфические – булевый, void и подстановочный [14], [15]. Переопределять контекст или задавать пользовательский контекст нельзя.

В языке M[UMPS] тип значения вершины дерева не декларируется. Строковая, числовая или логическая интерпретация значения определяется контекстом операций в выражениях [16], [17].

В JavaScript также существуют три типа исполняемого кода, называемых контекстом исполнения, – глобальный, локальный и eval-контекст. Область действия переменной определяется положением ее декларации в тексте программы, а именно контекстом исполнения [18].

Перегрузка операций в языке C++ также представляет собой контекстную привязку этих операций к типам данных, с которыми они работают, так может быть определена операция ‘+’ для типа Complex и Salary, например, и выполняемые действия над объектами этих типов будут отличаться в зависимости от того, какой из типов находится в контексте этих операций. Однако количество таких операций ограничено, количество элементов контекста для каждой операции строго фиксировано, задавать пользовательские операции нельзя.

В качестве вычислительного механизма использован магазинный автомат, имеющий 2 (3) стека: стек операндов, стек адресов возврата (стек контекста). Использование третьего стека позволяет реализовать динамическое связывание при произвольном контексте и произвести динамическую генерацию кода. При миграции кода подгрузка дерева прикладных словарей осуществляется в заданную точку базовой иерархии.

Вычислительный механизм будет выглядеть, как показано на рис. 1.

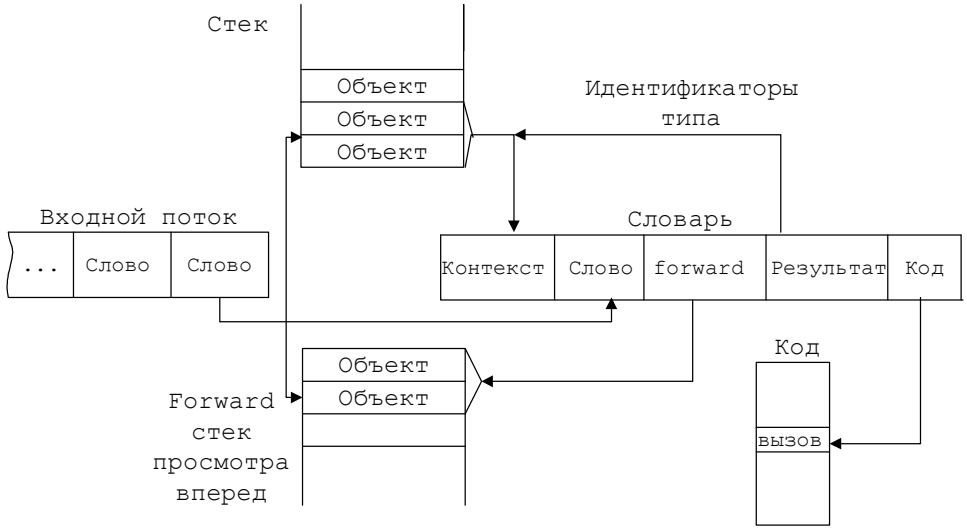

Рисунок 1 – Вычислительный механизм контекстного языка

Некая конструкция поступает на обработку виртуальной машине. К этому моменту в стеке контекста находится некий контекст слова, виртуальная машина осуществляет поиск по дереву словарей данного <имени> с контекстом из стека контекста. В случае успеха выполняется указанный в конструкции код (изменяется стек операндов) и в стек контекста подгружается результат. В случае неуспеха выдается предупреждение о неправильности применения синтаксической конструкции. Стек просмотра вперед используется для сопоставления конструкции forward, представляющей собой последовательность терминальных и нетерминальных знаков предложения, входному потоку**.**

Система контекстного программирования Esse

На основе разработанной контекстной технологии программирования была разработана система контекстного программирования (СКП) Esse (название происходит от англ. essence – сущность и esse – короткое литературное произведение) [19], правила определения элементов языковых конструкций которой представлены в [20].

Esse позволяет приблизить процесс проектирования программных средств к парадигмам постановки и решения прикладных задач и предлагает гибкие возможности в области представления знаний в естественно-языковой форме с использованием контекстной привязки терминов исследуемой предметной области.

Компилятор Esse имеет ряд необходимых в рамках исследуемой технологии черт: просмотр вперед для сопоставления правого контекста и проверки синтаксиса, расширенный перебор с возвратами, автоматическое создание локальных переменных, механизм реперных точек, компиляция на стековую машину, отсрочка компиляции кода до момента исполнения в ситуации, когда необходимую для порождения кода информацию трудно извлечь во время компиляции, но легко получить во время исполнения, нет ключевых слов и встроенных типов данных [21].

В соответствии с принципами иерархии и наследования объектно-ориентированной методологии каждое понятие находится в отношении эквивалентности или подобия с другим понятием, называемым родительским. То есть в результате формирования модели предметной области формируется дерево понятий, находящихся в родственных связях. На вершине этого дерева должно находиться понятие, которое не имеет родителя, оно называется корневым понятием и является эквивалентным только самому себе.

Результатом работы интерпретатора-компилятора Esse является промежуточный код виртуальной машины, которая может быть реализована на любом языке и адаптирована под конкретную операционную систему и платформу. На рис. 2 показаны некоторые базовые операции виртуальной машины Esse, их состав не играет принципиальной роли, главное – необходимо, чтобы они представляли функционально полный базис операций для обеспечения возможности реализации любых необходимых команд.

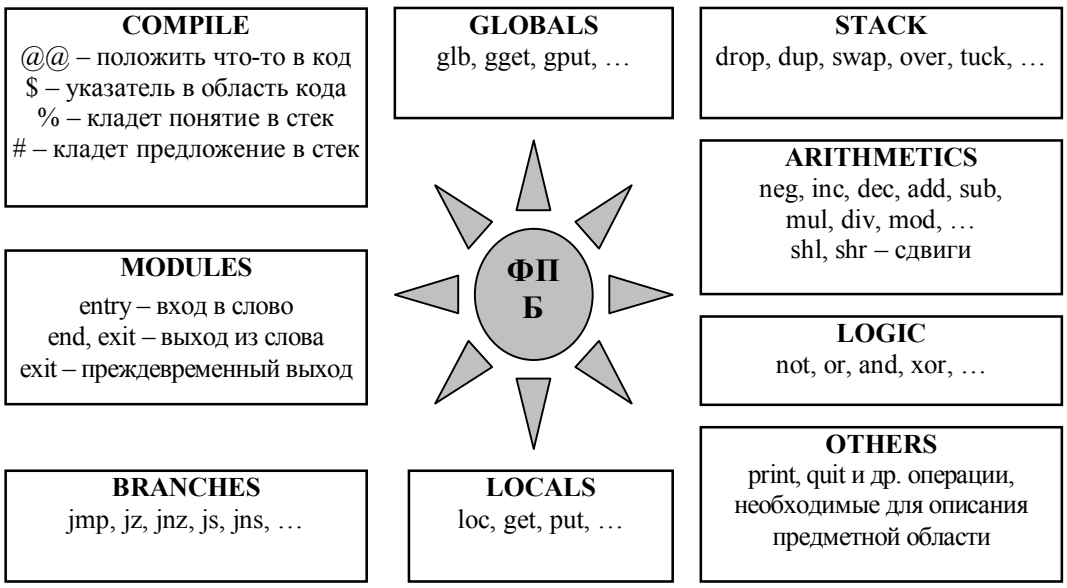

Рисунок 2 – Используемые базовые команды виртуальной машины Esse

Рассмотрим пример определения булевых операций на СКП Esse:

```
logical is global:
    'boolean' name {global %0 is logical};
```

```
        logical, boolean {%0 get};
        logical 'is' boolean {%1 %0 put}.
logic as logical:
          logic, boolean {};
        'false' ['0' @], logic;
        'true' ['1' @], logic;
        '(' boolean as a ')', logic {a}.
negation is logic:
        'not' negation, negation {%0 not}.
conjunction is negation :
        negation 'and' negation, conjunction {%0 %1 and}.
disjunction is conjunction :
        conjunction 'or' conjunction, disjunction {%0 %1 or}.
equaling is disjunction :
disjunction as a 'equal' disjunction as b, equaling { a and b or not a and not b }.
implication is equaling :
        equaling as a 'implicit' equaling as b, implication {not a or b}.
  boolean is implication.
```

Вначале определяется понятие logical, являющееся прототипом понятия «логическая переменная» языка Си. Результатом применения предложения «'boolean' name» является создание переменной типа 'boolean' (boolean x). Предложение «logical 'is'» boolean описывает конструкцию, которая присваивает ей логическое значение (x is true). А «logical, boolean» определяет конвертацию переменной в ее значение (фактически происходит получение значения с относительного адреса переменной). Понятие logic является первичным логическим понятием и определяет, что такое есть значение логической переменной (константы «истина», «ложь» или любое логическое выражение в скобках). Затем определяется понятие «отрицание» через понятие «logic» (т.е. отрицание – это слово not, после которого идет логическая константа или выражение в скобках), далее через отрицание определяется конъюнкция. Понятие слева от 'and' есть контекст применения данного слова, так как конъюнкция может быть применена, только если до этого стояло отрицание или первичное выражение. Если же реально там будет стоять, например, импликация, то будет обнаружена ошибка либо применено какое-то другое предложение. Аналогично определяются дизъюнкция, эквиваленция и импликация.

Определение выполняемых действий происходит в соответствии с приоритетом операций, чем позже операция определена – тем ниже приоритет [22]. В итоге импликация имеет самый низкий приоритет, эквиваленция – более высокий, отрицание – еще выше, и самый высокий приоритет у выражений в скобках и констант, – все так, как и должно быть. То есть новые понятия определяются через ранее известные понятия путем указания тех конструкций предложения, в которых эти понятия могут быть использованы с учетом контекста, описываемого также через ранее определенные понятия.

Рассмотрим код понятия «эквиваленция»: {a and b or not a and not b}. Как можно заметить, этот код есть не что иное, как ранее определенная дизъюнкция двух ранее определенных конъюнкций, т.е. эквиваленция интерпретируется через ранее определенные понятия – дизъюнкцию, конъюнкцию и др., а значит, в нем используются конструкции этих понятий, соответственно работает и приоритет операций.

То есть указанный код равносилен коду: «a b and, a not, b not, and, or» без применения последних определенных конструкций, но первый проще и понятнее. Преобразование этого кода к первому осуществляется компилятором автоматически.

На примере описания функции условного перехода if-then-else видно, насколько длинные конструкции можно задавать в одном предложении.

Рассмотрим пример определения операции условного перехода if-then-else:

```
ifthenelse is block:
  'if' boolean 'then' ['0' @, 'jz' @] block " [$2 $1 sub inc $1 @@];
  'if' boolean 'then' ['0' @, 'jz' @] block
  'else' ['0' @, 'jmp' @] block " [$2 $1 sub inc $1 @@, $3 $2 sub dec $2 @@];
```

Помимо автоматического создания локальных переменных автоматически также создаются указатели на счетчик команд, будем называть их реперными точками [21]; они позволяют легко реализовывать возврат на заданное место в генерируемом коде, например для того, чтобы его повторить или, наоборот, пропустить. Реперные точки задаются в форме $x, где x – номер реперной точки, создающейся автоматически при появлении в предложении терминальных знаков или " (пустого терминального знака). Этот механизм позволяет избежать необходимости применять адресные команды.

Таким образом, можно с помощью такой системы реализовать, например, словари языка Си, Pascal или даже Prolog, а затем, переключаясь между словарями, писать текст программы сразу на нескольких языках. Ниже представлен пример, задачей которого является распознавание предложений английского языка:

```
…
verbs as word:                    // Определение понятия «глаголы»
        main_verb, verbs;
        " main_verb, verbs;
        verbs "to|and|or" main_verb, verbs.
predicate as verbs:               // Определение понятия «сказуемое»
        " verbs, predicate;
        " modal_verb verbs, predicate.
// Задание структуры повествовательного предложения
statement as word:
        " subject predicate add_of_sentence, statement;
        'there' 'is' subject add_of_sentence, statement;
        " subject3 'is' add_of_sentence, statement;
        " personal_pronoun_2 primary_verb_2 add_of_sentence, statement;
        " personal_pronoun_1 primary_verb_1 add_of_sentence, statement.
sentence as word:
        " short_statement, sentence;
        " imperative, sentence;
        " special_question, sentence;
        " yes-no_question, sentence;
        " statement, sentence;
        " statement-, sentence;
        yes-no_question 'Yes' answer_statement, sentence;
        yes-no_question 'No' answer_statement-, sentence;
        interjections " sentence, sentence;
```

```
        interjections '!' sentence, sentence;
        statement " conjunction sentence, sentence.
english is sentence.
```

Так, после написания части кода на Си, можно, сменив словарь, писать на естественном английском языке.

Самое приятное, можно написать на СКП Esse свой собственный словарь для описания и управления какой-либо производственной системой, где после задания исходных данных управление объектами будет происходить автоматически, без вмешательства человека. Единственное, что необходимо – чтобы система своевременно получала информацию о текущем состоянии всех систем или событиях, а реакцию на них будет определять и выполнять сама программа [7].

Преимущества и недостатки контекстных языковых средств

Недостатком контекстных языковых средств является снижение производительности (по сравнению с их КС-аналогами) и как следствие – повышенные требования к аппаратному обеспечению (поскольку необходимо обрабатывать конструкции языка с учетом контекста, в результате увеличивается дерево перебора по сравнению с КС-языками). Для повышения производительности предлагается использовать ряд приемов.

1) Метод динамической кодогенерации, разработанный Михаэлем Францем [23], – компилятор как бы разрезается на две части – «препроцессор» (front-end) и «постпроцессор» (back-end). Как известно, на первой фазе компиляции (анализе) обычно производится сканирование исходного текста и формирование синтаксических деревьев [24]. На второй фазе (синтезе) осуществляется оптимизация и генерирование объектного кода. Так вот, Франц предложил заканчивать компиляцию на первой фазе, а результат сохранять в виде особым образом (на основе семантического словаря) сжатых файлов (тонкий бинарный код). Вторая же часть компилятора соединяется с загрузчиком.
2) Поиск слова для его исполнения происходит только в пределах того словаря, на который в данный момент ориентирована система.
3) Применение стековой архитектуры [25].
4) Ограничение ширины поиска. Результаты экспериментов определили, что вполне достаточна ширина в 7 – 8 слов. Большую глубину связи человеку просто тяжело охватить в памяти (в силу ограничений, связанных с особенностями мыслительной деятельности человека), поэтому маловероятно и использование таких конструкций; если такое и встретится, то скорее из-за неправильно или неконкретно заданных связей между объектами при их описании.

Достоинства технологии контекстного программирования:

- уменьшение сложности и повышение надежности программного обеспечения;
- сокращение сроков разработки;
- гибкая модификация отдельных частей ПО без изменения остальных;
- повторное использование кода;
- поддержка мультиязычности;
- внедрение и использование естественно-языковой нотации [26];
- повышение гибкости языка;

- применение методов динамического связывания ко всем, в том числе примитивным объектам;
- упрощенное описание моделей объектов и ситуаций и адекватная немедленная реакция на поступившие изменения;
- облегчение диалога между информационной системой и пользователем;
- разграничение описаний интерфейсной и реализационной части модуля;
- поддержка динамической генерации кода;
- поддержка контекстной интерпретации слов;
- сокращение расходов на разработку и сопровождение ПО [27].

Применение контекстной технологии позволит разрабатывать хорошо структурированные, надежные в эксплуатации, достаточно просто модифицируемые программные системы.

Выводы

Использование контекстной технологии программирования позволяет сократить семантический разрыв между математической моделью и языковыми программными средствами и между моделью и ее описанием на естественных языках, уменьшить число ошибок программирования, улучшить условия контроля за безопасностью кода, повысить надежность разработки программ.

Применение технологии исполнения мигрирующего кода в рамках контекстных языков также позволит сократить интенсивность обмена в глобальной сети путем предварительного согласования высокоуровневых протоколов для актуализации взаимодействия в терминах, в которых формулируется решение задачи на клиентской стороне, и породить быстрый и компактный код [28].

Литература

1. Lewis T. Software Architectures: Divine plan or digital Darwinism // Computer. – 1996. – № 8. – P. 13-15.
2. Canceled Software Development Project costs Billions // Computer. – 1995. – № 8. – P. 94.
3. Kiezales G., Lamping S., etc. Aspect-Oriented Programming // Proc. of the European Conference on Object-Oriented Programming (ECOOP'97). – Finland, Springer Verlag, INCS 1241, June 1997.
4. Иосенкин В.Я., Выхованец В.С. Формализация семантики искусственных языков // Мат-лы междунар. науч.-практ. конф. «Математическое моделирование в образовании, науке и производстве». – Тирасполь: РИО ПГУ. – 2001. – С. 477-480.
5. Iosenkin V. The technology of context-sensitive programming in the telecommunication systems // Мат-лы междунар. науч.-практ. конф. «Развивающие интеллектуальные системы автоматизированного проектирования и управление». – Ч. 3. – Новочеркасск: УПЦ «Набла» ЮРГТУ(НПИ). – 2001. – С. 19.
6. Иосенкин В.Я., Выхованец В.С. Контекстное программирование в моделировании // Мат-лы междунар. науч.-практ. конф. «Моделирование. Теория, методы и средства». – Часть 7. – Новочеркасск: УПЦ «Набла» ЮРГТУ(НПИ). – 2001. – С. 49-51.
7. Иосенкин В.Я., Выхованец В.С. Контекстная модель технологического процесса предприятия // Труды II междунар. конф. «Идентификация систем и задачи управления» (SICPRO'03). – М.: Институт проблем управления им. В.А. Трапезникова РАН. – 2003. – С. 859-871.
8. Гинзбург С. Математическая теория контекстно-свободных языков. – М.: Мир, 1970. – 326 с.
9. Entry Natural Language Understanding // Encyclopaedia of Artificial Intelligence. – P. 660-677.
10. Выхованец В.С. Теория автоматов: Учеб. пособие для вузов. – Тирасполь: РИО ПГУ, 2001.
11. Справочник по искусственному интеллекту / под ред. Д.А. Поспелова – М.: Радио и связь, 1990. – Кн. 2: Модели и методы.
12. Соломатин Н.М. Информационные семантические системы // Перспективы развития вычислительной техники. Учеб. пособие: В 11 кн. – М.: Высшая школа, 1989. – Кн. 1.
13. Джексон П. Введение в экспертные системы. – М.: Издательский дом «Вильямс», 2001.

14. Матросов А., Чаунин М. Самоучитель Perl. – СПб, 2000.
15. Wall L., Christiansen T., Orwant J. Programming Perl. – O'Reilly, 2000.
16. Original Canadian National Standard for Language Specification (identical to ANSI X11.1-1995). NSC 11756-1995.
17. Revised ANSI Standard for Language Specification. ANSI/MDC X11.1-1995.
18. Лукач Ю.С. Справочник Веб-разработчика // http://allo.usaaa.ru/ wdh/contents.htm.
19. Иосенкин В.Я., Выхованец В.С. Применение технологии контекстного программирования для решения больших прикладных задач // Труды междунар. конф. «Параллельные вычисления и задачи управления» (PACO'2001). – М.: Институт проблем управления им. В.А. Трапезникова РАН. – 2001. – С. (4)-121-139.
20. Иосенкин В.Я., Выхованец В.С. Контекстная модель технологического процесса предприятия // Труды II междунар. конф. «Идентификация систем и задачи управления» (SICPRO'03). – М.: Институт проблем управления им. В.А. Трапезникова РАН. – 2003. – С. 859-871.
21. Выхованец В.С., Иосенкин В.Я. Компиляция знаний, представленных на языке Esse // Тез. докл. II междунар. конф. по проблемам управления. – Том 2. – М.: Институт проблем управления им. В.А. Трапезникова РАН. – 2003. – С. 165.
22. Иосенкин В.Я. Качество и надежность проектирования программных средств // Труды междунар. симпозиума «Надежность и качество» / Под ред. Н.К. Юркова. – Пенза: Изд-во Пензенского гос. ун-та. – 2002. – С. 135-137.
23. Franz M. Code-Generation On-the-Fly: A Key to Portable Software: Doctoral Dissertation № 10497. – ETH Zurich, 1994.
24. Kistler Th., Franz M. A Tree-Based Alternative to Java Byte-Codes // University of California at Irvine. Dept. of Information and Computer Science. – 1996. – Technical Report № 96-58.
25. Иосенкин В.Я. Объектно-ориентированный Forth // Мат-лы междунар. науч.-практ. конф. «Региональные особенности развития машино- и приборостроения, информационных технологий, проблемы и опыт подготовки кадров». – Тирасполь: РИО ПГУ. – 2001. – С. 132-134.
26. Иосенкин В.Я. Контекстная интерпретация лексем // Мат-лы междунар. науч.-практ. конф. «Информационные технологии в науке и образовании». – Шахты: Изд-во ЮРГУЭС. – 2001. – С. 73-75.
27. Иосенкин В.Я., Выхованец В.С. Технология контекстного программирования в экономике и бизнесе // Мат-лы межвузовской электронной науч.-технич. конф. «Управляющие и вычислительные системы. Новые технологии». – Вологда: РИО ВоГТУ. – 2001. – С. 180.
28. Иосенкин В.Я., Выхованец В.С. Технология контекстного программирования в телекоммуникационных системах // Труды Второй междунар. науч.-практ. конф. «Современные информационные и электронные технологии». – Одесса: Друк. – 2001. – С. 118-119.

В.Я. Іосенкін
Контекстно-орієнтоване програмування
Розглядається технологія програмування, заснована на контекстній інтерпретації лексем. Наводиться обґрунтування актуальності і визначені цілі, задачі й об'єкт дослідження. Досліджуються проблеми сучасних мовних засобів і показується необхідність застосування викладених принципів контекстної технології, на основі яких розроблена і на прикладах представлена система контекстного програмування Esse. Робляться висновки про переваги і недоліки застосування технології та показується ефективність застосування технології для представлення знань.

V.Y. Iosenkin
Context-Oriented Programming
Programming technology is viewed based on context interpretation of lexemes. Substantiation of topicality is given and purposes, tasks and researching object are defined. Problems of modern lingual tool are researching and necessity of applying of given context-oriented technology principles is showed. With using of this principles system of context-oriented programming Esse was developed and represented with examples. Conclusions are drawn on merits and demerits of using technology and effectiveness of applying of technology for knowledge representation is illustrated.

Статья поступила в редакцию 17.06.2004.