Первое сообщение о модуле невидимой автоматической проверки заданий по программированию в среде PASCALABC.NET

Михалкович С.С.

Южный федеральный университет,
Институт математики, механики и компьютерных наук
E-mail: miks@sfedu.ru

Среда программирования PascalABC.NET активно развивается в последние 7 лет [1-2]. Связано это в основном с плавным изменением методики обучения школьников в Воскресной компьютерной школе Института математики, механики и компьютерных наук [3]. Поскольку разработчики PascalABC.NET активно преподают в Воскресной компьютерной школе, новые идеи в этой области генерируются, реализуются и внедряются непосредственно во время учебного процесса. Проверка реализованных сервисов может охватывать до 300 школьников в год – именно такое количество учеников компьютерной школы осваивает программирование каждый год.

Одним из широко используемых средств является электронный задачник Programming Taskbook, разработанный Абрамяном М.Э. и встроенный в среду PascalABC.NET. Он содержит более 2000 автоматически проверяемых задач по различным разделам программирования от элементарных задач до файлов, динамических структур данных, параллельного программирования и паттернов объектно-ориентированного проектирования. Электронный задачник кардинально превосходит известные олимпиадные системы проверки по многим параметрам. Это прежде всего проверка типизированных данных вместо проверки текстового вывода в файл, детальные сообщения о причинах ошибки, значительно более лёгкая разработка заданий и отсутствие необходимости разворачивать удалённую олимпиадную систему проверки. Эффективность электронного задачника в учебном процессе была доказана более чем 20-летним опытом использования в Детской компьютерной школе.

Электронный задачник имеет и определенные недостатки. Это прежде всего жёстко заданный набор задач, изменение которого требует создания нового дистрибутива PascalABC.NET. В реальности подобное изменение происходит не чаще раза в год. С другой стороны, для учебного процесса к каждому занятию разрабатываются новые задачи, для которых хотелось бы обеспечить легкую автоматическую проверку.

В настоящей работе представлен модуль невидимой проверки простых задач по программированию, некоторые идеи которого

заимствованы из работы [4]. Новый модуль обладает следующими уникальными возможностями:

- новые задания, предлагаемые к уроку, можно обеспечивать системой автоматической невидимой проверки с помощью написания специального модуля – время разработки такого модуля сопоставимо с временем разработки самих заданий к уроку;
- ввод данных по заданию может осуществляться учеником из различных источников: с клавиатуры или генерацией случайных чисел;
- вывод данных можно осуществлять в разной последовательности, предусмотренной заданием;
- можно вводить и выводить данные различных типов: например, в рамках одного задания можно вводить и выводить либо только целые, либо только вещественные значения.

Рассмотрим пример простого задания и способ его проверки.

```
{ Задание. Напишите программу, в которой вводятся числа x,y
    и выводятся их минимум и максимум
}
uses LightPT,Tasks;
begin

end.
```

Обратим внимание на подключение двух модулей: LightPT – общие функции проверки заданий (стандартный модуль), Tasks – модуль проверки заданий из текущей папки (хранится в текущей папке). Тело программы – пустое, в него ученик записывает решение. Например, решение может выглядеть так:

```
var (x,y) := ReadInteger2;
Print(Min(x,y),Max(x,y));
```

После запуска такая программа, находящаяся под управлением модуля проверки, выведет:

```
5 2
2 5
Задание выполнено
```

Наиболее интересной здесь является возможность неявного подключения модулей LightPT и Tasks при запуске программы из определенного каталога. Таким образом, итоговая программа для ученика будет выглядеть вообще без дополнительных модулей:

```
begin
  var (x,y) := ReadInteger2;
  Print(Min(x,y),Max(x,y));
end.
```

Проверка задания таким образом для ученика будет полностью невидимой.

Следующей интересной возможностью здесь является изменение типа вводимых и выводимых данных – ученик может вводить вещественные значения вместо целых, при этом и выходные значения тоже станут вещественными:

```
var (x,y) := ReadReal2;
Print(Min(x,y),Max(x,y));
```

Задание тем не менее будет считаться выполненным, поскольку в функции автоматической проверки заложена возможность разных типов данных.

Не менее интересной возможностью является изменение источника ввода с ввода с клавиатуры на генерацию случайных чисел. Если ученик напишет такую программу

```
var (x,y) := Random2(1,10);
Print(Min(x,y),Max(x,y));
```

то задание будет по-прежнему расценено как правильно решенное.

Наконец, данные могут выводиться в другой последовательности и перемежаться с поясняющим выводом:

```
var (x,y) := ReadInteger2('Введите x,y:');
Print('Максимум =',Max(x,y),' Минимум =',Min(x,y));
```

Удивительно, но и в этом случае проверка выведет, что задание успешно выполнено.

Многие задания в своей постановке предусматривают некоторый ввод – в этом случае проверка будет проще. Например, рассмотрим задание на массивы:

```
{ Отфильтруйте и выведите чётные элементы целого массива }
begin
  var a := new integer[10];
  for var i:=0 to a.Length-1 do
    a[i] := Random(1,99);
  a.Println;
// Напишите вывод отфильтрованных элементов

end.
```

Ученику достаточно написать решение после комментария:

```
for var i:=0 to a.Length-1 do
  if a[i] mod 2 = 0 then
    Print(a[i]);
```

и задание будет считаться выполненным. Здесь после заполнения случайными данными элементы массива выводятся для контроля с помощью `a.Println`. Однако можно не делать такой вывод – проверяющая программа сочтет решение правильным в обоих случаях.

Рассмотрим, что должен написать преподаватель для проверки решения в модуле Tasks. В функции

```
function CheckTaskT(name: string): TaskStatus;
```

в ветви оператора case должна быть написана ветвь, связанная с проверкой задания в указанном файле:

```
'04_Arr1_If': begin
  ClearOutputListFromSpaces;
  var a := IntArr(10);
  var b := a.FindAll(x->x mod 2 = 0);
  Result := CompareSeqWithOutput(a + b);
  if Result = NotCompleted then
    Result := CompareSeqWithOutput(b);
end;
```

Здесь `04_Arr1_If.pas` – файл с заданием, `ClearOutputListFromSpaces` очищает вывод от необязательной информации, функция `CompareSeqWithOutput` сравнивает выведенный результат с предполагаемым правильным и рассматривается два варианта решения – с эхо-печатью вводимых данных и без неё.

Таким образом, новый вариант проверки заданий по программированию позволяет оперативно разрабатывать функции автоматической проверки вместе с разработкой самого задания, делать данную проверку невидимой для учащегося и обеспечивать несколько вариантов правильного решения с разной последовательностью и типами данных для ввода и вывода.

Литература

1. *Бондарев И.В., Михалкович С.С.* Система программирования PascalABC.NET: новые возможности 2015-16 гг. / Труды XXIII Научно-методической конференции «Современные информационные технологии: тенденции и перспективы развития». Ростов-на-Дону: Изд-во ЮФУ, 2016. С. 69–71.
2. *Бондарев И.В., Михалкович С.С.* Система программирования PascalABC.NET: 15 лет развития / XXV Научная конференция «Современные информационные технологии: тенденции и перспективы развития». Материалы конференции. Ростов-на-Дону, 2018. С. 31–34.
3. *Михалкович С.С., Абрамян М.Э.* Основы программирования на языке PascalABC.NET. Скалярные типы данных, управляющие операторы, знакомство с массивами, процедуры и функции, работа с графикой / Учебник. Ростов-на-Дону - Таганрог: 2017. 248 с.
4. *Абрамян М.Э., Михалкович С.С.* Конструктор учебных заданий для системы Pascal ABC. Труды конференции «Современные информационные технологии в образовании: Южный федеральный округ». Ростов-на-Дону: Изд-во ЮФУ, 2007. С. 20-21.