УДК 519.682

А. М. Магомедов, С. А. Лавренченко**Автоматизация создания тестовых единиц по языку программирования**

Предложен алгоритм компьютерного формирования тестовых заданий по основам программирования. По каждой теме учебной дисциплины рассмотрены пять формализованных структур тестовых заданий. Подробно изложен способ воплощения формализованных структур тестовых заданий в программное обеспечение. Результаты статьи находят применение для автоматической генерации практически неограниченного количества тестовых пунктов для компьютерного тестирования по основам языков программирования.

Библиография: 3 названия.

The algorithm of the computer forming the test tasks on the basics of the programming language. For each topic of the discipline examined five formal structures of test tasks. Described in detail the method of implementing formalized structures of test items to the software. The results of this paper find application to automatically generate a virtually unlimited number of test items for computer-based testing for the fundamentals of programming languages.

Bibliography: 3 items.

Ключевые слова: тестирование, компьютер, программное обеспечение, язык, программирование.

Keywords: testing, computer, software, language, programming.

Введение

Вопросы организации компьютерного тестирования рассмотрены в ряде работ, см., например, [1]. Наша статья посвящена описанию алгоритма автоматизации создания тестовых единиц по языку программирования Delphi 7.0.

Сформулирован алгоритм построения тестовой единицы в одной из нескольких общепринятых форм и приведены основные фрагменты воплощения алгоритма средствами Delphi 7.0. Подчеркнем, что нам не удалось отыскать в журнальной (и иной) литературе работы, посвященные вопросам автоматизации создания тестовых единиц.

1. Структура программного кода формирования тестовой единицы

Количество тем по основам языка программирования сравнительно невелико (20-30), количество форм тестовых единиц обычно считается равным пяти. По каждой i -й теме формулируются тестовые единицы в каждой из j форм (« i, j -термы»), $j = 1 \dots, 5$:

1) выбор одного ответа из нескольких предложенных, 2) выбор двух из нескольких предложенных, 3) упорядочение предложенных ответов по некоторому признаку, 4) установление соответствия между двумя множествами, 5) вычисление ответа.

В первой части программного кода для генерации i, j -терма объявляются и инициализируются основные объекты (главным образом — тех типов, которые изучаются в i -й теме), а также вспомогательные объекты, среди которых, в частности, могут присутствовать текстовые объекты, предназначенные для запоминания результатов (назовем их *r-объектами*). Инициализация производится в виде присвоения одиночных констант (например, 8, 'A', FALSE, []) или константных выражений (например, *not 7*, *trunc(-6.5)*, *'A'+'B'*, *random(2)*).

Например, для рассматриваемого ниже случая генерации тестовых единиц по теме «Множества», форма 3 (в классификации авторской программы — 15,3-терм), объявления могут иметь вид:

```
var
s: array[1..5] of set of byte;
rz: array[1..5] of string;
m: array [1..5] of byte;
a, k: byte;
OutString1, OutString2: string;
//две части формулировки тестовой единицы
```

Пример инициализации объектов:

```
for k:=1 to 5 do
begin
s[k]:=[];
m[k]:= 1+ random (5); //мощности множеств
while SetCount(s[k])< m[k] do
include (s[k],random (20));
//SetCount(q) - функция вычисления мощности множества q
end;
a:=2+random (5);
```

Затем над объектами выполняются действия, проверка понимания которых, — т.е. интерпретация действий и успешная бескомпьютерная имитация их выполнения, — и составляет цель опроса по данному i, j -терму. Результаты таких действий запоминаются в *r-объектах* в строковом формате.

Пример блока действий:

```
s[1]:=s[1]*s[5]; s[1]:=s[1]+s[2]; s[3]:=s[4]-s[1];
if a in s[5] then include (s[4],a) else exclude (s[4],a);
for k:=1 to 5 do rz[k]:=SetString(s[k]);
```

здесь `SetString(q)` — пользовательская функция для строкового представления множества `q`.

Во второй части программного кода i, j -терма формируются две текстовые величины: открытая часть `OutString1`, непосредственно предъявляемая учащемуся, и скрытая часть `OutString2`, предъявляемая учащемуся лишь после специального преобразования программным обеспечением, осуществляющим обработку созданных тестовых единиц и организующим интерфейс с учащимся.

Пример формирования `OutString1`:

```
OutString1:=
'№Вопрос3'+#13#10+
'Пусть a='+intToStr(a)+ ' и над множествами ';
for k:=1 to 5 do OutString1:=OutString1+'s[k]='+rz[k]+' , ';
OutString1:=OutString1+ #13#10+
'выполнены следующие действия:'+#13#10+
's[1]:= s[1]*s[5]; s[1]:= s[1]+s[2]; s[3]:= s[4]-s[1];'+#13#10+
'if a in s[5] then include (s[4], a) else exclude (s[4],
↪ a).'+#13#10+
'Выписать количество элементов в каждом множестве, начиная с
↪ s[1].';
```

Пример формирования `OutString2`:

```
OutString2:='';
for k:=1 to 5 do
begin
OutString2:=OutString2+'№да'+#13#10+IntToStr(SetCount(s[k]));
if k<5 then OutString2:=OutString2+#13#10;
end.
```

В конце второй части осуществляется вывод обоих текстов в файл, соответствующий *модулю*, — относительно самодостаточной совокупности из нескольких тем, включая и рассматриваемую i -ю тему. Пример текста тестовой единицы, выведенной программой в файл:

```
№Вопрос3
Пусть a=5 и над множествами s[1]=[7,8,17], s[2]=[7,8,17], s[3]=[6],
s[4]=[6], s[5]=[0,15],
выполнены следующие действия:
s[1]:=s[1]*s[5]; s[1]:=s[1]+s[2]; s[3]:=s[4]-s[1];
if a in s[5] then include (s[4],a) else exclude (s[4],a).
Выписать количество элементов в каждом множестве, начиная с s[1].
№да
3
№да
3
№да
1
№да
```

1

№да

2

В соответствии с требованиями программного обеспечения, используемого в конкретном вузе (ФГБОУ ВО «Дагестанский государственный университет») для обработки тестовых единиц и реализации интерфейса с учащимися, каждая тестовая единица начинается с ключевого слова «№Вопрос» с последующей цифрой от 1 до 5 — номером формы тестовой единицы; каждый элемент ответа предваряется служебным словом «№да» или «№нет», при этом только в формах 1–2 данные служебные слова соответствуют элементам ответа по смыслу: «верный» или «ошибочный», в формах же 3–5 служебное слово «№да» применяется в качестве префикса во всех элементах ответа, обозначая их начало.

2. Заключение

Приведенный описательный алгоритм компьютерной генерации тестовой единицы 15,3 (тема «Множества», форма 3) позволяет, как мы надеемся, получить вполне определенное представление о структуре и способе генерации тестовых единиц и по другим темам, независимо от выбора одной из пяти перечисленных выше форм.

Алгоритм воплощен в компьютерные программы [2]–[3], которые в течение нескольких лет используются для автоматизации разработки практически неограниченного количества разнообразных тестовых единиц по основным темам языка Delphi 7.0 и для достижения надежности тестового материала, а также для проведения независимого промежуточного контроля усвоения материала.

Работа выполнена при поддержке Отдела математики и информатики ДНЦ РАН.

Список литературы

- [1] Лаптев В. Н. Методы разработки тестовых заданий в автоматизированной контролирующей системе «Контроль» / В. Н. Лаптев, Е. В. Михайленко // Политетический сетевой электронный научный журнал Кубанского государственного аграрного университета (Научный журнал КубГАУ) [Электронный ресурс]. – Краснодар: КубГАУ. 2015. – № 09(113). С. 826–840. – IDA [article ID]: 1131509061. – Режим доступа: <http://ej.kubagro.ru/>.
- [2] Магомедов Т. А. Свидетельство № 2012616494 от 18.06.2012 о государственной регистрации программы для ЭВМ «Генерация тестовых пунктов по дисциплине “Основы программирования”».
- [3] Магомедов А. М. Свидетельство № 2018612561 от 20.02.2018 о государственной регистрации программы для ЭВМ «Автоматизация генерации тестовых пунктов по теме “Множества в языке Delphi”».

А. М. Магомедов (A. M. Magomedov)

ОМИ ДНЦ РАН

E-mail: magomedtagirl@yandex.ru

Поступила в редакцию

01.07.2018

С. А. Лавренченко (S. Lawrencenko)

Институт туризма и гостеприимства (г. Москва), ФГБОУ

ВО «Российский государственный университет туризма и
сервиса»

E-mail: lawrencenko@hotmail.com