Новиков Михаил Дмитриевич

СИСТЕМА АВТОМАТИЧЕСКОГО ТЕСТИРОВАНИЯ ПРОГРАММ, НАПИСАННЫХ НА ЯЗЫКЕ ПАСКАЛЬ

В статье описывается система, позволяющая автоматически проверять правильность работы программ на языке программирования Паскаль путем их тестирования, т.е. запуска на выполнение для некоторых фиксированных наборов исходных данных и сверки полученных результатов с эталонными. Система создана на Факультете вычислительной математики и кибернетики Московского государственного университета имени М. В. Ломоносова. Она предназначена для студентов факультета, изучающих язык Паскаль на 1-м курсе. Система позволяет контролировать правильность работы студенческих программ без необходимости их тестирования вручную, т.е. ввода исходных данных с клавиатуры и визуального анализа результатов.

Адрес статьи: www.gramota.net/materials/1/2017/6/18.html

Источник

Альманах современной науки и образования

Тамбов: Грамота, 2017. № 6 (119). С. 68-71. ISSN 1993-5552.

Адрес журнала: www.gramota.net/editions/1.html

Содержание данного номера журнала: www.gramota.net/materials/1/2017/6/

активных компонентов капсулы на фармакокинетическом и микробиологическом уровнях, так как действующие активные компоненты разделены оболочками, имеющими разную временную устойчивость к растворению в ЖКТ. Применяя капсулу с двумя или большим числом оболочек с разной временной устойчивостью к растворению, возможно получить смещенный по времени лечебный эффект.

Список источников

1. **Данилова Е. В.** Особенности клинического течения и эффективность лечения больных туберкулезом органов дыхания с первичной лекарственной устойчивостью возбудителя: автореф. дисс. … к. мед. н. М., 2005. 24 с.
2. **Кириллов Ю. В., Елькин А. В., Кобак М. Э., Басек Т. С.** Адъювантная лимфотропная химиотерапия в хирургическом лечении прогрессирующего туберкулеза легких // Проблемы туберкулеза. 2009. № 10. С. 41-45.
3. **Красносельских И. В., Гарасько Е. В., Щепочкина Ю. А.** Капсула для лекарственных и/или витаминных препаратов // Изобретения. Полезные модели: бюллетень. М., 2007. № 12. Ч. II.
4. **Красносельских И. В., Гарасько Е. В., Щепочкина Ю. А.** Капсула для лекарственных и/или витаминных препаратов // Изобретения. Полезные модели: бюллетень. М., 2008. № 19. Ч. I.
5. **Лебедева М. О., Сухова Е. В.** Формирование мотивации к лечению у больных туберкулезом легких // Проблемы туберкулеза. 2006. № 12. С. 13-15.
6. **Машковский М. Д.** Лекарственные средства. М.: Новая волна, 2000. Т. 2. 608 с.
7. **Муравьев И. А.** Технология лекарств. Изд. 2-е. М.: Медицина, 1971. 752 с.
8. **О совершенствовании противотуберкулезных мероприятий в Российской Федерации** [Электронный ресурс]: Приказ МЗ РФ № 109 от 21.03.2003 г. Доступ из СПС «КонсультантПлюс».
9. **Практическое руководство по антиинфекционной химиотерапии** / под ред. Л. С. Страчунского, Ю. Б. Белоусова, С. Н. Козлова. М., 2007. 462 с.

INNOVATIVE MEDICINAL FORM FOR TREATMENT OF TUBERCULOSIS

Krasnosel'skikh Irina Vadimovna
Garas'ko Ekaterina Vladimirovna, Doctor in Medicine, Professor
Shchepochkina Yuliya Alekseevna, Doctor in Technical Sciences, Professor
Ivanovo State Medical Academy
e-adm@isma.ivanovo.ru

An innovative development is described – a new medicinal form: a capsule for medicinal and/or vitamin preparations for treatment of tuberculosis patients. Uniqueness of this medicinal form is shown, which consists in the fact that antibacterial drugs acting on different targets in a microbial cell do not mix with each other. There is no interaction of active components of the capsule at the pharmacokinetic and microbiological levels, since active antibacterial preparations are separated by coats that have different temporal resistance to dissolution in the gastrointestinal tract. Expediency of using this medicinal form is scientifically substantiated, which will allow the medical staff to monitor patients taking medications, to reduce the toxic effect of antibiotics on the body and also to contribute to completion of the course of antituberculous therapy.

Key words and phrases: medicinal form; capsule; coats; targets in microbial cell; toxicity of antibiotics; patients with pulmonary tuberculosis.

УДК 004.42; 004.438
Физико-математические науки

В статье описывается система, позволяющая автоматически проверять правильность работы программ на языке программирования Паскаль путем их тестирования, т.е. запуска на выполнение для некоторых фиксированных наборов исходных данных и сверки полученных результатов с эталонными. Система создана на Факультете вычислительной математики и кибернетики Московского государственного университета имени М. В. Ломоносова. Она предназначена для студентов факультета, изучающих язык Паскаль на 1-м курсе. Система позволяет контролировать правильность работы студенческих программ без необходимости их тестирования вручную, т.е. ввода исходных данных с клавиатуры и визуального анализа результатов.

Ключевые слова и фразы: программирование; язык Паскаль; язык *Delphi*; тестирование; тестирующие программы; обучение программированию.

Новиков Михаил Дмитриевич, к. ф.-м. н.
Московский государственный университет имени М. В. Ломоносова
novikov_57@mail.ru

СИСТЕМА АВТОМАТИЧЕСКОГО ТЕСТИРОВАНИЯ ПРОГРАММ, НАПИСАННЫХ НА ЯЗЫКЕ ПАСКАЛЬ

Введение

Каждую программу, написанную на некотором языке программирования, необходимо проверить на правильность перед ее использованием, т.е. удостовериться, что она выдает верный результат для любых

допустимых наборов исходных данных. Наиболее распространенным способом проверки правильности работы программы является ее тестирование, т.е. задание ей некоторых наборов исходных данных и сверка выдаваемых программой результатов с ответами, заранее вычисленными для этих наборов. Проверить правильность работы программы, задавая ей все множество допустимых наборов исходных данных, обычно бывает невозможно, так как это множество оказывается слишком велико. Поэтому необходимо выделить такое сравнительно небольшое множество исходных данных, чтобы по правильности работы программы на этих данных можно было сделать вывод о правильности ее работы на любых других допустимых исходных данных. Это бывает непросто, и часто программа, правильно работающая на некотором множестве данных, работает неправильно для каких-либо других данных. Поэтому необходим тщательный подбор данных для тестирования с учетом особенностей конкретной задачи. Проблемой при тестировании программ вручную является также сложность вычисления результатов для того или иного множества исходных данных.

Классификация типичных ошибок в программах

1. При составлении программы для какой-либо задачи часто игнорируются (остаются не рассмотренными) некоторые специальные, т.е. частные наборы исходных данных, которые являются допустимыми для данной задачи, и на которых программа должна выдавать верный результат. Это может быть, например, пустая строка в задаче на обработку строк, пустое множество чисел в задаче на обработку числовой последовательности, число 0 или отрицательное число в задаче на обработку или преобразование числа и т.п.

2. Программы часто выдают неверный результат на исходных данных, которые являются «критическими» для данной задачи. Это такие исходные данные, минимальное изменение которых приводит к изменению результата. В качестве примера можно привести задачу вычисления наименьшего числа Фибоначчи, большего (большего или равного) заданного значения. Ошибка в неравенстве, т.е. использование знака «>» вместо «≥», или наоборот, приводит к неверному результату для любого числа, совпадающего с каким-либо числом Фибоначчи.

3. При обработке последовательности чисел зачастую неверно обрабатывается первый или последний член этой последовательности. Нередко происходит выход за границы допустимых индексов.

4. Часто не учитывается весь возможный диапазон исходных данных. Такие ошибки возникают, например, в задаче поиска максимума или минимума, определения позиции первого или последнего элемента с заданным значением и т.п.

5. При ручном тестировании иногда бывает трудно подсчитать (на калькуляторе) результат для какого-либо набора исходных данных, требующего громоздких вычислений. Поэтому при тестировании студенты и преподаватели часто ограничиваются заданием лишь сравнительно простых множеств исходных данных, не требующих сложных вычислений. Этого бывает недостаточно, так как программа может содержать ошибки, обнаруживаемые только на нетривиальных исходных данных.

6. Частой ошибкой студентов является использование в программах типа данных *integer* вместо *real*. Такие ошибки выявляются только при задании программе исходных данных с ненулевой дробной частью, что обычно требует более сложных вычислений, чем для целых исходных данных. Поэтому нецелые исходные данные зачастую не используются при ручном тестировании, и ошибки остаются незамеченными.

7. Нередко не продумывается, какой тип цикла должен быть использован: пошаговый (for), с проверкой условия до цикла (while) или с проверкой условия после цикла (repeat). Для большинства задач это не имеет значения, но в ряде задач выбор типа цикла принципиален.

8. Типичной ошибкой начинающих является отсутствие *else*-части в нескольких подряд идущих условных операторах. Такая ошибка иногда пропускается при ручной проверке и не проявляется на простых тестах.

9. Конечно же, главными являются алгоритмические ошибки, не позволяющие решить задачу в общем случае или приводящие к крайне неэффективным алгоритмам. Для их определения необходимо использовать большие наборы данных.

Примеры ошибочных программ и их коррекция

1. Рассмотрим задачу нахождения количества значащих цифр целого неотрицательного числа. Можно предложить следующую программу на Паскале для ее решения:

```
var n, k:integer;
begin readln(n); k:=0;
while n>0 do begin k:=k+1; n:=n div 10 end;
writeln(k);
end.
```

Эта программа выдает верный ответ для всех исходных данных, кроме нуля. Данную ошибку можно исправить, использовав цикл *repeat* вместо цикла *while* или заменив оператор присваивания `k:=0` оператором `k:=ord(n=0)`.

2. Дана последовательность чисел. Требуется найти в ней самую длинную последовательность стоящих подряд нулей (вывести количество элементов в ней). Программа

```
const n=30;
var i, r, k, m:integer;
begin
m:=0; {искомое количество}
k:=0; {текущее количество}
for i:=1 to n do begin
readln (r);
```

```
if r=0 then k:=k+1 else
begin if k>m then m:=k; k:=0;end;
end;
writeln (m)
end.
```

дает верный результат для всех исходных данных, кроме тех, в которых наиболее длинная последовательность нулей находится в конце. Для исправления этой ошибки достаточно перед оператором `writeln(m)` вставить оператор `if k>m then m:=k.`

Из вышесказанного можно сделать вывод, что тестирование программ является сложной задачей; набор тестов для каждой программы должен подбираться индивидуально, с учетом ее специфики и анализа возможных ошибок при ее составлении. Система тестирования программ, описываемая в данной статье, служит для автоматизации этого процесса.

Описание системы автоматического тестирования программ

После запуска системы на выполнение пользователю предлагается выбрать задачу для тестирования из списка. В качестве задач взяты упражнения из книги [1]. Этот задачник используется студентами Факультета вычислительной математики и кибернетики Московского государственного университета при изучении языка Паскаль. После выбора задачи предлагается выбрать файл с программой студента. Затем производятся анализ и тестирование выбранной программы.

Вначале производится предварительный анализ программы, что позволяет выявить ряд ошибок в алгоритме до начала тестирования. В частности, из программы удаляются все комментарии. Далее для некоторых задач из [Там же] предъявляются определенные требования к алгоритмам их решения. Например, в задачах на составление циклов (§ 5) и на обработку символов (§ 6) запрещается использование массивов. Для некоторых задач количество исходных данных (членов последовательности или элементов массива) должно быть задано в виде целочисленной константы. В некоторых задачах запрещено использование стандартных функций, вложенных циклов и т.п. Подобные условия проверяются системой на этапе предварительного анализа, и, если указанное требование к алгоритму нарушено, выдается соответствующее сообщение, и дальнейший анализ и тестирование программы не производятся. Кроме того, на этом этапе отсеиваются заведомо непригодные алгоритмы, которые, тем не менее, дают правильный результат для всех тестов. В качестве примера можно привести следующую программу для решения задачи нахождения количества значащих цифр неотрицательного целого числа:

```
var n, k:integer;
begin read(n);
if n<10 then writeln (1)
else if n<100 then writeln (2)
else if n<1000 then writeln (3)
else if n<10000 then writeln (4)
else writeln (5)
end.
```

Эта программа дает верный результат для всех чисел типа *integer* (его неотрицательный диапазон – [0…32767]), но тем не менее, не может рассматриваться как правильная. Для каждой задачи на составление циклов системой проверяется наличие в программе хотя бы одного цикла *while*, *repeat* или *for*, и, в случае их отсутствия, фиксируется ошибка.

Если на этапе предварительного анализа ошибок не выявлено, тестируемая программа компилируется для последующего исполнения. Используется FPC-компилятор системы *Free-Pascal* [2]. В случае отсутствия синтаксических ошибок в программе производится ее тестирование. Программа последовательно запускается на выполнение для различных встроенных в систему наборов исходных данных, консольный ввод-вывод при этом перенаправляется на ввод-вывод из специальных файлов, результаты работы программы сравниваются с эталонами. По окончании тестирования выдается сообщение о результате каждого теста и количестве ошибок при тестировании.

К настоящему времени реализовано тестирование для случаев, когда результирующие данные могут быть следующих типов: *boolean*, *integer*, *real*, *char* и *string*. Если результат имеет тип *boolean*, то в качестве ответа программа может вывести любое из слов, содержащих буквы ‘T’, ‘Y’, ‘Д’, ‘t’, ‘y’ или ‘д’, для значения ‘true’ и любое из слов, содержащих буквы ‘F’, ‘N’, ‘Н’, ‘f’, ‘n’ или ‘н’, для значения ‘false’. Если тип результата *real*, то проверяется не точное совпадение результата с эталонным, а близость его на заданную малую константу ε. Что касается результатов типа *integer*, *char* или *string*, то проверяется полное совпадение данных, выдаваемых программой, с эталонами.

Система разрабатывается на Факультете вычислительной математики и кибернетики МГУ с января 2017 года. К настоящему времени созданы и встроены в систему тесты к 100 задачам из [1] по темам «Оператор цикла», «Символьный тип», «Перечислимые и ограниченные типы. Оператор варианта» и «Регулярные типы: векторы». Система написана в среде программирования *Delphi* на языке *Object-Pascal* и может выполняться в операционной среде *Windows*.

Программный пакет занимает объем около 10 Мб на диске без FPC-компилятора. Общий объем системы *Free-Pascal*, включающий компилятор, – около 140 Мб.

В заключение следует отметить аналогичную систему *Ejudge* [3], которая используется на факультете ВМК МГУ уже в течение нескольких лет и также предназначена для автоматического тестирования программ. Система *Ejudge* требует доступа в Интернет, в отличие от описанной в статье системы, которая может быть запущена на локальном компьютере. Это более удобно, так как студенты часто выполняют задания на своих персональных ноутбуках, не всегда имеющих выход в Интернет. Система *Ejudge* является более универсальной, поддерживает разные языки программирования и используется, в основном, при проведении олимпиад по программированию и при обучении студентов второго курса. Она рассчитана на хорошо подготовленных школьников и студентов. Описанная же выше система автоматического тестирования настроена на обучение студентов первого курса.

Иллюстрация работы системы

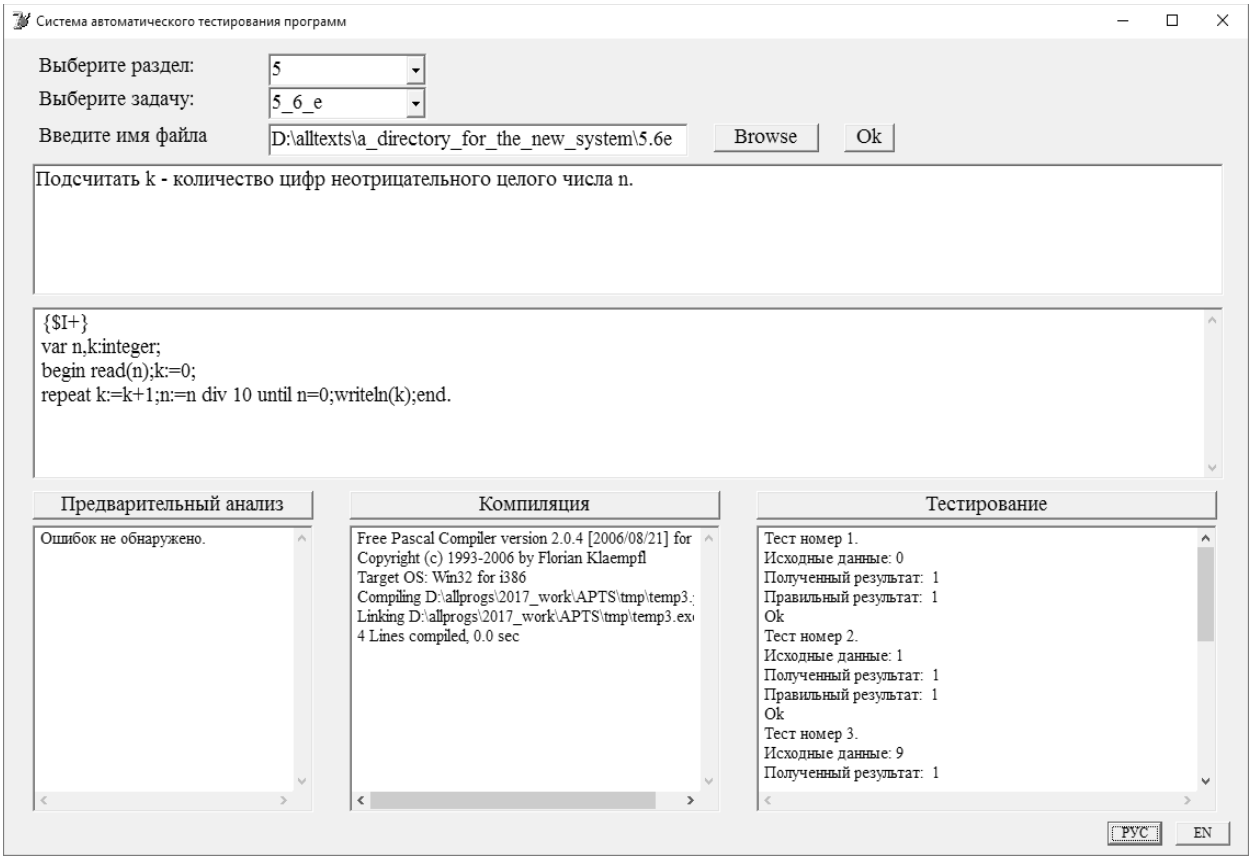

Рис. 1. *Тестирование программы нахождения количества значащих цифр неотрицательного целого числа (копия графического экрана)*

Перспективы развития системы

В данный момент продолжается расширение набора тестов за счет включения в систему задач из разных разделов книги [1]. Планируется встроить в систему анализатор ошибок, обнаруженных при тестировании, и выдачу соответствующих рекомендаций по их исправлению.

Список литературы

1. **Пильщиков В. Н.** Язык Паскаль. Упражнения и задачи. М.: Научный мир, 2003. 224 с.
2. **http://arch32.cs.msu.su/semestr1/** (дата обращения: 25.05.2017).
3. **http://heap.altlinux.org/pereslavl2007/chernov/abstract.html** (дата обращения: 25.05.2017).

AUTOMATIC TESTING SYSTEM OF PASCAL-LANGUAGE PROGRAMS

Novikov Mikhail Dmitrievich, Ph. D. in Physical-Mathematical Sciences
Lomonosov Moscow State University
novikov_57@mail.ru

The article describes a system that allows automatically checking the correctness of programs in the Pascal programming language by testing them, i.e. starting on execution for some fixed sets of input data and checking the obtained results and the reference ones. The system was created at the Faculty of Computational Mathematics and Cybernetics of Lomonosov Moscow State University. It is intended for students of the Faculty, who study the Pascal language during their first year. The system allows monitoring correctness of students' programs operating without having to test them manually, namely the input of initial data from the keyboard and the visual analysis of the results.

Key words and phrases: programming; Pascal language; Delphi language; testing; testing programs; teaching programming.