ственное. Таким образом, влияние этой пары неоднозначно и разнонаправленно, что может отрицательно повлиять на аудиторию зрителей.

Доминантные слова рекламы Maybelline – «колоссальный» и «объем» – производят впечатление чего-то угловатого, однако сильного, хорошего, величественного, яркого. Таким образом, можно сделать вывод, что в отношении эмоционального воздействия акценты расставлены верно.

Говоря об окраске рекламных слоганов компаний, следует отметить, что система VAAL считает слоган компании Max Factor не обладающим выраженными фоносемантическими характеристиками, а Maybelline производит впечатление низменного, слабого, тихого и тусклого [3].

Таким образом, существует множество факторов, определяющих эффективность воздействия рекламных роликов на потребителей. Наличие одних из них может компенсировать отсутствие других, однако к процессу создания рекламы стоит относиться максимального внимательно и творчески, особенно в условиях современного рынка и конкуренции.

Литература:

1. Контент-анализ рекламы. http://psyfactor.org/lib/content-analysis4.htm/.
2. *Логвинов А. М.* Социология рекламной деятельности: учебное пособие. Красноярск. Поликом, 2005. С. 440.
3. Интернет-адрес: http://vaal.ru/.

ТЕОРИЯ КОНЕЧНЫХ АВТОМАТОВ В БАЗОВЫХ АЛГОРИТМИЧЕСКИХ СТРУКТУРАХ НА DELPHI

А. Г. Третьяк

Цель: исследование вопроса управления изначально непредсказуемого поведения кода программы.

Долгое время в русскоязычной литературе по программированию методология автоматного программирования задач разнообразного назначения замалчивалась. Тем временем, на Западе эта методология активно развивалась. В начале девяностых появились средства быстрой разработки приложений (RAD – Rapid Application Development), к которым относится, в частности, и Delphi. Эта система программирования позволяет не только программировать в привычном смысле слова, но и фактически рисовать программы, перетаскивая в оконные формы визуальные компоненты из VCL (Visual Component Library). VCL имеет тенденцию скрывать точную реализацию определенных объектов, тем самым не давая посторонним менять умалчиваемое поведение кода, что означает непредсказуемость маршрута выполнения программы или функциональных объектов [1].

Теория конечных автоматов очень актуальная тема на сегодняшний день [2]. Теория автоматов занимается изучением процессов, протекающих в автоматах различного рода, и общих закономерностей, которым они подчинены, широко применяя для этого алгебраический аппарат, математическую логику, комбинаторный анализ и теорию вероятностей. Теория автоматов находит применение, как в математике, так и в решении практических задач.

Ежегодно интернет сообщество The Code Project проводит голосование на лучший выбор пользователей Windows в различных категориях. По итогам этого голосования в 2011 году Embarcadero Delphi XE является лучшим языком программирования.

Существуют различные варианты задания конечного автомата. Например, конечный автомат может быть задан с помощью пяти параметров [3]:

$$M = (Q, \Sigma, \delta, q_0, F), \quad (1)$$

где Q – конечное множество состояний автомата, q_0 – начальное состояние автомата ($q_0 \in Q$), F – множество заключительных (или допускающих состояний) таких, что $F \subset Q$. При достижении одного из этих состояний работа автомата прекращается, $\sum$ – допустимый входной алфавит (конечное множество допустимых входных символов), из которого формируются строки, считываемые автоматом, δ – заданное отображение множества $Q \times \sum$ во множество подмножеств $P(Q)$ $\delta : Q \times \sum \rightarrow Q$ (иногда δ называют функцией переходов автомата).

Автомат начинает работу в состоянии q_0, считывая по одному символы входной строки. Считанный символ переводит автомат в новое состояние из Q в соответствии с функцией переходов.

Функционирование автомата состоит в порождении двух последовательностей: последовательности очередных состояний автомата $s_1[1]$ $s_2[2]$ $s_3[3]$… и последовательности выходных символов $y_1[1]$ $y_2[2]$ $y_3[3]$…, которые для последовательности символов $x_1[1]$ $x_2[2]$ $x_3[3]$… разворачиваются в моменты дискретного времени $t = 1,2,3…$

Функционирование автомата в дискретные моменты времени t может быть описано системой рекуррентных соотношений:

$$s(t+1) = \delta(s(t),x(t)),$$

$$y(t) = \lambda(s(t),x(t)).$$

В структурном программировании с помощью 5 основных конструкций, как из кубиков строится любая процедура. Это такие операторы, как: sequence, if-then-else, while-do, repeat-until and case.

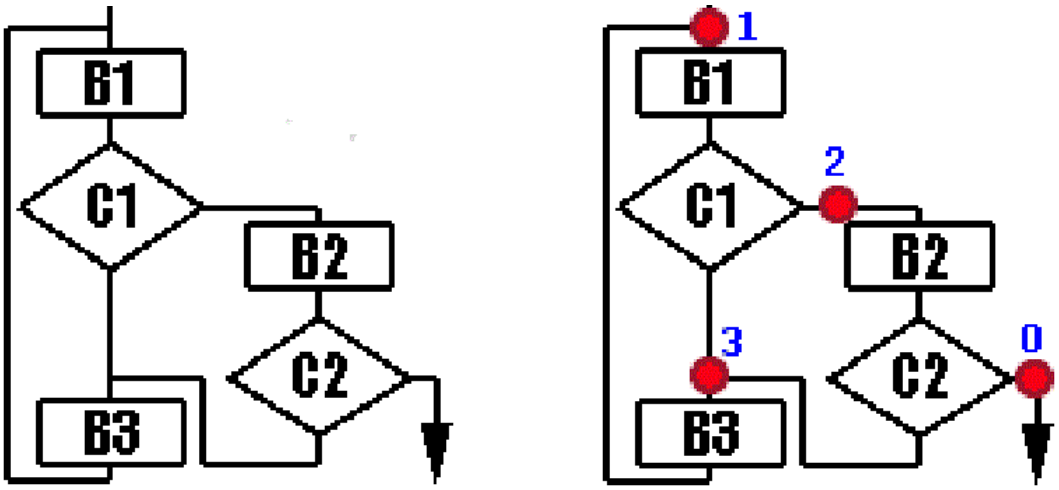

Рис. 1. Расстановка контрольных точек

Очень просто управлять поведением кода программы – приемом автоматного программирования – расстановкой «контрольных точек» с использованием оператора case – программной реализацией конечного автомата.

Рассмотрим случай расстановки «контрольных точек» на следующем примере.

Для того чтобы облегчить задачу следует разделить алгоритм на блоки [4].

Представим, что есть некий блок (рис.1), который может находиться в одном из 4 состояний. И есть набор действий, в результате которых блок переходит из одного состояния в другое.

Для отображения этого самого состояния, заведем в программе некоторую переменную, например, state. А внутри веток case будем изменять ее состояние.

Пишем программу:

```
var state:integer
begin
  state:=1;  {для любого алгоритма}
  repeat
    case state of
...
    end;
  until state=0; {тоже для любого алгоритма}
end;
```

Теперь пропишем ветки case. Не забудьте в конце каждой ветки уточнить состояние:

case state of

1: begin b1; if c1 then state:=2 else state:=3 end;

2: begin b2; if c2 then state:=0 else state:=3 end;

3: begin b3; state:=1 end;

end;

Программа готова.

Добавление или уменьшение контрольных точек только ухудшает читабельность и устойчивость работы программы. Поэтому для решения подобных задач лучше выбрать оптимальный вариант, как в нашем примере.

В качестве примера реализации конечного автомата приведем программу тестирования студентов по какому-либо определённому предмету (рис. 2).

Есть несколько уровней сложности (0–7) вопросов и соответствующая их оценка (2–9) баллов. Стартовый уровень знаний – 0 и т.д. Норматив временного интервала между уровнями – 5 минут. Перескакивание через уровни запрещено. При ответе на вопрос на конкретном уровне студент нажимает кнопку. Если ответ правильный, то разрешается переход на следующий уровень теста.

Автомат заставляет студента пройти «контрольные точки» именно так, как *расписал программист.*

Код конечного автомата изобилует ветвлениями, но *ветвлений* – **логических структур «выбор»** – в явном виде *как будто бы и нет*. Оператор **case** делает текст программы абсолютно «прозрачным».

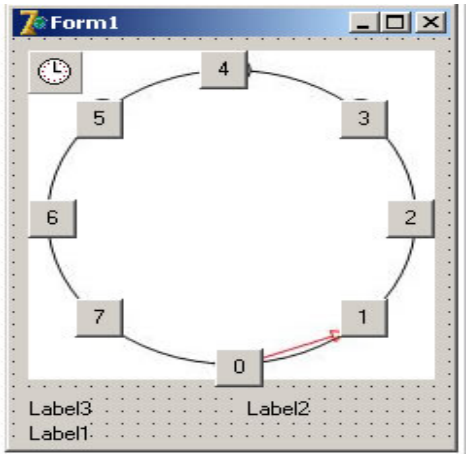

Рис. 2. Программная реализация конечного автомата на Delphi

Код конечного автомата *взаимно ортогонален*. При модификации кода, при добавлении в код новых **case**-фрагментов изменять исходный

код не потребуется. Программа не «поползет» – снижается вероятность ошибок при кодировании.

Выводы:

1. Автоматный подход – быстро развивающееся направление программирования, являющееся на сегодняшний момент одним из наиболее актуальных.

2. Автоматное программирование способствует решению циклических задач практически любой сложности с минимальными затратами времени на отладку.

3. Автоматный подход может использоваться при реализации вычислительных алгоритмов.

Литература

1. *Кузнецов Б. П.* Психология автоматного программирования // BYTE. Россия. 2000. N11.
2. Интернет-адрес: http://teorya.hut.ru/page4.htm.
3. Интернет-адрес: http://ru.wikipedia.org/wiki/Конечный_автомат.
4. Интернет-адрес: http://www.interface.ru/home.asp?artId=2779.

РАСЧЕТ И АНАЛИЗ ХАРАКТЕРИСТИК ТЕСТА В ONLINE РЕЖИМЕ

А. В. Фалей, С. И. Березюк

Слово «тест» английского происхождения и используется, как правило, для обозначения всевозможных проб, испытаний и проверок, направленных на оценку состояния объекта или явления [1].

Педагогический тест – это система параллельных заданий равномерно возрастающей трудности, позволяющая оценить структуру и качественно измерить уровень подготовленности испытуемых [2, с. 148].

Хотя тестирование уже давно стало распространенным явлением в образовательной практике, многие учителя и преподаватели, пробующие себя в составлении тестов, могут не отдавать себе отчет в том, что результат тестирования зависит не только от знаний тестируемого, но и от качества самого теста. Очень часто происходит так, что, составив определенное количество заданий в тестовой форме и объединив их, думают, что получили тест. Но задания в тестовой форме могут вовсе не быть тестовыми, а тем более не образовывать тест как таковой. Попытки же превращения заданий в тестовой форме в тестовые задания неизбежно приводят к эмпирической проверке их тестовых свойств.

Как известно [2; 3; 4], качество теста можно оценить с помощью определенных характеристик: надежность, валидность, сложность, точность, дискриминативность, социокультурная адаптированность, достоверность, однозначность, стандартизированность, нормирование и др. На настоящем этапе нашего исследования мы рассчитываем и оцениваем коэф-