Статья опубликована в журнале «Компьютерные инструменты в образовании». 2007. № 5, с. 62–67, а также в тезисах научно-технической конференции «Научное программное обеспечение в образовании и научных исследованиях». СПбГУ ПУ. 2008, с. 264–271.

Н. Н. Красильников, В. Г. Парфенов,
Ф. Н. Царев, А. А. Шалыто

ВИРТУАЛЬНАЯ ЛАБОРАТОРИЯ ДЛЯ ПЕРВОНАЧАЛЬНОГО ОБУЧЕНИЯ ПРОЕКТИРОВАНИЮ ПРОГРАММ

Санкт-Петербург, Санкт-Петербургский государственный университет информационных технологий, механики и оптики
krasilnikov@rain.ifmo.ru

ВВЕДЕНИЕ

В настоящее время школьников даже старших классов не учат **проектированию** программ, а студентов обучают лишь в некоторых ВУЗах и только на старших курсах.

Так же как до последнего времени программированию учили на Паскале как языке, наиболее удобном для первоначального обучения профессиональному программированию, так авторами настоящей работы предлагается первоначально **обучать проектированию на основе автоматного подхода**. По мнению авторов, автоматы являются естественной формой описания поведения сущностей, так как состояния и переходы из одного состояния в другое присущи и естественны для человека.

Этот подход в течение уже нескольких лет используется на кафедре компьютерных технологий СПбГУ ИТМО студентами третьего курса, которые разрабатывают проекты и проектную документацию к ним на основе автоматного подхода [1]. Документация и проекты публикуются на сайте http://is.ifmo.ru. Для поддержки автоматного программирования создано инструментальное средство *UniMod* [2]. Однако для первоначального знакомства с проектированием это средство является достаточно сложным.

Поэтому в рамках настоящей работы предложен упрощенный подход к обучению проектированию программ.

ОСНОВНЫЕ ПОЛОЖЕНИЯ

Разработана «Виртуальная лаборатория для обучения проектированию программ», в которой предлагается создать программные модули для управления различными сущностями.

При этом по словесному описанию сущности проектируется граф переходов конечного автомата, по которому текст программы строится формально и изоморфно. Для такой реализации разработан текстовый язык автоматного программирования.

Текстовый язык автоматного программирования

Предлагаемый язык позволяет в простой форме описывать графы переходов автоматов Мили – состояния и переходы между ними.

В языке предусмотрены следующие конструкции:

- `automata <name>` – объявление автомата, за которым должна следовать его реализация, заключенная в фигурные скобки. Начальное состояние – нулевое.
- `state <name>` – объявление состояния, за которым в фигурных скобках должно следовать описание его переходов.
- `on <event name> if(<logic statement>) do {<job1>, <job2>,…} go to <state name>;` – описание перехода из текущего состояния в состояние `<state name>` по событию `<event name>` при выполнении логического условия `<logic statement>`. При переходе выполняются действия `<job1>,…` В этой конструкции допускается пропуск `if(<logic statement>), do {<job1>, <job2>,…}` и `go to <state name>`.

Рассмотрим пример описания автомата на этом языке (листинг 1).

Листинг 1

```
automata Auto
{
	state state0
	{
		on event0 if(stmt0 && stmt1) do {job0};
		on event0 if(!stmt2) do {job1} go to state1;
	}
	state state1
	{
		on event0 if(stmt1) go to state0;
		on event0 do {job1};
	}
}
```

Здесь описывается автомат `Auto`, состоящий из двух состояний `state0` и `state1`. С состояниями связаны переходы, выполняющиеся в зависимости от условий, и действия, производимые во время переходов.

Аналогичный язык уже был предложен в работе [3], но он был реализован только в среде *MPS* (*Meta Programming System*) от компании *JetBrains*, что сильно затрудняет его использование.

Для языка, описанного в настоящей работе, была написана формальная грамматика и реализован простой компилятор.

Компилятор для текстового языка автоматного программирования

Компилятор преобразует текст программы на языке автоматного программирования в экземпляр класса языка *Java*. Полученный экземпляр класса сразу готов к использованию.

Описание процесса построения компилятора и описание тонкостей его работы выходит за рамки данной статьи, так как о построении компиляторов подробно написано в книге [4].

Виртуальна лаборатория

В «Виртуальной лаборатории для обучения проектированию программ» представлен набор заданий, в каждом из которых предлагается самостоятельно реализовывать (по спроектированному учащимися графу переходов) поведение соответствующей сущности на предложенном текстовом языке автоматного программирования. В лаборатории можно найти и готовые программы, которые легко модифицируются пользователем.

В состав виртуальной лаборатории входит редактор кода для редактирования текстов программ и указанный выше компилятор. Поскольку результатом работы компилятора является готовый для использования экземпляр класса, то имеется возможность наблюдать за изменениями поведения объектов без перекомпиляции остальных модулей виртуальной лаборатории.

Таким образом, виртуальная лаборатория содержит все необходимое для обучения основам проектирования на «живых» примерах непосредственно в окне интернет-браузера.

Виртуальная лаборатория расположена на сайте http://is.ifmo.ru в одноименном разделе.

ПРИМЕР ИСПОЛЬЗОВАНИЯ

Покажем, как использовать возможности, предоставляемые виртуальной лабораторией, на примере задачи об «Умном муравье» [5].

Условие задачи

Игра происходит на поверхности тора размером 32 на 32 клетки (рис. 1). В некоторых из них находятся яблоки. Всего на поле 89 клеток с яблоками. В левой верхней клетке находится муравей, который смотрит направо. В любой момент времени он занимает одну клетку и может смотреть в одном из четырех направлений.

Муравей умеет определять, находится ли непосредственно перед ним яблоко (входная переменная **`a`**). За один ход муравей может совершить одно из четырех действий: сделать шаг вперед (действие **`g`**), съедая яблоко, если оно там находится; повернуть налево (действие **`tl`**); повернуть направо (действие **`tr`**); ничего не делать.

Съеденные муравьем яблоки не пополняются, а муравей жив на протяжении всей игры. Расположение яблок фиксировано. Игра длится 200 ходов, по истечении которых подсчитывается число яблок, съеденных муравьем. Цель игры – создать муравья «с минимальным числом состояний», который съест как можно больше яблок за число шагов, не превышающее указанное.

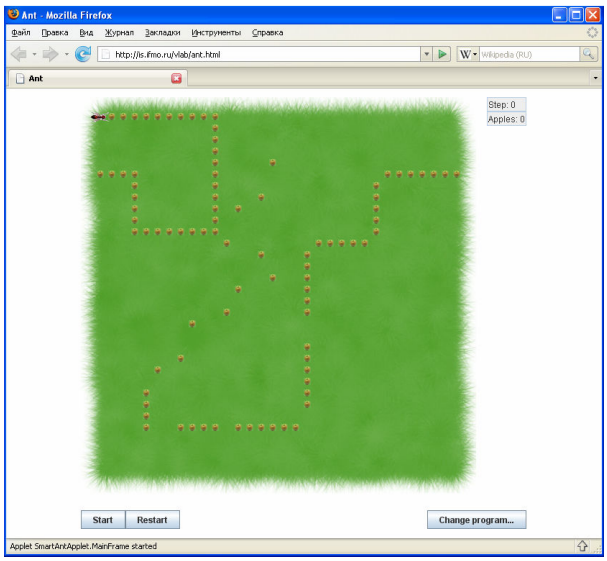

Рис. 1. Игровое поле

Исследование задачи в виртуальной лаборатории

Опишем поведение муравья графами переходов автоматов Мили с различным числом состояний и проанализируем поведение автоматов.

Автомат с одним состоянием

Простейший муравей (граф переходов которого для экономии места опущен) съедает 42 яблока (листинг 2).

Листинг 2

```
automata Auto2
{
      state s0
      {
            on e0 if(a)  do {g};
            on e0 if(!a) do {tr};
      }
}
```

Далее открываем соответствующую страницу виртуальной лаборатории с задачей об умном муравье, выбираем режим редактирования программы (`Change Programm...`) и в окно редактора кода копируем записанную программу (рис. 2).

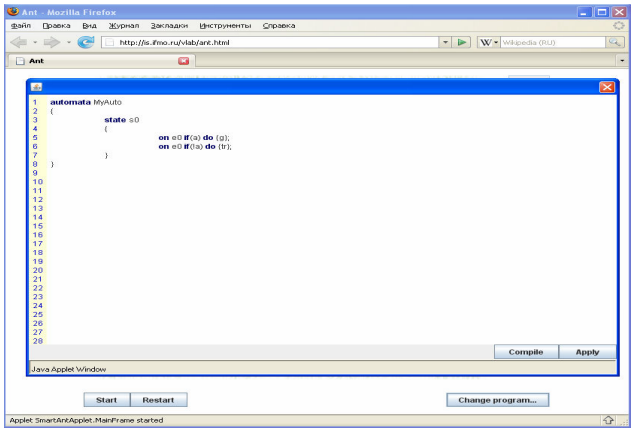

Рис. 2. Редактора кода виртуальной лаборатории

Откомпилировав и запустив программу, получаем ожидаемый результат – за 200 ходов съедено 42 яблока (рис. 3).

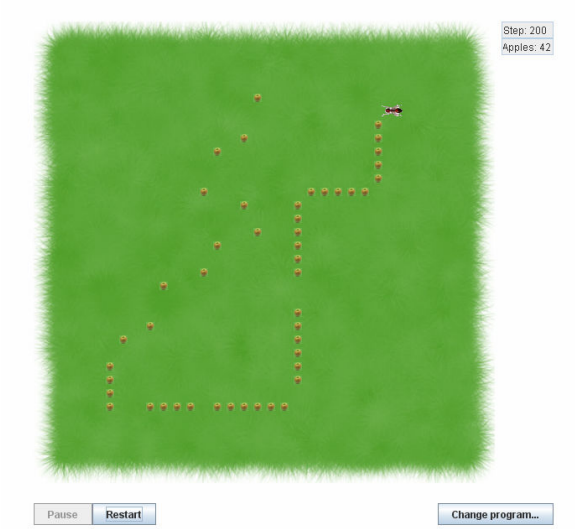

Рис. 3. Поле, на котором съедено 42 яблока

Автомат с пятью состояниями

Особенность разработанной лаборатории состоит в том, что в ней легко выполнять редактирование программы. Поэтому рассмотрим муравья, поведение которого описывается автоматом с пятью состояниями (рис. 4).

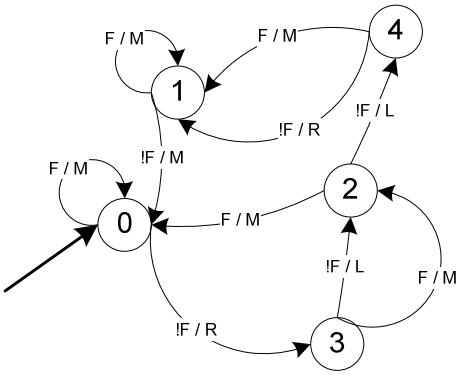

Рис. 4. Автомат с пятью состояниями

Автоматная программа, реализующая этот автомат, приведена в листинге 3.

Листинг 3

```
automata Auto3
{
      state s0
      {
            on e0 if(a)  do {g};
            on e0 if(!a) do {tr} go to s3;
      }
      state s1
      {
            on e0 if(a)  do {g};
            on e0 if(!a) do {g} go to s0;
      }
     state s2
     {
           on e0 if(a)  do {g}  go to s0;
           on e0 if(!a) do {tl} go to s4;
     }
     state s3
     {
           on e0 if(a)  do {g}  go to s2;
           on e0 if(!a) do {tl} go to s2;
     }
     state s4
     {
           on e0 if(a)  do {g}  go to s1;
           on e0 if(!a) do {tr} go to s1;
     }
}
```

Этот муравей за 200 ходов съедает только 83 яблока (рис. 5).

Рис. 5. Поле, на котором съедено 83 яблока

Автомат с семью состояниями

Рассматриваемая задача может быть решена с помощью автомата с семью состояниями (рис. 6).

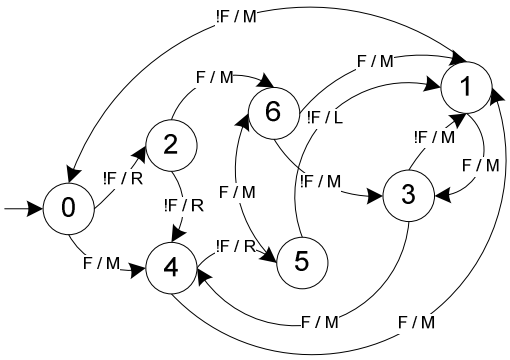

Рис. 6. Автомат, позволяющий муравью съесть всю еду

Этот автомат построен в работе [5]. В листинге 4 приведена автоматная программа, реализующая этот автомат.

Листинг 4

```
automata Auto4
{
      state s0
      {
            on e0 if(a)  do {g}   go to s4;
            on e0 if(!a) do {tr}  go to s2;
      }
      state s1
      {
            on e0 if(a)  do {g}  go to s3;
            on e0 if(!a) do {g}  go to s0;
      }
      state s2
      {
            on e0 if(a)  do {g}   go to s6;
            on e0 if(!a) do {tr}  go to s4;
      }
      state s3
      {
            on e0 if(a)  do {g}  go to s4;
            on e0 if(!a) do {g}  go to s1;
      }
      state s4
      {
            on e0 if(a)  do {g}  go to s1;
            on e0 if(!a) do {tr} go to s5;
      }
      state s5
      {
            on e0 if(a)  do {g}  go to s6;
            on e0 if(!a) do {tl} go to s1;
      }
      state s6
      {
            on e0 if(a)  do {g}  go to s1;
            on e0 if(!a) do {g}  go to s3;
      }
}
```

Этот муравей съедает все 89 яблок за 200 ходов (рис. 7).

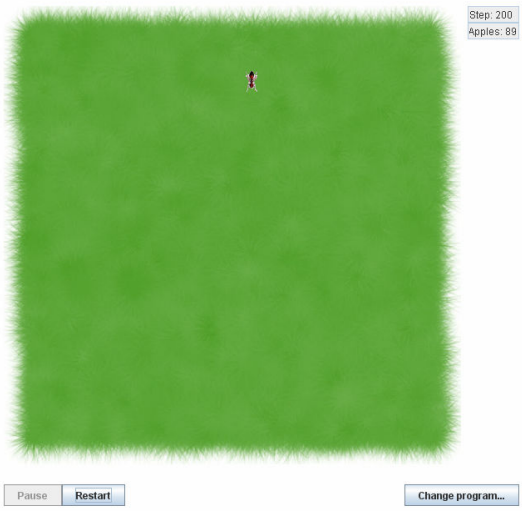

Рис. 7. Поле, на котором съедено все 89 яблок

ЗАКЛЮЧЕНИЕ

Авторы надеются, что предложенный подход к начальному обучению проектированию программ окажется полезным, совмещая в себе увлекательность и интерактивность.

Кроме рассмотренной задачи виртуальная лаборатория содержит еще несколько задач, состав которых пополняется и будет пополняться в дальнейшем.

ЛИТЕРАТУРА

1. *Шалыто А. А.* Технология автоматного программирования / Труды первой Всероссийской научной конференции «Методы и средства обработки информации» М.: МГУ. 2003. http://is.ifmo.ru/works/tech_aut_prog/

2. *Инструментальное средство Unimod*. http://unimod.sourceforge.net/intro.html

3. *Гуров В. С., Мазин М. А., Шалыто А. А.* Текстовый язык автоматного программирования / Тезисы докладов международной научной конференции, посвященной памяти профессора А. М. Богомолова «Компьютерные науки и технологии». Саратов: СГУ. 2007, с. 66–69. http://is.ifmo.ru/works/_2007_10_05_mps_textual_language.pdf

4. *Ахо А., Сети Р., Ульман Дж.* Компиляторы. Принципы, технологии, инструменты. М.: Вильямс. 2003.

5. *Царев Ф. Н., Шалыто А. А.* О построении автоматов с минимальным числом состояний для задачи об «Умном муравье» / Сборник докладов X международной конференции по мягким вычислениям и измерениям. Том 2. СПб ГУ ЭТУ «ЛЭТИ». 2007, с. 88–91. http://is.ifmo.ru/download/ant_ga_min_number_of_state.pdf