КРАТКО О СОДЕРЖАНИИ ЖУРНАЛА «КОМПЬЮТЕРНЫЕ ИНСТРУМЕНТЫ В ОБРАЗОВАНИИ» № 6, 2009 ГОДА

1. ЧТО ТАКОЕ СОВРЕМЕННАЯ ПРОГРАММНАЯ ИНЖЕНЕРИЯ?

Рассказывая в статье о роли унифицированного языка программирования UML в индустрии программного обеспечения, авторы начинают с описания области, которая получила название «технологии программирования» или «инженерия программного обеспечения». Это очень поучительный материал как для учителей информатики, ведущих кружки по программированию, так и для ребят, увлекающихся программированием и планирующих сделать его своей профессией.

Авторы рассматривают технологию программирования в трех аспектах:

Модель процесса, то есть порядок проведения типового проекта по разработке программного обеспечения. Сюда относятся понятия жизненного цикла программного обеспечения, определение модели процесса – выделение в нем фаз, вех, потоков работ и других составляющих. Характерным для данного аспекта является рассмотрение на уровне программирующей организации в целом.

Модель команды, то есть отношения между людьми в процессе разработки. Сюда относятся определение обязанностей работников — участников процесса, регламенты их взаимодействия, рабочие процедуры и т. п. Характерным для данного аспекта является рассмотрение на уровне группы (команды) или проекта.

Дисциплина программирования, то есть методы создания конкретных артефактов, входящих в состав программного обеспечения. Сюда относятся описание и применение образцов проектирования, стандарты кодирования, методы тестирования и отладки и т. д. Характерным для данного аспекта является рассмотрение проекта на уровне отдельного работника.

(Д.Ю. Иванов, Ф.А. Новиков. «Влияние UML на процесс разработки программного обеспечения», с. 3–11)

2. ЧТО НАЗЫВАЮТ ОБЪЕДИНЕННЫМИ ВИДАМИ В ПРОГРАММИРОВАНИИ?

Об этом можно узнать из статьи, которая заключает серию, посвящённую проекту реализации алгоритмических языков на базе объектно-ориентированного описания семантики. В статье рассматриваются некоторые нерешённые вопросы реализации семантики алгоритмических языков на примере языка Алгол 68.

В языке Алгол 68 не существует никаких конструкций, исполнение которых непосредственно даёт значения объединённых видов. Значения таких видов возникают только в результате приведений, называемых объединениями.

Объединение есть один из способов приведения видов. Объединение не изменяет вида значения, выдаваемого конструкцией во время счёта, а просто увеличивает свободу его использования. Такое значение должно быть приемлемо не только для одного какого-то вида значения конструкции, но для целого множества видов. Однако после объединения

это значение может использоваться примитивным действием только после динамической проверки его сопоставляющим предложением.

Чтобы лучше понять смысл термина, рекомендуется внимательно изучить пример:

***union* (*bool, char*)** *t, v; t* := *"a"*; *t* := true; *v* := *t*

(Б.К. Мартыненко. «Учебный исследовательский проект реализации алгоритмических языков: объединения, сопоставляющие предложения и окружения», с. 12–21)

3. ЧТО БУДЕТ ДЕЛАТЬ ПРОГРАММИСТ БУДУЩЕГО: ПИСАТЬ ПРОГРАММЫ ИЛИ ИЗОБРЕТАТЬ ЯЗЫКИ?

Существует большое количество языков программирования. Многие из них являются языками общего назначения, то есть языками, на которых можно написать любую программу для весьма широкого круга задач. Помимо этих языков, уже достаточно давно известны языки, специализированные под конкретную предметную область, но непригодные для решения задач из других предметных областей. Такие языки называются языками предметной области, Domain-Specific Languages, или, сокращённо, DSL.

Примерами существующих известных DSL могут служить такие языки, как SQL (язык запросов к базам данных), Prolog, который исключительно удобен для решения определённого круга задач, связанных с использованием средств логического вывода; также в какой-то мере языком DSL можно считать небольшой язык регулярных выражений, использующийся для поиска подстрок во многих языках общего назначения. Во многих визуальных средствах существуют языки описания содержимого экранных форм и расположения в них элементов управления (например, такой язык был в Delphi, в котором окно и его содержимое описывались в отдельном от кода программы файле; именно этот файл редактировался визуальным редактором Delphi, но можно было его редактировать и руками).

Почему же DSL, которые, казалось бы, удобнее библиотек – намерения программиста в них выражаются короче, яснее и в терминах предметной области – распространены весьма ограниченно?

(К.С. Конопко. «Некоторые проблемы создания и использования DSL и как они решаются в среде разработки JetBrains MPS», с. 22–30)

4. КАК ПРОСТЫЕ СРЕДСТВА ВИЗУАЛИЗАЦИИ ПОМОГАЮТ КЛАССИФИЦИРОВАТЬ ПРОФЕССИОНАЛЬНУЮ ИНФОРМАЦИЮ?

Получение изображений, соответствующих оригиналам, есть одна из важнейших задач оптики. Она решается с помощью разнообразных оптических приборов и систем, для создания которых, в свою очередь, используется огромное количество специальных оптических стекол. Все эти стекла, которых только в России насчитывается порядка четырех сотен, обладают уникальными рефрактометрическими, физическими и химическими свойствами.

Изучение «сухих» цифр, характеризующих параметры этих стекол, – не очень интересная задача. В статье представлен продукт, обладающий удобным графическим интерфейсом, возможностью интерактивного взаимодействия и звуковым сопровождением, помогающий изучить характеристики оптических стекол.

(Е.Е. Селявка, С.К. Стафеев. «Использование современных программных продуктов для изучения рефрактометрических свойств оптических материалов», с. 31–35).

5. КАК КРАСИВО ПРЕДСТАВИТЬ МАТЕМАТИЧЕСКИЕ ФОРМУЛЫ НА HTML-СТРАНИЧКЕ?

Речь в статье идет о типографской системе под названием TeX. Как известно, TeX был изобретен Дональдом Кнутом. Основным поводом для создания системы стало недовольство тем качеством, с которым публиковались статьи математического характера.

Препятствием к широкому использованию системы TeX стала необходимость выучивания большого количества специальных правил для разметки текста (правда, многие выходили из положения простым использованием шаблонов).

На самом деле сегодня нет необходимости вникать во все тонкости языка разметки. Для TeXa создана удобная графическая оболочка LyX (http://lyx.org). Программа, в отличие от редакторов типа Microsoft Word, не следует концепции WYSIWYG (What You See Is What You Get), а исповедует концпецию WYSIWYM (What You See Is What You Mean). То есть пользователь по-прежнему работает с логической разметкой документа, а конечный вид документа будет сформирован компилятором TeXa. Достоинство такого инструмента в том, что автору не нужно запоминать множество различных команд. Он работает в привычном графическом окружении.

(А.М. Магляс. «Использование LaTeX для Интернет-публикаций», с. 36–38)

6. ЧТО ВЫ ЗНАЕТЕ О ФУНДАМЕНТАЛЬНЫХ ПРИНЦИПАХ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ?

Фундаментальными принципами объектно-ориентированного программирования являются инкапсуляция (ограничение доступа к членам класса для предотвращения несанкционированного их использования) и наследование (возможность определить новый класс, основанный на существующем классе и имеющий доступ к членам базового класса). Третьим фундаментальным принципом объектно-ориентированного программирования является полиморфизм, суть которого заключается в следующем. Можно определить несколько функций с одинаковыми именами, но различным поведением. Поведение каждой из указанных функций определяется данными, к которым она применяется.

Данная статья является шестой из серии статей, посвященных изложению «нулевого уровня» языка C#. Рассматриваются следующие приемы программирования на языке C#: использование механизма наследования, определение и использование виртуальных методов, интерфейсов, абстрактных классов и методов, назначение и объявление пространства имен, перегрузка операторов, определение и использование свойств и индексаторов.

(Л.А. Керов. «Дополнительные приёмы программирования на языке C#», с. 39–57)

7. КАК ОЦЕНИВАЕТСЯ СЛОЖНОСТЬ АЛГОРИТМОВ ИЛИ ЧТО ВЫ УЗНАЕТЕ О ПРОБЛЕМЕ P ≠ NP НА МИЛЛИОН ДОЛЛАРОВ, ИГРАЯ В «САПЁР»?

Проблема определения трудности решения математических задач возникла практически одновременно с возникновением математики. Человечество всегда волновал вопрос, почему одни задачи решаются легко, а другие с трудом; один человек мог решить задачу, а другой нет. Другой важнейший вопрос, связанный с трудностью задач, может быть сформулирован так: если задачу не удаётся решить, то существует ли у неё решение.

С появлением реальных электронных вычислительных машин разработки по теории алгоритмов стали особенно актуальны и фактически обеспечили начало современной информационной революции. На часть вопросов, которые исследовались ранее, ответы были получены, но возникли новые более глубокие проблемы, связанные с решением задач и вычислимостью функций. Одной из других важных проблем является оценка сложности алгоритмов, о которой и идет речь в статье.

В работе на основании теории сложности задач показано, что если алгоритм раскрытия некоторого поля игры «Сапер» существует, то он имеет сложность NP. Результаты экспериментального исследования игры показали, что алгоритм раскрытия существует с некоторой вероятностью. Этот факт позволяет предложить новую классификацию сложности задач.

В приложении приводится описание методов оценки результатов в международных спортивных соревнованиях по игре в «Сапёр».

(А.Ф. Ляхов. «Трудно решаемые задачи и игра «Сапёр»», с. 58–68)