УДК 681.3.07

А.Г. ЧУХРАЙ

Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Украина

ОБ ОДНОМ МЕТОДЕ ПРОВЕРКИ ПРОФЕССИОНАЛЬНЫХ УМЕНИЙ АЛГОРИТМИЗАЦИИ

Изложен метод автоматической проверки правильности профессиональных умений алгоритмизации, базирующийся на принципах рационального тестирования промышленного программного обеспечения. Суть метода заключается в сопоставлении результатов работы эталонного алгоритма, разработанного педагогом, и реального алгоритма, разработанного студентом, посредством ряда автоматических тестов. Приведена структурная схема метода, отображающая последовательность его этапов и связи между ними. Описаны особенности реализации метода в среде Borland Developer Studio 2006 и перспективы его развития.

Ключевые слова: *умения алгоритмизации, автоматическая проверка умений, необходимые и достаточные условия проверки.*

Введение

Неотъемлемым требованием, предъявляемым к специалистам с высшим техническим образованием, сегодня стали умения разрабатывать алгоритмы и программное обеспечение для автоматизированного или автоматического решения задач, диктуемых потребностями современного наукоемкого производства. Для обучения таким умениям, впрочем, как и многим другим инженерным умениям, требуется индивидуальный подход со стороны педагога и мотивированный кропотливый труд студента. Вместе с тем традиционное массовое обучение во ВТУЗах, даже при наличии очень хорошего педагога, характеризуется принципиальным недостатком – невозможностью адаптироваться к каждому студенту. И здесь на помощь могут прийти интеллектуальные компьютерные обучающие программы. История создания и внедрения компьютерных обучающих программ насчитывает уже около 50 лет. К настоящему времени созданы различные системы, обучающие, в том числе, и IT-умениям [1 – 3]. Тем не менее, задача создания интеллектуального компьютерного педагога, способного найти и реализовать эффективный индивидуальный подход к каждому обучаемому, все еще далека от своего окончательного разрешения [4, 5].

В статье представлена первая часть одного из перспективных подходов к компьютерному обучению профессиональным умениям алгоритмизации: описан метод проверки правильности этих умений. В качестве идейной основы для разработки послужили методы и принципы рационального тестирования промышленного программного обеспечения [6].

Постановка задачи. На основе принципов и методов рационального тестирования промышленного программного обеспечения сформировать метод проверки правильности профессиональных умений алгоритмизации.

Описание метода

В настоящее время существует два наиболее широко используемых и в то же время часто противопоставляемых подхода к разработке программного обеспечения: подход, основанный на алгоритмической декомпозиции, и подход, основанный на объектно-ориентированной декомпозиции. Несмотря на ощутимые преимущества последнего с точки зрения повторного использования, сопровождаемости кода и возможностей эффективного абстрагирования, востребованных при разработке сложного программного обеспечения, можно утверждать следующее: 1) подпрограмма – суть метод класса (объекта), поэтому в топологически отсортированном учебном плане дисциплины, связанные с выработкой умений алгоритмической декомпозиции, предшествуют объектно-ориентированному анализу, проектированию и программированию; 2) во многих реальных задачах, характеризующихся жесткими ограничениями на быстродействие и ресурсы, единственной альтернативой остается алгоритмическая декомпозиция.

Поэтому в данном исследовании базовым компонентом умений считается основной компонент алгоритмической декомпозиции – подпрограмма.

Для конструктивизма и в то же время без потери общности дальнейших рассуждений рассмотрим суть излагаемого метода, представленного на рис. 1, на примере подпрограммы, сигнатура которой включает кортеж арности $n \in N \cup \{0\}$ параметров

$$s(x_1 : type\_x_1; x_2 : type\_x_2; ...; x_n : type\_x_n) : type\_s, \quad (1)$$

где $x_1, x_2, ..., x_n$ – входные параметры типов $type\_x_1, type\_x_2, ..., type\_x_n$; $type\_s$ – тип результирующего значения подпрограммы.

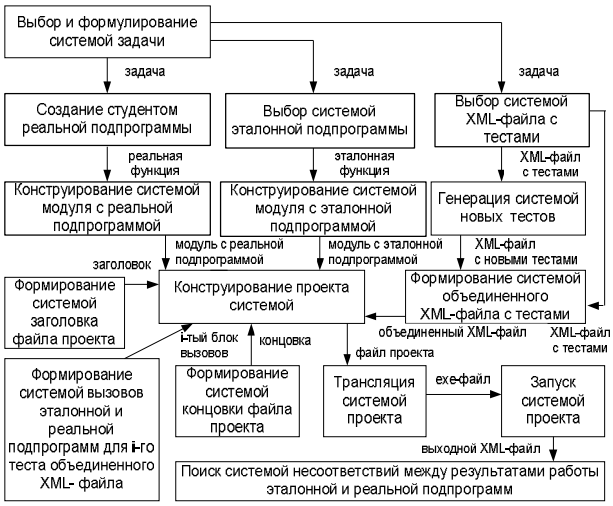

Рис. 1. Схема метода

Пусть практическая задача, требующая алгоритмического решения, описывается четверкой

$$Task = (Descr, Req\_Input, Req\_Output, Tests), \quad (2)$$

где Descr – вербальная постановка задачи, состоящая из набора строк;

$Req\_Input = (type\_x_1, type\_x_2, ..., type\_x_n)$ – требования к типам входных параметров;

$Req\_Output = (type\_s)$ – требования к типу выходного параметра;

$Tests = \{ t_1 = (i_{11} : type\_x_1, i_{12} : type\_x_2, ..., i_{1n} : type\_x_n, o_1 : type\_s), ..., t_k = (i_{k1} : type\_x_1, i_{k2} : type\_x_2, ..., i_{kn} : type\_x_n, o_k : type\_s)\}$ – множество мощности k кортежей, каждый из которых представляет собой тест, состоящий из конкретных значений входных параметров и значения выходного параметра.

Тогда необходимое условие того, что подпрограмма s является решением задачи Task, можно записать в виде

$$\forall t_j \in Tests : s(\Pi_{(i_{j1}, i_{j2}, ..., i_{jn})} t_j) = o_j .$$

Очевидно, что для сложных подпрограмм практически невозможна проверка на всевозможных значениях входных параметров, поэтому для разумной уверенности в достижении достаточного условия используют различные методы, например, разбивают диапазоны значений на классы эквивалентности, с особой тщательностью проверяют граничные значения, в том числе за пределами диапазона и т.д.

Утверждение. Пусть некоторая подпрограмма $s_1(i_1 : type\_i_1; i_2 : type\_i_2, ..., i_n : type\_i_n) : type\_o$ представима в виде кортежа арности z, $z \in N$ строк, т.е. $s_1 = (s_{11}, s_{12}, ..., s_{1z})$ и является решением некоторой задачи вида (2). Тогда возможно составить $y, y \in N, Y >> 1$ подпрограмм s_j' неэквивалентных s_1 и друг другу, но являющихся решениями этой задачи.

Доказательство этого утверждения очевидно из того факта, что во всякую подпрограмму можно добавить некоторую переменную или оператор, не влияющие на результат вычисления. Вместе с тем необходимость такого утверждения вызвана другим интуитивно понятным фактом: различных вариантов решения одной и той же задачи существует столько, сколько привлечено программистов. Отсюда, для проверки правильности подпрограммы, составленной студентом, необходимо сравнить результаты ее выполнения с результатами выполнения эталонной программы, составленной педагогом, при достаточно большом k.

В настоящее время реализован прототип программного обеспечения (ПО), реализующего приведенную схему в среде Borland Developer Studio 2006 для операционной системы Microsoft Windows XP Professional. Экранная форма системы приведена на рис. 2. Для динамической трансляции проекта использовался компилятор dcc32.exe.

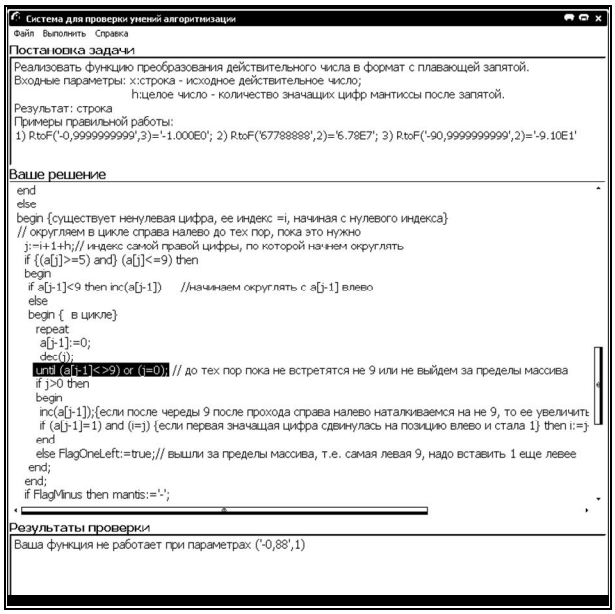

Рис. 2. Экранная форма системы

Отличительной особенностью разработки является ее универсальность, прежде всего, связанная с независимостью ПО от задач и их решений. Первые эксперименты – анализируемые системой студенче-

ские решения – показали, что необходимо отказаться от монолитной структуры проекта, содержащего вызовы студенческой функции с разными значениями входных параметров, в пользу многопоточного приложения, поскольку при "зависании" одного из тестов, например, из-за бесконечного цикла, "зависает" все приложение.

В свою очередь, в многопоточном приложении существует возможность корректной обработки зависания одного из потоков (конкретного теста), например, посредством использования функции WinApi WaitForSingleObject.

Второй параметр этой функции – допустимый временной интервал, по достижению которого поток снимается с выполнения путем вызова функции TerminateThread.

Заключение

Таким образом, разработан метод автоматической проверки правильности профессиональных умений алгоритмизации. В перспективе планируется расширение множества языков программирования, перенос системы в Web-среду, а также наделение ее интеллектуальной поддержкой обучения умениям алгоритмизировать.

Литература

1. Kumar A. A tutor for using dynamic memory in C++ / A. Kumar // IEEE Frontiers in Education Conference. – 2002. – Vol. 1. – P. 12-16.

2. Mitrovic A. An intelligent SQL tutor on the web / A. Mitrovic // International Journal of Artificial Intelligence in Education. – 2003. – Vol. 13. – P. 171-195.

3. Jurado F. Learning to program with COALA, a distributed computer assisted environment / F. Jurado, A. Molina, M. Redondo, M. Ortega // Journal of Universal Computer Science. – 2009. – Vol 15., No 7. – P. 1472-1485.

4. VanLehn K. The Behavior of Tutoring Systems / K. VanLehn // International Journal of Artificial Intelligence in Education. – 2006. – Vol. 16, Issue 3. – P. 227-265.

5. Diagnostic models of intelligent tutor system for teaching skills to solve algebraic equations [Электронний ресурс] / A. Kulik, A. Chukhray, M. Chukhray // In Proceedings of the Interactive Computer Aided Learning Conference, Villach, Austria, Sept. 26-28 2006, International Journal of Emerging Technologies in Learning.– 2007. – Vol. 2, № 1. – Режим доступу до журн. : http://www.i-jet.org.

6. Дастин Э. Автоматизированное тестирование программного обеспечения / Э. Дастин, Д. Рэшка, Дж. Пол. – «Лори», 2003. – 592 с.

Поступила в редакцию 6.11.2009

Рецензент: д-р пед. наук, проф., зам. директора С.А. Раков, Украинский центр оценивания качества образования, Харьков, Украина.

ПРО ОДИН З МЕТОДІВ ПЕРЕВІРКИ ПРОФЕСІЙНИХ ВМІНЬ АЛГОРИТМІЗАЦІЇ

А.Г. Чухрай

Викладено метод автоматичної перевірки правильності професійних вмінь алгоритмізації, який базується на принципах раціонального тестування промислового програмного забезпечення. Сутність методу полягає у порівнянні результатів роботи еталонного алгоритму, розробленого педагогом, та реального алгоритму, розробленого студентом, за допомогою ряду автоматичних тестів. Наведено структурну схему методу, яка містить послідовність його етапів та зв'язки між ними. Описано особливості реалізації методу в середовищі Borland Developer Studio 2006 та перспективи його розвитку.

Ключові слова: вміння алгоритмізації, автоматична перевірка вмінь, необхідні та достатні умови перевірки.

ABOUT ONE METHOD OF PROFESSIONAL ALGORITHMIC SKILLS TESTING

A.G. Chuhray

The method of automatic testing of professional algorithmic skills is described. It is based on the principles of rational testing of industrial software. The gist of the method is in results comparing of developed by teacher etalon algorithm and developed by student real algorithm by means of automatic tests. The structural scheme of the method is presented. It includes stages of the method and their relations. The features of method's implementation by means of Borland Developer Studio 2006 as well as perspectives of the method are stated.

Key words: algorithmic skills, automatic testing of skills, necessary and sufficient conditions of testing.

Чухрай Андрей Григорьевич – канд. техн. наук, доцент, доцент кафедры систем управления летательных аппаратов Национального аэрокосмического университета им. Н.Е. Жуковского «ХАИ», Харьков, Украина, e-mail: achukhray@gmail.com.