УДК 004.588:512

Л. Г. Алсынбаева, К. С. Алсынбаев

ФОРМАЛЬНЫЕ МОДЕЛИ ДЛЯ СИСТЕМЫ АВТОМАТИЗИРОВАННОЙ ГЕНЕРАЦИИ ТЕСТОВ ПО ПРОГРАММИРОВАНИЮ

Рассматриваются формальные подходы к созданию алгоритмов, реализованных в системе автоматизированной генерации тестовых заданий. Представлена структура программной системы генерации тестовых заданий по программированию. Описаны постановки задач и математические модели для некоторых модулей системы.

Formal approaches to creation of algorithms implemented in automated tests generation system are considered. Structure of a software system of programming tests generation is presented. Problem statements and mathematical models for some modules of the system are described.

Ключевые слова: математическая модель, тестовое задание, программирование, информатика.

Key words: mathematical model, test, programming, informatics.

Введение

Настоящее время характеризуется стремительным развитием информационных технологий, которые пронизывают все сферы жизни и деятельности общества. Растет потребность в специалистах, способных использовать существующие, а также создавать новые информационные технологии.

Вестник Балтийского федерального университета им. И. Канта. 2012. Вып. 10. С. 159 – 166.

Для специалиста в области информатики и программирования, создающего новые информационные технологии, «алгоритмизация» и «алгоритмическое мышление» служат базовым инструментом профессиональной деятельности и наиболее типичным мыслительным процессом.

Многолетняя практика авторов работы со студентами и слушателями сформировала убеждение, что профессиональные умения программиста базируются на наборах умственных умений, своего рода инструментов. Важнейший из них мы назовем «внутренний процессор», который предполагает умение «читать» фрагмент программы и воссоздавать в уме ее результат. Можно утверждать, что большинство программистов профессионально сформировались при разборе чужих программ и отладке собственных.

Один из эффективных способов формирования «внутреннего процессора» — «прокручивание кусочков программ „в уме“». По такому принципу создаются тесты по алгоритмике типа: «Что будет в результате работы данного фрагмента программы?».

Заметим, что данный тип тестов имеет двойную ценность:

а) формирование «внутреннего процессора» обучаемого;

б) контроль знания языка программирования высокого уровня.

Одно из требований использования тестов в учебном процессе — их множественность, то есть наличие большого, в идеале — неограниченного количества тестов по каждой тематике с адекватной сложностью каждой группы однотипных тестов. Выполнение данного требования позволит применять тесты при контрольном тестировании как в аудитории, так и в дистанционном варианте без опасений «списывания» или возможности поиска тестов с ответами в Интернете.

Создание контрольных измерительных материалов — актуальная, творческая задача в любой дисциплине, в то же время чрезвычайно трудоемкая при «ручной» подготовке. Разработка множества вариантов тестовых заданий требует высокой квалификации преподавателя и значительных затрат времени. Для повышения эффективности процесса подготовки тестов целесообразно создание технологии автоматизированной генерации тестовых заданий и ее программной реализации в виде инструментального средства, позволяющего преподавателю сгенерировать тестовые задания по основным разделам алгоритмизации и программирования на языке высокого уровня (Паскаль, Delphi, C, Java и другие).

В Югорском НИИ информационных технологий под руководством одного из авторов статьи, Л. Г. Алсынбаевой, проводилась разработка системы автоматизированной генерации тестовых заданий по алгоритмике [1, с. 131]. В качестве базового языка программирования для генерации тестовых заданий выбран Паскаль, который многие годы является одним из лучших инструментов для обучения основам алгоритмизации и программирования в рамках школьного курса «Информатика и ИКТ», а также используется в программах профессиональной подготовки на начальном этапе.

Программная система, позволяющая автоматически генерировать тестовые задания по указанным параметрам, состоит из нескольких модулей (по количеству изучаемых тем), имеющих единый интерфейс. В основе технологии автоматизированной генерации тестов заложен принцип «Один фрагмент программы — одно тестовое задание». Сгенерированные варианты тестовых заданий могут быть «отбракованы» вручную через пользовательский интерфейс системы, выгружены на диск в формате системы электронного обучения MoodleXML, в текстовых форматах. doc,.html,.rtf или распечатаны в виде вариантов тестов для бланочного тестирования. Тестовые задания генерируются в виде фрагментов программы на языке Паскаль.

На рисунке приведена структура системы, включающая перечень модулей. Разработка программы велась группой программистов Югорского НИИ информационных технологий и студентов Югорского госуниверситета. Для организации работы по написанию отдельных модулей требовалось создать формальное описание постановки задачи и заложенных в них алгоритмов.

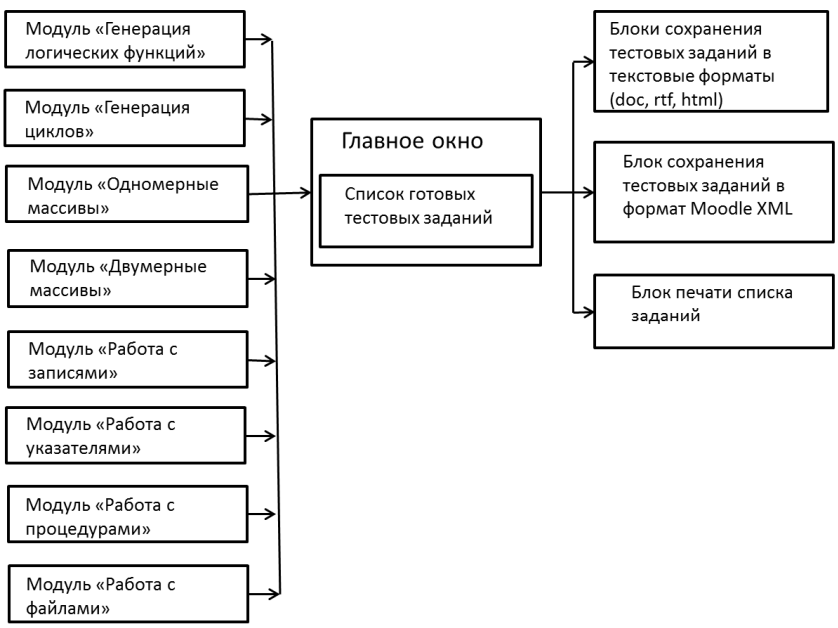

Рис. Структура системы автоматизированной генерации тестовых заданий

Для этой цели был использован формальный математический язык. Учитывая ограничения на размер журнальной статьи, приведем полное описание математических моделей для модулей «Генерация логических функций» и «Одномерные массивы» и краткую характеристику некоторых других моделей. Полное описание модели для модуля «Генерация циклов» приведено в [1, с. 131].

Формальная модель модуля «Генерация логических функций»

Постановка задачи. Требуется разработать программный модуль, обладающий следующей функциональностью:

1) генерация строк, являющихся корректным представлением записи логической (булевой) функции;

2) вычисление таблиц истинности для каждой сгенерированной булевой функции.

Интерфейс модуля должен позволять пользователю задавать количество переменных, от которых зависит функция, набор функциональных символов (логических операций), использующихся в аналитической записи булевой функции, длину записи аналитического представления булевой функции.

Математическая модель. Для описания математической модели генерации строк, являющихся корректным представлением булевой функции, использован раздел математики «Алгебра логики».

Согласно [2, с. 45], [3] и [4, с. 107] пусть $E = \{0, 1\}$, X — некоторое множество переменных, E_2^n — множество всех наборов $(\alpha_1, \ldots, \alpha_n)$ из нулей и единиц, $n \geqslant 1$. Булева функция $f^{(n)}(x_1, \ldots, x_n)$ от n переменных задает отображение E_2^n в E, $n \geqslant 1$, $x_1, \ldots, x_n$ — переменные из множества X, E_2^n — область определения, E — область значений функции.

Каждую булеву функцию можно представить в виде таблицы истинности. В [2] и других учебниках по дискретной математике даются понятия элементарных булевых функций и приводятся их табличные преставления.

В [3] дается понятие формулы над множеством функций $A = \{f_1(x_1, \ldots, x_{n_1}), f_2(x_1, \ldots, x_{n_2}), \ldots\}$. Формулы над A — это слова определенного вида (конечной длины) в алфавите $X \bigcup B \bigcup V$, где X — множество переменных; B — множество функциональных символов, соответствующих функциям из A; V — множество вспомогательных символов, состоящее из символов левой и правой скобок, а также запятой.

Каждой формуле назначается функция алгебры логики $f(x_1, \ldots, x_n)$. Функция $f(x_1, \ldots, x_n)$ — *суперпозиция* функций $g(y_1, \ldots, y_m)$ и $h_1(x_1, \ldots, x_n)$, $h_2(x_1, \ldots, x_n)$, …, $h_m(x_1, \ldots, x_n)$, если выполняется следующее равенство:

$$f(x_1, \ldots, x_n) = g(h_1(x_1, \ldots, x_n), h_2(x_1, \ldots, x_n), \ldots, h_m(x_1, \ldots, x_n)).$$

Используя элементарные булевы функции и операцию суперпозиции, можно сконструировать из элементарных булевых функций любую требуемую булеву функцию.

Для формирования таблицы истинности для каждой формулы $f(x_1, \ldots, x_n)$ требуется 2^n строк и $(n + 1)$ столбцов. В столбцах с 1-го по n-й формируются все возможные значения аргументов, принимающих значения 0 или 1, в столбце $(n + 1)$ вычисляются значения функции на основе таблицы истинности элементарных булевых функций, используемых на этапе генерации формулы путем их суперпозиции. Очевидно, в модуле должен присутствовать алгоритм интерпретации формул

для их вычисления. Возможно использование рекурсивного алгоритма [4, с. 112] или методики программной реализации теории графов, а именно обходы дерева, получающегося при генерации формулы.

Ниже приведены примеры булевых функций, сгенерированных в результате работы модуля системы «Генерация логических функций» на основе суперпозиции элементарных булевых функций.

```
n = 2, F(x1, x2) = ¬x1 & x2 → x2.
n = 3, F(x1, x2, x3) = x1 & ¬x2 v x2 & x1 → x3.
n = 3, F(x1, x2, x3) = x1 v ¬x2 & x1 & x3
```

В качестве второго способа автоматизированной генерации логических функций можно использовать алгоритмы восстановления конъюнктивной или дизъюнктивной нормальной формы функции по таблице истинности.

Этот подход, с точки зрения преподавателя, имеет плюсы в том, что, выполняя задания, студенты узнают еще одну форму представления булевых функций. Однако сгенерированные таким способом задания могут быть излишне громоздкими и преподаватель должен произвести отбор заданий наиболее целесообразных для включения в тест.

Для получения множества уникальных заданий при формировании таблицы истинности для значений логической функции, то есть столбца $(n+1)$, применяется генератор случайных чисел, который выдает значения 0 или 1. Уникальность обеспечивается большим количеством возможных функций [4, с. 108].

Формальная модель модуля «Одномерные массивы»

Постановка задачи. Требуется разработать программный модуль для генерации фрагментов программ на языке Паскаль, реализующих простейшие алгоритмы работы с одномерными массивами. Основной акцент в алгоритме обработки массива делается на формирование умения вычисления значений индексов элементов массива и извлечения значения элемента по его номеру. Алгоритмы должны производить манипуляции по перестановкам элементов массива, выбирая значения индексов из другого массива.

Интерфейс модуля должен позволять пользователю выбирать тип элементов массива: числовой (целый) или символьный, количество элементов массива, количество итераций цикла, в котором происходит обработка элементов массива, количество вариантов тестовых заданий.

Математическая модель. Для формального описания алгоритмов, демонстрирующих работу с одномерными массивами, использовано математическое понятие «перестановка» из области комбинаторики. Всякое расположение чисел 1, 2, …, n в некотором определенном порядке называется *перестановкой* из n чисел [4, с. 184; 5, с. 28], обычно трактуемое как *биекци*я (взаимно-однозначное соответствие) на множестве $\{1, 2, \ldots, n\}$. Число n при этом называется *порядком* перестановки.

Перестановка π множества X может быть записана в виде подстановки, например $\begin{pmatrix} x_1, \ldots, x_n \\ y_1, \ldots, y_n \end{pmatrix}$, где $\{x_1, \ldots, x_n\} = \{y_1, \ldots, y_n\} = X$ и $\pi(x_i) = y_i$.

Собственно, для генерации фрагментов алгоритмов перестановки местами элементов одномерного массива A мы можем воспользоваться подстановкой порядка n, где числа от 1 до n будут указывать индексы элементов массива A. Таким образом, требуется сконструировать подстановку π: $\pi(x_i) = y_i$ и оформить ее в виде программного кода на Паскале, изменяющего порядок элементов в некотором одномерном массиве A. При конструировании используется утверждение, что всякая подстановка представима в виде произведения транспозиций (обмен местами двух элементов перестановки) [5, с. 34] и любое произведение транспозиций является подстановкой.

Ниже приводится описание процесса построения функции π, которая преобразует исходный набор значений элементов массива в результирующий.

Пусть n — размерность исходного массива A, m — количество перестановок элементов в массиве A (соответствует также числу итераций цикла).

Значения массива A генерируются случайным образом. Перестановка элементов массива A производится при участии элементов вспомогательного массива B в качестве индексов элементов массива A. Размерность массива B соответствует количеству итераций цикла, умноженному на 2, и каждая последовательная пара элементов массива B определяет транспозицию. Значения элементов массива B формируются случайным образом из множества значений индексов элементов массива A с помощью функций $f_1(x)$ и $f_2(x)$ (x — лежит в диапазоне от 1 до $2m$, значения функций $f_1(x)$ и $f_2(x)$ находятся в пределах от 1 до $2n$).

Общий вид генерируемых фрагментов программы:

```
for i:=1 to m do
begin
   t:= A[B[f1(i)]];
   A[B[f1(i)]]:= A[B[f2(i)]];
   A[B[f2(i)]]:= t;
end;
```

Тело цикла на шаге i формирует транспозицию, определяющую подстановку π_i. Функция π имеет вид: $\pi = \pi_n\pi_{n-1} \ldots \pi_1 (A)$.

Одним из примеров функций $f_1(x)$ и $f_2(x)$ могут быть такие: $f_1(x)$ — вычисление индекса очередного четного значения элемента массива A, $f_2(x)$ — вычисление индекса очередного нечетного значения элемента массива B.

Формальная модель модуля «Работа с записями»

Постановка задачи предполагает разработку программного модуля для генерации фрагментов программ на языке Паскаль, использующих тип данных «запись». Студент должен получить понимание структурированного типа данных «запись», правил формирования составного имени для доступа к полям записи, в том числе для вложенных записей.

Интерфейс модуля должен позволять пользователю (преподавателю) задавать необходимое количество различных вариантов тестовых заданий, а также получать правильные ответы для автоматической проверки результатов тестирования (результаты выполнения вычислений над полями записи, вычисление объема памяти, необходимого для хранения переменной типа «запись» или «массив записей»).

Для описания **математической модели** модуля «Работа с записями» использован аппарат теории графов. В модуле могут генерироваться тексты программ, содержащих вложенные записи, для описания которых использованы ориентированные графы, а именно корневые деревья. Аппарат теории графов позволяет вычислять длины путей и строить составные имена полей записей. Графы представляются матрицами смежностей, которые легко реализуются в программном коде.

Алгоритм нахождения наибольшего пути в графе полезен для автоматической отбраковки матриц смежности деревьев, которые генерируются случайным образом. В тестовых заданиях целесообразно ограничить глубину вложенности записей. Практика показывает, что это число должно быть в пределах от 3 до 7.

Модуль «Работа с процедурами»

Для описания математической модели модуля «Работа с процедурами» применен подход, описанный в работе профессора А. С. Устенко «Основы математического моделирования и алгоритмизации процессов функционирования сложных систем», выложенной на сайте http://ustenko.fromru.com, в котором используется понятие типового алгоритмического процесса как совокупности действий и причинно-следственных связей между ними.

Заключение

Использование программы «Система автоматизированной генерации тестовых заданий» позволяет преподавателю за короткое время подготовить необходимое количество индивидуальных вариантов тестов, равноценных по сложности. Практика показала, что подготовка индивидуальных тестов для учебной группы из 30 человек и загрузка их в систему дистанционного обучения занимает у преподавателя не более 15 минут.

Применение математического аппарата для формального описания функциональных возможностей модулей системы дало возможность корректно и однозначно сформулировать задания на их программную реализацию, а в ряде случаев дать детальное описание алгоритмов их функционирования.

Работы по усовершенствованию системы и дополнение новых возможностей продолжаются в настоящее время. Разрабатывается общий список необходимых атрибутов генерируемых заданий для того, чтобы учесть особенности максимального количества типов заданий и предметных областей.

Система является расширяемой, может легко быть дополнена новыми модулями генерации тестовых заданий. Данный подход может быть расширен на другие предметные области при возможности их формализации.

Список литературы

1. *Алсынбаева Л. Г.* Система автоматизированной генерации тестовых заданий // Программные продукты и системы. 2009. №4.

2. *Морозенко В. В.* Дискретная математика. Пермь, 2008.

3. Фонд знаний «Ломоносов». URL: http://www.lomonosov-fund.ru/enc/ru/encyclopedia:0135750.

4. *Новиков Ф. А.* Дискретная математика для программистов. СПб., 2009.

5. *Курош А. Г.* Курс высшей алгебры. М., 1965.

Об авторах

Людмила Георгиевна Алсынбаева — канд. физ.-мат. наук, доц., Балтийский федеральный университет им. И. Канта.

E-mail: lalsynbaeva @kantiana.ru

Камил Салихович Алсынбаев — канд. техн. наук, доц., Балтийский федеральный университет им. И. Канта.

E-mail: kalsynbaev@kantiana.ru

Authors

Dr Ludmila Alsynbaeva — assistant professor, I. Kant Baltic Federal University.

E-mail: lalsynbaeva@kantiana.ru

Dr Kamil Alsynbaev — assistant professor, I. Kant Baltic Federal University.

E-mail: kalsynbaev@kantiana.ru