**Инженерный вестник Дона**, №3 (2023)
ivdon.ru/ru/magazine/archive/n3y2023/8299

# Тренажерно-обучающая система для контроля знаний и навыков по основам программирования на языках высокого уровня

*И.С. Полевщиков, Ю.Н. Белова, Р.М. Романов*

*Московский государственный университет технологий и управления имени К.Г. Разумовского (ПКУ)*

**Аннотация:** Статья посвящена развитию тренажерно-обучающих систем для подготовки специалистов по автоматизации и информатизации. Разработаны структура и математическая модель тренажерно-обучающей системы (ТОС) для контроля навыков при обучении данных специалистов (в частности, по основам программирования). Параметры математической модели основаны на представлении знаний об изучаемых объектах и процессах в области разработки автоматизированных информационных систем, что позволяет с применением ТОС автоматически генерировать и оценивать небольшие практические задания для студентов. Разработан прототип программного обеспечения ТОС в форме веб-приложения. В практическом аспекте применение ТОС в учебном процессе по дисциплинам в области программирования позволит уменьшить долю трудоемкой работы преподавателя по составлению и проверке выполнения заданий.
**Ключевые слова:** информационные технологии в образовании, тренажерно-обучающая система, контроль знаний и навыков, языки программирования высокого уровня.

## 1. Введение

Изучение программирования на современных языках высокого уровня (Java, Python, C#, C++ и др.) является важной составляющей учебного процесса разработчиков автоматизированных информационных систем [1-3] в ВУЗах (в частности, по направлениям подготовки «Информатика и вычислительная техника», «Прикладная информатика»).

Дисциплина, обучающая основам алгоритмизации и программирования, как правило, изучается на 1 курсе и включает формирование навыков записи арифметических и логических выражений на соответствующем языке, разветвляющихся и циклических алгоритмов, выполнения действий с одномерными и многомерными массивами.

С увеличением числа студентов возрастает сложность выполнения преподавателем задач, связанных с контролем знаний и навыков. Требуется значительное время на составление и проверку индивидуальных вариантов

практических заданий по основам алгоритмизации и программирования.

В работах различных авторов (например, [4-6]) рассматриваются возможности применения информационных технологий для повышения эффективности обучения программированию и смежным дисциплинам [7].

Использование систем автоматизированного тестирования знаний упрощает процессы контроля, но подобные средства обучения не предоставляют возможность производить оценку начальных навыков выполнения практических заданий без участия преподавателя. Повысить эффективность процессов сбора и обработки данных о выполнении студентами учебных практических заданий позволяют различные средства электронного обучения и дистанционные образовательных технологии [8], однако их применение незначительно снижает трудоемкость работы преподавателя по составлению и проверке выполнения заданий.

Актуальной является задача развития информационных систем в сфере контроля навыков с целью упростить процессы формирования и оценки большого числа индивидуальных вариантов практических заданий по основам программирования. Результаты решения этой задачи, основанные на разработке математического и программного обеспечения тренажерно-обучающей системы для подготовки специалистов в сфере информационных технологий и автоматизации, рассмотрены далее.

## 2 Математическое обеспечение тренажерно-обучающей системы для контроля навыков по основам программирования

Разработаны структура и математическая модель тренажерно-обучающей системы (ТОС) для контроля навыков при обучении специалистов по информатизации и автоматизации (в частности, по программированию). Множество подсистем ТОС и параметров математической модели представлено схематично на рис. 1. Разработка ТОС

является развитием результатов исследований [8-10].

Параметры математической модели основаны на представлении знаний об изучаемых объектах и процессах в области разработки автоматизированных информационных систем (например, при изучении основ программирования – знаниях об элементах программного кода и алгоритмических конструкциях).

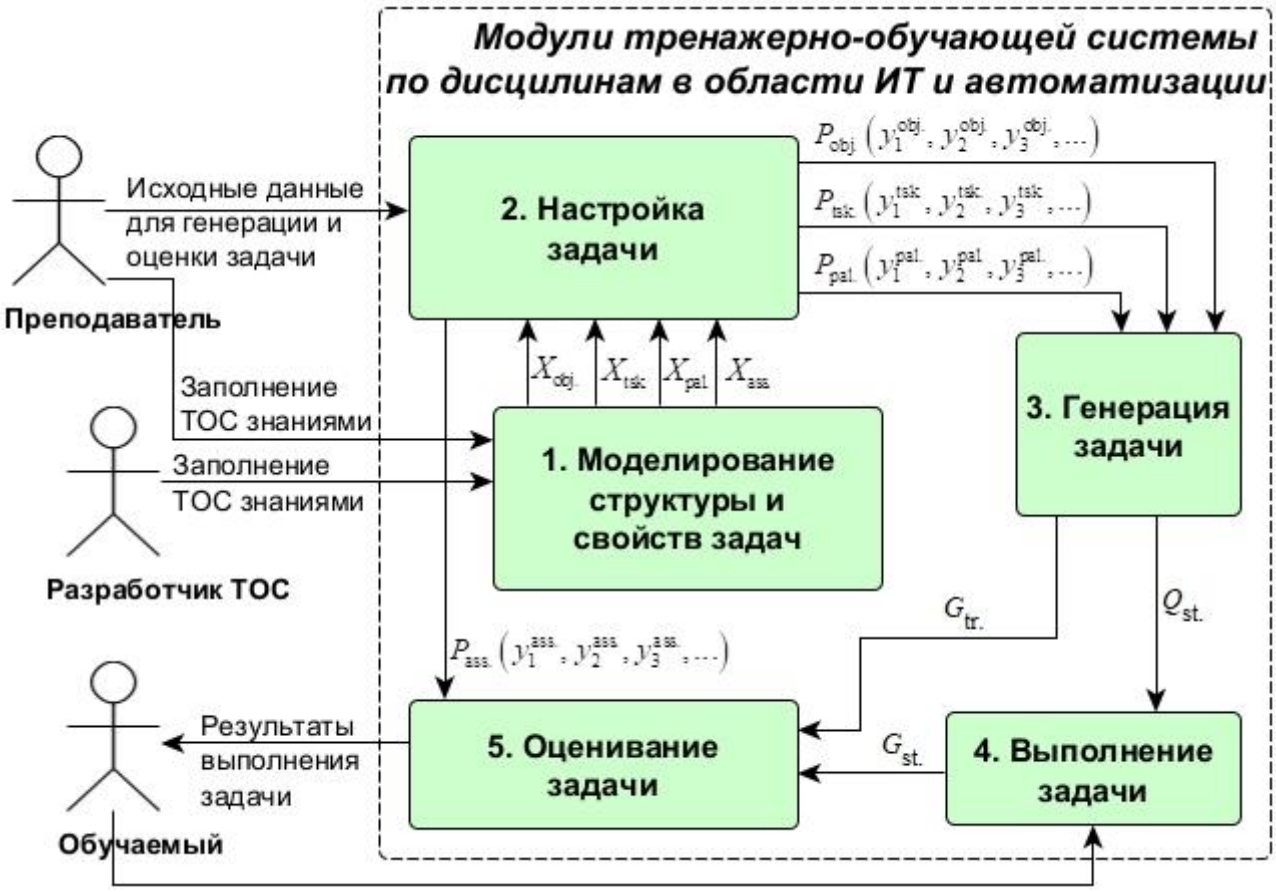


Рис. 1. – Структура и математическая модель ТОС

Ключевыми особенностями ТОС, показанной на рис. 1, являются:

1) настройка преподавателем параметров небольших практических заданий в области ИТ и автоматизации (например, по основам программирования: задания по определению результата выполнения фрагмента программного кода; задания по представлению алгоритма или структуры кода в графическом виде, в частности, в форме потокового графа, блок-схемы, диаграммы Activity UML);

2) генерация каждому студенту индивидуального варианта данного задания в соответствии с настроенными параметрами;

3) выполнение студентом задания с помощью наглядного пользовательского интерфейса;

4) автоматическое оценивание выполнения задания на основе сравнения с эталонным решением, сформированным системой.

Рассмотрим особенности определения параметров, показанных на рис. 1, на примере настройки, выполнения и оценки заданий по основам программирования. В данном задании, согласно сгенерированному индивидуальному варианту программного кода (например, на Java или Python), студенту требуется определить результат выполнения программы или построить графическую модель программы (например, алгоритм на языке UML, потоковый граф).

Изначально преподаватель выполняет настройку задания, используя множество параметров $X_{\text{obj.}}$ (блоки №№ 1, 2 на рис. 1), определяющих изучаемый объект. В рассматриваемом примере $X_{\text{obj.}}$ – это множество параметров $X_{\text{obj.}}^{\text{prg.}}$ программного кода для генерации индивидуального варианта программы требуемой структуры с целью выполнения задания студентом. Для рассматриваемого примера параметры множества $X_{\text{obj.}}^{\text{prg.}}$ используются для описания структуры кода программы, в частности, определяют наличие или отсутствие операторов ввода, вывода, выбора, циклов, простых или составных условий, комбинаций и вложенности различных алгоритмических конструкций.

Множество параметров $X_{\text{obj.}}^{\text{prg.}}$ представим следующим образом: $X_{\text{obj.}}^{\text{prg.}} = \{x_1^{\text{prg.}}, x_2^{\text{prg.}}, x_3^{\text{prg.}}, x_4^{\text{prg.}}, x_5^{\text{prg.}}, x_6^{\text{prg.}}, x_7^{\text{prg.}}\}$, где $x_1^{\text{prg.}}$ – наличие оператора `if` (1 – да, 0 – нет); $x_2^{\text{prg.}}$ – число простых условий оператора `if`; $x_3^{\text{prg.}}$ – наличие одного из

циклов (1 – `while`, 2 – `do…while`, 3 – `for`, 4 – нет); $x_4^{\text{prg.}}$ – число простых условий цикла; $x_5^{\text{prg.}}$ – вложенность операторов (1 – да, 0 – нет); $x_6^{\text{prg.}}$ – наличие операторов ввода данных в начале кода (1 – да, 0 – нет); $x_7^{\text{prg.}}$ – наличие операторов вывода данных в конце кода (1 – да, 0 – нет). Допустимы дополнительные параметры множества $X_{\text{obj.}}^{\text{prg.}}$, например, возможный диапазон значений переменных программы.

Программный код, генерируемый в индивидуальном варианте задания, определяется предикатом, представляемым обобщенно: $P_{\text{prg.}}\left(y_1^{\text{prg.}}, y_2^{\text{prg.}}, y_3^{\text{prg.}}, ...\right)$. Переменные предиката соответствуют значениям элементов множества $X_{\text{obj.}}^{\text{prg.}}$.

В таблицах 1-3 представлены примеры программного кода в соответствии с шаблоном кода (настроенным преподавателем), определяемым условием на основе комбинации значений параметров множества $X_{\text{obj.}}^{\text{prg.}}$. Условие влияет на сложность структуры программы и выполнения задания обучаемым.

Описание шаблона и примеров программного кода для задачи №1 представлено в таблице 1, для задач №2 и №3 – в таблице 2 и таблице 3 соответственно.

Таким образом, преподавателю не требуется составлять индивидуальный вариант кода программы каждому студенту, поскольку, настроив желаемую структуру кода программных модулей, при выполнении задания автоматически формируется вариант программного кода.

Помимо настройки параметров исходного объекта, преподавателю доступна возможность установить (блоки №№ 1-2 на рис. 1):

1) Множество подзадач в задании (определяется предикатом $P_{\text{tsk.}}\left(y_1^{\text{tsk.}}, y_2^{\text{tsk.}}, y_3^{\text{tsk.}}, ...\right)$ на основе множества параметров $X_{\text{tsk.}}$, отражающих допустимый перечень подзадач). Например, задание по основам

программирования может включать подзадачу для вычисления результата выполнения программы и подзадачу для построения алгоритма программы в виде диаграммы Activity UML.

Таблица № 1

Примеры программ на Java в соответствии с шаблоном (задача №1)

<table>
<tr><th>Описание шаблона для примера задачи</th><th>Примеры программ (кода) на языке Java в соответствии с шаблоном и их особенности</th></tr>
<tr><td rowspan="3">Шаблон задачи №1:<br>– наличие любого вида цикла (<code>while</code>, <code>do-while</code> или <code>for</code>);<br>– отсутствие операторов выбора и вложенных циклов;<br>– простое условие цикла;<br>– наличие операторов ввода данных;<br>– наличие операторов вывода данных.<br><br>Условие, определяющее шаблон:<br>$\left(x_1^{\text{prg.}}=0\right)\wedge\left(x_2^{\text{prg.}}=0\right)\wedge$<br>$\left(x_3^{\text{prg.}}=1\vee x_3^{\text{prg.}}=2\vee x_3^{\text{prg.}}=3\right)\wedge$<br>$\left(x_4^{\text{prg.}}=1\right)\wedge\left(x_5^{\text{prg.}}=0\right)\wedge$<br>$\left(x_6^{\text{prg.}}=1\right)\wedge\left(x_7^{\text{prg.}}=1\right)$</td><td>Пример №1.1 кода программы.<br>Наличие цикла с предусловием (<code>while</code>).<br><pre>int y=sc.nextInt(); // 1
int z=sc.nextInt(); // 1
while (z>=500){ // 2
    z-=5*y; // 3
    y--; // 3
} // 3
System.out.println("y="+y); // 4
System.out.println("z="+z); // 4</pre>Примечание: в приведенных в таблицах 1-3 примерах кода, в форме комментариев присутствует нумерация операторов программы, используемая в заданиях на построение графической модели программы (например, потокового графа).</td></tr>
<tr><td>Пример №1.2 кода программы.<br>Наличие цикла с параметром (<code>for</code>).<br><pre>double x=sc.nextDouble(); // 1
int N=sc.nextInt(); // 1
for (int j=N /*2*/; j>15 /*3*/; j-- /*5*/)
    x+=5.75*j-1; // 4
System.out.println("x="+x); // 6</pre></td></tr>
<tr><td>Пример №1.3 кода программы.<br>Наличие цикла с постусловием (do-<code>while</code>).<br><pre>int x1=sc.nextInt(); // 1
int x2=sc.nextInt(); // 1
do // 2
    x1+=2*x2; // 2
while (x1<=175); // 3
System.out.println("x1="+x1); // 4</pre></td></tr>
</table>

2) Палитру для построения графической модели объекта (определяется предикатом $P_{\text{pal.}}\left(y_1^{\text{pal.}},y_2^{\text{pal.}},y_3^{\text{pal.}},...\right)$ на основе множества параметров $X_{\text{pal.}}$, отражающих свойства палитры для построения модели студентом).

Таблица № 2

Примеры программ на Java, в соответствии с шаблоном (задача №2)

<table>
<tr>
<td rowspan="3">Шаблон задачи №2:<br>– наличие оператора выбора <code>if</code>;<br>– отсутствие циклов и вложенных операторов выбора;<br>– составное условие оператора <code>if</code> из 2-х простых условий;<br>– наличие операторов ввода данных;<br>– наличие операторов вывода данных.<br><br>Условие, определяющее шаблон:<br>$\left(x_1^{\text{prg.}}=1\right)\wedge\left(x_2^{\text{prg.}}=2\right)\wedge$<br>$\left(x_3^{\text{prg.}}=0\right)\wedge\left(x_4^{\text{prg.}}=0\right)\wedge$<br>$\left(x_5^{\text{prg.}}=0\right)\wedge\left(x_6^{\text{prg.}}=1\right)\wedge$<br>$\left(x_7^{\text{prg.}}=1\right)$</td>
<td>Пример №2.1 кода программы.<br>Операция дизъюнкции в составном условии.<br><pre>int y1=sc.nextInt(); // 1
int y2=sc.nextInt(); // 1
if (y1<60 /*2*/ || y2==300 /*3*/)
    y2*=3; // 4
else { // 5
    y1-=10; // 5
    y2%=2; // 5
} // 5
System.out.println("y1="+y1); // 6
System.out.println("y2="+y2); // 6</pre></td>
</tr>
<tr>
<td>Пример №2.2 кода программы.<br>Операция конъюнкции в составном условии.<br><pre>int x=sc.nextInt(); // 1
int y=sc.nextInt(); // 1
if (y>2*x /*2*/ && x>=25 /*3*/) {
    x+=y; // 4
    --y; // 4
} // 4
else // 5
    y*=2*x; // 5
System.out.println("x="+x); // 6
System.out.println("y="+y); // 6</pre></td>
</tr>
<tr>
<td>Пример №2.3 кода программы.<br>Операция дизъюнкции в составном условии (по сравнению с примером 2.1, отличаются названия переменных, операции отношений, наличие только одного оператора в ветви <code>else</code>).<br><pre>int w1=sc.nextInt(); // 1
int w2=sc.nextInt(); // 1
if (w1>=w2 /*2*/ || w2<=7 /*3*/)
    w1=w2/2; // 4
else // 5
    w2=w1+9; // 5
System.out.println("w1="+w1); // 6
System.out.println("w2="+w2); // 6</pre></td>
</tr>
</table>

3) Процесс оценивания результатов выполнения задания (определяется предикатом $P_{\text{ass.}}\left(y_1^{\text{ass.}},y_2^{\text{ass.}},y_3^{\text{ass.}},...\right)$ на основе множества параметров $X_{\text{ass.}}$, отражающих свойства палитры для построения модели студентом).

Таблица № 3

Примеры программ на Java в соответствии с шаблоном (задача №3)

<table>
<tr>
<td rowspan="3">Шаблон задачи №3:<br>– наличие оператора выбора <code>if</code>;<br>– наличие одного из 2-х видов цикла (<code>while</code> или <code>do-while</code>);<br>– простые условия операторов <code>if</code>, <code>while</code>, <code>do-while</code>;<br>– наличие вложенности операторов (оператор <code>if</code> в оператор цикла или наоборот);<br>– наличие операторов ввода данных;<br>– наличие операторов вывода данных.<br><br>Условие, определяющее шаблон:<br>$\left(x_1^{prg.}=1\right)\wedge\left(x_2^{prg.}=1\right)\wedge$<br>$\left(x_3^{prg.}=1\vee x_3^{prg.}=2\right)\wedge$<br>$\left(x_4^{prg.}=1\right)\wedge\left(x_5^{prg.}=1\right)\wedge$<br>$\left(x_6^{prg.}=1\right)\wedge\left(x_7^{prg.}=1\right)$</td>
<td>Пример №3.1 кода программы.<br>В оператор <code>if</code> вложен оператор <code>while</code>.<br><pre>
int a=sc.nextInt(); // 1
int b=sc.nextInt(); // 1
if (a>=100){ // 2
    while (b<50){ // 3
        a-=3; // 4
        b+=7; // 4
    } // 4
} // 5
System.out.println("a="+a); // 6
System.out.println("b="+b); // 6
</pre></td>
</tr>
<tr>
<td>Пример №3.2 кода программы.<br>В оператор <code>while</code> вложен оператор <code>if</code>.<br><pre>
int a=sc.nextInt(); // 1
int b=sc.nextInt(); // 1
while (b>200){ // 2
    if (a!=15) // 3
        a--; // 4
    b-=25; // 5
    a+=4; // 5
} // 5
System.out.println("a="+a); // 6
System.out.println("b="+b); // 6
</pre></td>
</tr>
<tr>
<td>Пример №3.3 кода программы.<br>В оператор <code>do-while</code> вложен оператор <code>if</code>.<br><pre>
int a=sc.nextInt(); // 1
int b=sc.nextInt(); // 1
do { // 2
    if (b==20){ // 3
        a/=3; // 4
        b=a+10; // 4
    } // 4
    b++; // 5
    a--; // 5
} while (2*a>b); // 6
System.out.println("a="+a); // 7
System.out.println("b="+b); // 7
</pre></td>
</tr>
</table>

На основе настроек, выполненных в блоках №№ 1-2, далее в блоке №3 формируется индивидуальный вариант задания студенту, определяемый отношением $Q_{st.}$ на основе описанных выше предикатов.

По результатам выполнения задания в блоке №4 студентом формируется решение задания $G_{st.}$. В процессе автоматического оценивания (блок №5) решение обучаемого $G_{st.}$ сравнивается с эталонным решением $G_{tr.}$, сформированным в блоке №3. В зависимости от сложности задания, сравниваются или ответ обучаемого с эталонным ответом (для заданий на определение результата выполнения программы), или построенная обучаемым и эталонная, графические модели программы (согласно алгоритмам, описанным в работах [9-10]).

## 3 Программное обеспечение тренажерно-обучающей системы для контроля навыков по основам программирования

Разработан прототип программного обеспечения ТОС в форме веб-приложения. Рассмотрим основные функциональные возможности ТОС на примере заданий для проверки знаний алгоритмических конструкций на Java.

Преподаватель настраивает задание, состоящее из нескольких задач. В каждой задаче устанавливается структура программы. Также преподаватель указывает вес задачи в зависимости от ее сложности. Макет веб-интерфейса преподавателя для настройки задания показан на рис. 2.

На основе настроек на рис. 2 студенту генерируется индивидуальный вариант задания, в котором для каждой программы необходимо определить выводимое на экран значение переменной. Автоматически определяются названия переменных, константы. В задаче №2 случайным образом определяется, какой вид цикла – `while` или `do-while` – будет предоставлен обучаемому. Для составного условия определяется, как логическая операция – `&&` или `||` – будет соединять простые условия.

Макет интерфейса выполнения задания студентом, в котором отображается пример сгенерированного варианта задания (на основе настроек из рис. 2), показан на рис. 3.

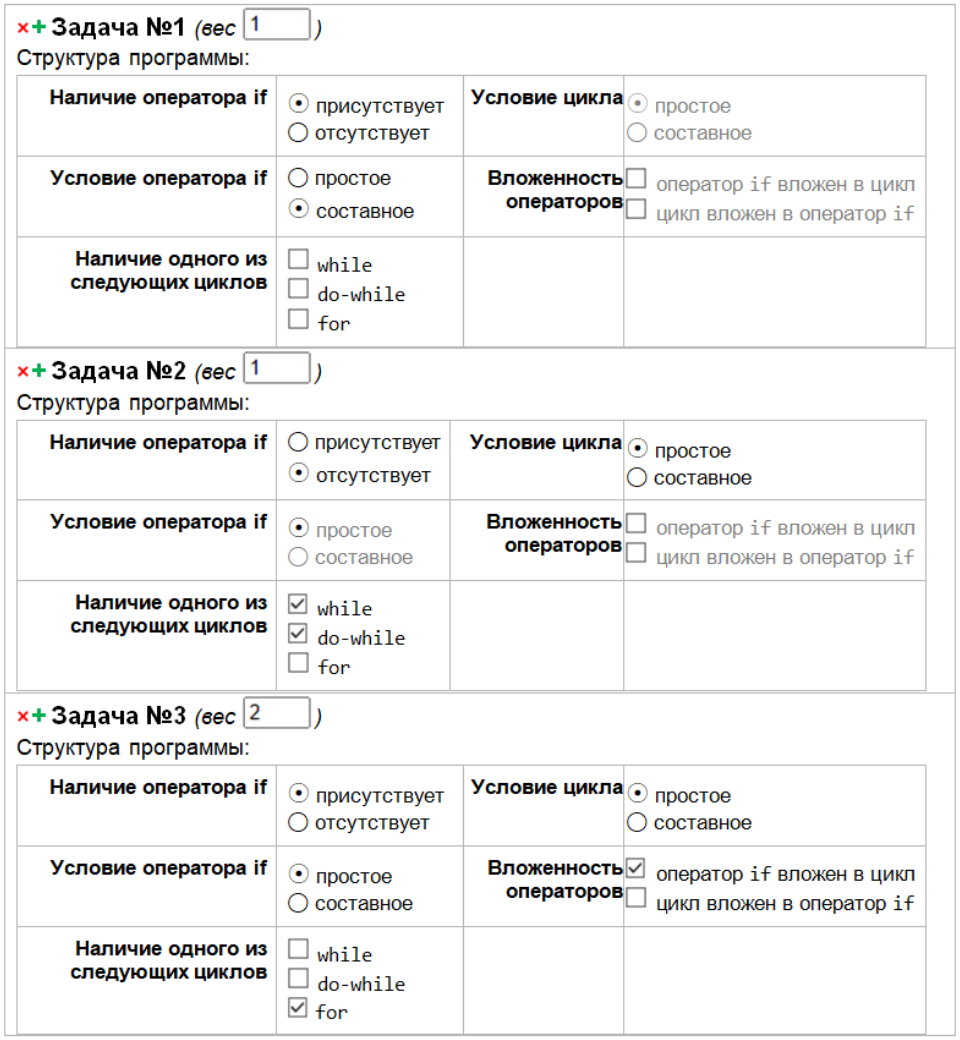


Рис. 2. – Настройка задания по основам программирования

Еще один пример варианта задания (согласно настройкам из рис. 2) представлен на рис. 4.

Предусмотрены интерфейсы просмотра преподавателем результатов выполнения студентами заданий, а студентом – своих результатов.

## 4 Заключение

1) Разработаны структура и математическая модель тренажерно-обучающей системы (ТОС) для контроля навыков при обучении специалистов по информатизации и автоматизации (в частности, по программированию). Параметры математической модели основаны на представлении знаний об изучаемых объектах и процессах в области

разработки автоматизированных информационных систем, что позволяет с применением ТОС автоматически генерировать и оценивать небольшие практические задания для студентов.

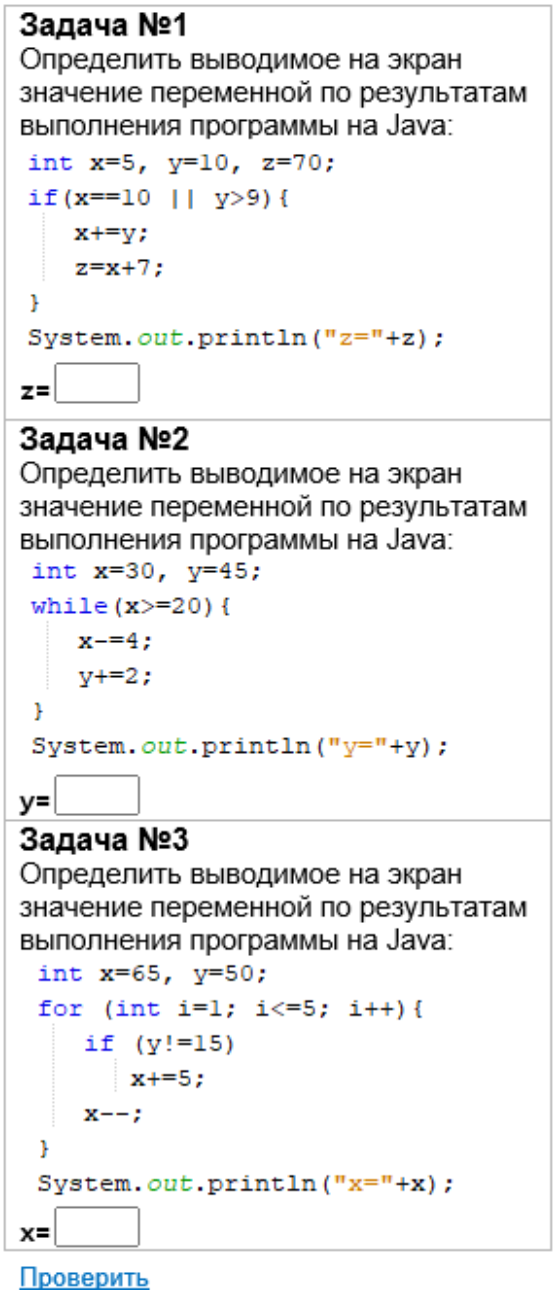


Рис. 3. – Пример №1 варианта задания

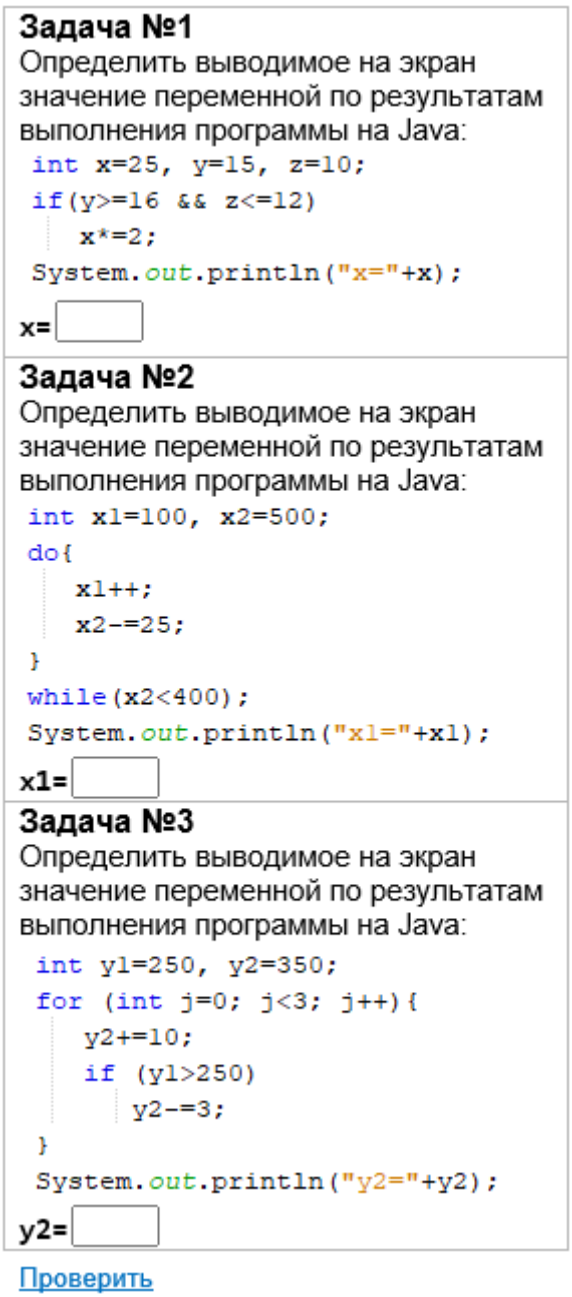


Рис. 4. – Пример №2 варианта задания

2) Разработан прототип программного обеспечения ТОС в форме веб-приложения. На примере практических заданий по определению результатов работы программных модулей на языке Java продемонстрированы

возможности настройки заданий преподавателем и выполнения студентами сгенерированных автоматически вариантов заданий.

3) В практическом аспекте применение ТОС в учебном процессе по дисциплинам в области алгоритмизации и программирования позволит уменьшить долю трудоемкой работы преподавателя по составлению и проверке выполнения заданий. Рассматриваемую ТОС возможно применять в ходе проведения текущего и промежуточного контроля в дополнении к тестированию знаний (например, при защите отчетов по лабораторным и практическим работам, при проведении экзаменов).

4) Перспективами исследования являются совершенствование математического и программного обеспечения ТОС, внедрение ТОС в учебный процесс МГУТУ имени К.Г. Разумовского (ПКУ).

## Литература

1. Глущенко А.Г., Бестужев М.П. Методология организации преподавания дисциплины «Программирование» при переходе к профильным дисциплинам обучения // Современное образование: содержание, технологии, качество. 2020. Т. 1. С. 435-438.

2. Садыкова Ф.Э. Персонализация обучения программированию на примере применения интернет-сервисов // Информатика и образование. 2022. Т. 37. № 6. С. 62-68.

3. Бужинская Н.В. Дистанционное обучение программированию студентов заочного отделения // Научное обозрение. Педагогические науки. 2020. № 6. С. 12-16.

4. Ратанова О.В. Методы создания автоматизированных средств обучения программированию // Прикладная информатика. 2021. Т. 16. № 1 (91). С. 14-21.

5. Федоров А.С., Шиков А.Н. Метод преобразования семантической сети

для автоматизации оценивания решения задач по программированию в процессе электронного обучения // Вестник Астраханского государственного технического университета. Серия: Управление, вычислительная техника и информатика. 2020. № 4. С. 7-17.

6. Кузнецов И.А., Наумова С.Б. Игровые онлайн-платформы как средство обучения программированию // Столица науки. 2019. № 6 (11). С. 270-274.

7. Aldwairi M. Evaluating virtual laboratory platforms for supporting on-line information security courses // Global Journal of Engineering Education. 2022. Vol. 24. №2. pp. 143-148.

8. Полевщиков И.С. Методика разработки практических задач для автоматизированного контроля знаний и навыков при обучении ИТ-специалистов // Инженерный вестник Дона. 2020. №12. URL: ivdon.ru/ru/magazine/archive/n12y2020/6785.

9. Полевщиков И.С., Морозов М.О., Шабалин С.И. Компьютерные тренажерные комплексы для обучения операторов погрузочно-разгрузочных машин навыкам выполнения технологических операций // Инженерный вестник Дона. 2022. №12. URL: ivdon.ru/ru/magazine/archive/n12y2022/8062.

10. Fayzrakhmanov R.A., Polevshchikov I.S. The Use of Mathematical Methods to Automate the Control of Skills in the Study of software Testing Algorithms // Proceedings of 2019 XXII International Conference on Soft Computing and Measurements (SCM): St. Petersburg, Russia, May 23–25, 2019 / IEEE Russia North West Section, Saint Petersburg Electrotechnical Univ. «LETI» (ETU «LETI»). [S. l.]: IEEE, 2019. pp. 107-110. URL: ieeexplore.ieee.org/document/8903779.

## References

1. Glushchenko A.G., Bestuzhev M.P. Sovremennoe obrazovanie:

soderzhanie, tekhnologii, kachestvo. 2020. Vol. 1. pp. 435-438.

2. Sadykova F.E. Informatika i obrazovanie. 2022. Vol. 37. № 6. pp. 62-68.

3. Buzhinskaya N.V. Nauchnoe obozrenie. Pedagogicheskie nauki. 2020. № 6. pp. 12-16.

4. Ratanova O.V. Prikladnaya informatika. 2021. Vol. 16. № 1(91). pp. 14-21.

5. Fedorov A.S., Shikov A.N. Vestnik Astrakhanskogo gosudarstvennogo tekhnicheskogo universiteta. Seriya: Upravlenie, vychislitel'naya tekhnika i informatika. 2020. № 4. pp. 7-17.

6. Kuznetsov I.A., Naumova S.B. Stolitsa nauki. 2019. № 6 (11). pp. 270-274.

7. Aldwairi M. Global Journal of Engineering Education. 2022. Vol. 24. №2. pp. 143-148.

8. Polevshchikov I.S. Inzhenernyj vestnik Dona, 2020, №12. URL: ivdon.ru/ru/magazine/archive/n12y2020/6785.

9. Polevshchikov I.S., Morozov M.O., Shabalin S.I. Inzhenernyj vestnik Dona, 2022, №12. URL: ivdon.ru/ru/magazine/archive/n12y2022/8062.

10. Fayzrakhmanov R.A., Polevshchikov I.S. The Use of Mathematical Methods to Automate the Control of Skills in the Study of software Testing Algorithms. Proceedings of 2019 XXII International Conference on Soft Computing and Measurements (SCM): St. Petersburg, Russia, May 23–25, 2019. IEEE Russia North West Section, Saint Petersburg Electrotechnical Univ. «LETI» (ETU «LETI»). [S. l.]: IEEE, 2019. pp. 107-110. URL: ieeexplore.ieee.org/document/8903779.