УДК 372.8:37.04
DOI

ЛОГИЧЕСКИЕ ПЕРЕМЕННЫЕ КАК ИНСТРУМЕНТ ОПТИМИЗАЦИИ ПРОВЕРКИ УСЛОВИЙ В ПРОГРАММАХ

[1]Козлов С. В. ORCID ID 0000-0001-9945-2098, [2]Быков А. А.

[1]Федеральное государственное бюджетное образовательное учреждение высшего образования «Смоленский государственный университет», Смоленск, Российская Федерация, e-mail: svkozlov1981@yandex.ru;

[2]Филиал Федерального государственного бюджетного образовательного учреждения высшего образования «Национальный исследовательский университет «МЭИ», Смоленск, Российская Федерация

Цель исследования – рассмотреть роль логических переменных в оптимизации проверки условий в программах и сформировать у обучающихся навыки осознанного их использования при написании кода. Изучение математической логики в школе является междисциплинарной темой. В курсе информатики она представляет собой одну из основных дидактических линий. Так, в программировании конструкция ветвления выступает как базовый элемент записи кода при решении задач. Без грамотного ее применения невозможно реализовать запросы в информационных системах, осуществить поиск и отбор необходимой информации. В связи с этим обучающиеся должны в полной мере овладеть навыками эффективного составления логических условий. Они должны уметь интегрировать переменные логического типа в запись составных условий при выборе элементов последовательности, отвечающих заданным критериям, для оптимизации кода. В статье описаны подходы использования логических переменных при формировании условий, когда только один или хотя бы один из компонентов удовлетворяет поисковому запросу. Авторами описаны особенности введения в код программы переменных логического типа, которые позволяют упростить проверку условий, снизить количество ошибок и ускорить отладку. Ввиду этого сокращается время разработки и повышается читаемость программного кода. Это позволяет сформировать у обучающихся профильных классов эффективный инструмент записи логических выражений и оптимально использовать его в программировании с точки зрения производительности. Авторами охарактеризованы методические приемы включения в запись выражений переменных логического типа при обработке данных текстовых файлов. Проанализированы примеры такого применения, описаны особенности составления кода программы на языке Python. Представлены результаты экспериментальной работы по освоению обучающимися навыков оптимизации логических конструкций в программах. Описаны примеры оптимальной записи условий отбора элементов, их пар, троек и четверок при формировании у обучающихся навыков осознанного использования логических переменных.

Ключевые слова: школа, профильные классы, программирование, логические выражения, логические переменные, оптимизация условий, язык Python

BOOLEAN VARIABLES AS A TOOL FOR OPTIMIZING CONDITION CHECKING IN PROGRAMS

[1]Kozlov S. V. ORCID ID 0000-0001-9945-2098, [2]Bykov A. A.

[1]Federal State Budgetary Educational Institution of Higher Education "Smolensk State University", Smolensk, Russian Federation, e-mail: svkozlov1981@yandex.ru;

[2]Branch of the Federal State Budgetary Educational Institution of Higher Education "National Research University MPEI", Smolensk, Russian Federation

The purpose of the study is to consider the role of logical variables in optimizing the verification of conditions in programs and to form in students the skills of their conscious use when writing code. The study of mathematical logic at school is an interdisciplinary topic. In the course of computer science, it represents one of the main didactic lines. So, in programming, the branch construction acts as the basic element of writing code when solving problems. Without its competent application, it is impossible to implement requests in information systems, search for and select the necessary information. In this regard, students must fully master the skills of effective preparation of logical conditions. They must be able to integrate Boolean variables into the compound condition record when selecting sequence elements that meet specified criteria for code optimization. The article describes approaches to the use of logical variables in the formation of conditions when only one or at least one of the components satisfies the search query. The authors describe the peculiarities of introducing Boolean variables into the program code, which make it possible to simplify condition checking, reduce the number of errors and speed up debugging. In view of this, the development time is reduced and the readability of the program code is increased. This allows you to form an effective tool for recording logical expressions in students of specialized classes and optimally use it in programming from the point of view of performance. The authors have described methods for including logical type variable expressions in the recording when processing text file data. Examples of such applications are analyzed, the features of drawing up program code in Python are described. The results of experimental work on mastering the skills of optimizing logical designs in programs by students are presented. Examples of optimal recording of selection conditions for elements, their pairs, triples and fours in the formation of students' skills of conscious use of logical variables are described.

Keywords: school, profile classes, programming, logical expressions, logical variables, optimization of conditions, Python language

Введение

Изучение математической логики в школьном курсе информатики занимает центральное место. Понятия темы формируют одну из основных линий школьной дидактики, а ее методы используются не только в разных приложениях информатики, но и во многих смежных областях науки [1, 2]. Например, в физике математическая логика позволяет построить и описать ряд моделей физических явлений и процессов. В математике логика составляет целый раздел, который задает систему логических обозначений и определяет доказуемость суждений. Этот раздел является общим как для математики, так и для информатики.

В информатике инструменты математической логики применяются для записи логических выражений в программировании и при формировании запросов в базах данных [3, 4]. В теоретической информатике логика позволяет описать модели хранения, передачи и обработки данных. В теории алгоритмов и параллельных вычислениях задает условия и принципы их выполнения с помощью систем функциональных и логических зависимостей [5]. В искусственном интеллекте она служит для формирования системы принятия решений [6]. Функционал математической логики многогранен ввиду системообразующего набора компонентов, требуемых как при обозначениях условий, так и при анализе объективности выводов причинно-следственных связей между объектами исследований.

В то же время в школьном курсе информатики несмотря на то, что математическая логика выделена как самостоятельная дидактическая единица, ее часто изучают достаточно формально. На уроках информатики рассматривают определение логических функций, формулируют законы логики, определяют правильность выводов суждений, строят таблицы истинности, проводят параллели с операциями на множествах, решают логические задачи с применением диаграмм Эйлера – Венна [7, 8]. Эти и другие компоненты учебного материала и образуют раздел математической логики школьного курса информатики. При этом связи с другими разделами зачастую не демонстрируются, каждый блок теоретических сведений и практических навыков постигают независимо друг от друга. Это характеризует во многом весь курс информатики, для которого взаимное проникновение тем остается лишь на бумаге, а на практике обучающиеся в полной мере не умеют пользоваться средствами, применяемыми ими при решении заданий других разделов.

Цель исследования – рассмотрение роли логических переменных в оптимизации проверки условий в программах и формировании у обучающихся навыков осознанного их использования при написании кода.

Материалы и методы исследования

В педагогическом исследовании для изучения причинно-следственных связей обучения применению переменных типа *bool* языка программирования Python для записи логических выражений использовались наблюдение и тестирование. В ходе эксперимента происходило обобщение педагогического опыта оперирования на практике логическими константами и переменными в инструментальной системе PyCharm. В исследовании проведены формирующий и констатирующий этапы экспериментальной работы.

Продолжительность педагогического эксперимента составляла 3 месяца. В нем принимали участие обучающиеся 11-х классов Смоленского физико-математического лицея ЯВИР при МИФИ и МЭИ и средней школы № 6 г. Смоленска. В лицее в исследовании были задействованы 19 чел., а в IT-классе школы № 6 – 14 чел.

Результаты исследования и их обсуждение

При изучении дидактической линии информационных технологий при работе с электронными таблицами важно продемонстрировать связь выбора данных в соответствии с заданными условиями. Здесь «вступают» в работу логические функции «И» и «ИЛИ», знание правил выполнения которых позволяет корректно и оптимально сформировать запись для определения каждого из условий задания. При этом необходимо помнить, что в их записи в редакторах электронных таблиц сначала указывают имя используемой функции, а затем записывают в скобках через запятую ее параметры, в отличие от высказываний, когда функция «ставится» между операндов логического выражения. Еще одной широко применяемой логической функцией в записи условий в электронных таблицах является функция «ЕСЛИ», которая определяет выбор альтернативных или множественных действий в той или иной ситуации. Ее достаточно широко используют не только для формирования условия выполнения указанных в задании характеристических свойств набора данных, но и, например, для подсчета количества элементов в таблице, которые им

удовлетворяют. Для этого при использовании функции «ЕСЛИ» в качестве ее параметров указывают «1» и «0» в строках истинности и ложности логического высказывания. Затем в получившемся наборе данных, как правило, отображенных по столбцу электронной таблицы, осуществляют суммирование всех ячеек диапазона значений. Это можно выполнить с использованием функции «СУММ», служащей для вычисления суммы, или данных статистических функций, результаты применения которых по умолчанию отражены в правом нижнем углу строки состояния электронной таблицы. Кроме того, отметим, что «продвинутыми» встроенными функциями с логической составляющей для организации табличных расчетов повышенной сложности служат такие функции, как «СЧЁТ», «СЧЁТЕСЛИ», «СУММЕСЛИ», «МАКСЕСЛИ», «МИНЕСЛИ» и другие [9, 10]. С помощью них можно объединить в одно действие результаты выполнения сразу двух функций – логической функции «ЕСЛИ» и одной из статистических функций.

Также, чтобы оптимизировать процесс записи условий в редакторах электронных таблиц, результаты вычисления значений математических выражений можно формировать с помощью логических констант. Они принимают значения либо «ИСТИНА», либо «ЛОЖЬ». Это позволяет сразу сгенерировать для набора строк или столбцов электронной таблицы ответ в формате «ДА-НЕТ», требуемый в дальнейшем оценки для проверки составного условия. Так, например, при решении задачи о количестве строк в таблице, удовлетворяющих перечню начальных условий, с помощью такого подхода удобно сразу получить результат для подсчета числа элементов. Это обусловлено тем, что в редакторе электронных таблиц имеется возможность интерпретировать логические константы в числовые значения «1» и «0» и использовать уже их в расчетах.

Именно на этом основано использование логических констант при обработке выражений с использованием систем программирования. В большинстве из них поток входных данных является текстовым и его уже при считывании часто бывает необходимо конвертировать в числовой или иной формат. В связи с этим перевод в ходе решения задачи данных из одной формы записи в другую форму является естественным процессом. Это позволяет применять разнообразный функционал для обработки данных, представленных в разном виде. Так, например, в языке программирования Python удобно манипулировать текстовыми, числовыми и логическими типами данных с помощью широкого набора встроенных функций [11, 12]. Для перевода в формат целых чисел и обратно служат функции *int()*, *str()*, а для обработки логических констант можно воспользоваться оператором присваивания значения логического выражения. Например, команда t = (x % 2) == 0 позволяет определить четность числа x. После ее исполнения переменная *t* примет одно из значений *True* или *False* в зависимости от четности исходного числа *x*.

Форма сравнения с помощью отдельных логических констант облегчает способ формирования записи логических выражений при проверке сложных составных условий. Особенно полезным такой подход оказывается в случаях, когда необходимо учитывать количество компонентов списка, удовлетворяющих условию, которое определено в задании. Например, такими формулировками могут быть «только один элемент пары чисел является четным числом» или «хотя бы один элемент тройки чисел оканчивается на восемь». При этом, чем больше проверяется чисел на выполнение условия, тем эффективнее становится запись логических выражений в виде констант. Это значительно сокращает запись условий и преобразует их в связку с помощью конъюнкции и дизъюнкции. Отметим, что выражение можно еще упростить, опираясь на интерпретацию этих логических операций, как логического умножения и логического сложения. Для этого в языке программирования Python используются арифметические операторы «+» и «*», семантика которых в данном случае определяется типом логических констант. Например, выполнение условия только для одного значения из пары чисел может быть отражено, как x + y == 1, а – хотя бы для одного значения из тройки чисел – x + y + z >= 1. В этих ситуациях значения логических переменных принимаются равными «1» или «0» в зависимости от проверки условий при определении логических констант.

Рассмотрим пример (составлен авторами). Текстовый файл содержит целые числа в диапазоне от -10000 до 10000 включительно. Требуется определить в нем количество всех троек подряд идущих чисел, среди которых только одно делится на 5 и оканчивается на 8, при этом их сумма является четным числом кратным минимальному положительному двузначному числу среди всех заданных чисел. В ответе необходимо указать количество найденных троек чисел, а также максимальную сумму чисел из найденных троек.

Представим решение задачи.

```
f = open('example.txt')
a = [int(i) for i in f]
f.close()
mnp2 = min([x for x in a if x > 0 and 10 <= x < 99])
k = 0
mxsum = -10000000
for i in range(len(a) - 2):
    x = a[i] % 5 == 0 and abs(a[i]) % 10 == 8
    y = a[i + 1] % 5 == 0 and abs(a[i + 1]) % 10 == 8
    z = a[i + 2] % 5 == 0 and abs(a[i + 2]) % 10 == 8
    if x + y + z == 1:
        if (a[i] + a[i + 1] + a[i + 2]) % 2 == 0 and (a[i] + a[i + 1] + a[i + 2]) % mnp2 == 0:
            k += 1
            mxsum = max(mxsum, a[i] + a[i + 1] + a[i + 2])
print(k, mxsum)
```

Решение задачи можно разбить на три блока. В первом из них происходит стандартное считывание данных, записанных в файле. Сначала файл открывается на чтение с помощью команды *open()*. Для этого в качестве его параметра указывается имя текстового файла. При этом необходимо помнить, что если файл располагается в другом месте, отличном от расположения файла, содержащего решение задачи, то путь к файлу задается полностью. Затем для обработки данных создается список «*a*», элементами которого являются все числа из текстового файла. Для оперирования ими, как целыми числами, использована команда *int()*, которая переводит значения в числовой формат. В завершение первого шага файл необходимо закрыть, чтобы освободить оперативную память.

Во втором блоке необходимо произвести инициализацию переменных. Заметим, что для переменных, предназначенных для определения количества пар элементов последовательности, удовлетворяющих условию задачи, и максимальной суммы из таких пар элементов, начальные значения задаются стандартным образом. Количество искомых пар элементов до начала момента вычислений должно быть равно нулю, а максимальная сумма должна быть равна или меньше минимальной суммы двух элементов списка. Кроме того, на этом этапе среди всех заданных чисел в файле необходимо вычислить минимальный положительный двузначный элемент. Для этого необходимо организовать цикл по всем элементам списка и выбрать из чисел в диапазоне от 10 до 99 наименьшее значение. После выбора двузначных положительных элементов это можно сделать с помощью встроенной функции *min()*, которая применима к итерируемым объектам, в частности спискам.

В третьем блоке осуществляется собственно поиск пар искомых элементов последовательности чисел. Так как последнюю пару соседних элементов образуют предпоследний и последний элементы, то диапазон для переменной *i* цикла должен содержать *len(a)-1* повторений. В противном случае произойдет выход за границы списка. Для выбора элементов, отвечающих заданным в задаче признакам, необходимо указать ряд условий. Именно здесь целесообразно предварительно сформировать значения логических переменных, которые определяют выполнимость проверки числа из пары на делимость на 5 и на окончание на 8. При этом заметим, что так как среди чисел могут быть отрицательные значения, то для выделения последней цифры числа его необходимо взять по модулю. Для этого в языке программирования Python служит встроенная функция *abs()*. Это обстоятельство обусловлено тем, что в противном случае, в силу определения остатка от деления, например, для числа -7 будет вычислено значение 3 при выполнении команды *(-7 % 10)*, а при исполнении директивы *(abs(-7) % 10)* – 7. Именно последняя запись отвечает поиску чисел, оканчивающихся на указанную в задании цифру. При этом для проверки делимости числа на 5 число пары необязательно рассматривать по модулю. Такое действие было бы излишним и только загромождало бы формируемую запись логической переменной.

Отметим, что формируемое условие будет составным. В нем использовано два простых логических выражения. В связи с этим каждое из них для повышения читабельности программного кода целесообразно взять в круглые скобки. Также для второго числа пары «записывать» условие заново не следует. Для его определения можно скопировать всю структуру предыдущей записи для первого числа пары и поменять значение индексной переменной на *i + 1*. Это следует

из того, что в цикле индексы текущего и следующего за ним числа последовательности задаются как *i* и *i + 1*, а значения соответствующим этим позициям – *a[i]* и *a[i+1]*. Такой подход при формировании условий обеспечит более краткую структуру основного оператора выбора *if*. В нем для отражения условия, что ровно одно число пары соответствует указанному составному условию, будет достаточно в программном коде написать *x + y = 1*. Выполнение этого условия возможно только тогда, когда только одна из логических переменных принимает значение «0», а другая – «1». Это сокращает структуру записи основного логического условия выбора пар соседних элементов, снижая тем самым вероятность ошибки при расстановке скобок для выбора приоритета выполняемых при проверке действий. Остальные условия являются при записи менее громоздкими и не требуют отдельного выделения логических переменных для хранения их значений. В связи с этим их можно не выносить из записи условия оператора *if* в отдельный программный код. Это связано с тем, что при задании очередной переменной для ее хранения выделяется место в оперативной памяти, что сокращает объем свободного пространства. В силу этого определение дополнительных переменных логического типа для записи составных условий следует проводить исходя из принципа оптимальности, когда их введение «облегчает» читабельность структуры записи программного кода и снижает риск некорректной интерпретации критериев отбора элементов в задаче.

Задания такого типа предлагались обучающимся в ходе экспериментальной работы. Существенной характеристикой каждого из них является целесообразность использования переменных логического типа для записи условий. Решение задачи без введения дополнительных логических переменных делает программный код достаточно громоздким, что приводит к скрупулезной проверке правильности расстановки скобок и логических операторов в конструкции формулирования условия. При этом если при исследовании зависимостей между парами соседних элементов такое использование только начинает демонстрировать свой потенциал, то при анализе троек и четверок чисел оно определяет неоспоримое преимущество в компактности программного кода.

Система разноуровневых заданий, представленных в программном комплексе «SmartTeacher» [13, 14], позволяла в ходе экспериментальной деятельности в автоматизированном виде предлагать обучаемым в соответствии с их индивидуальной траекторией обучения упражнения для тренировки от простого к сложному. На начальном этапе обучения задания содержали простой поиск элементов последовательности. Обучающиеся отрабатывали навыки составления и записи простых логических выражений в цикле для каждого элемента последовательности заданных чисел. На втором этапе им предстояло освоить навыки составления простых логических условий для пар соседних чисел последовательности. На третьем этапе для определения условий для пар элементов было необходимо сначала определить один из элементов последовательности, минимальный или максимальный, который удовлетворяет заданным условиям делимости чисел. На четвертом этапе требовалось демонстрировать навыки обработки троек и четверок чисел. На пятом, заключительном этапе обучающимся предлагались разнообразные задания, аккумулирующие в себе разные ситуации, состоящие в поиске как отдельных элементов, так и их подряд идущих пар, троек и четверок элементов, которые содержат в своей записи, заданные цифры и отвечают разным признакам делимости.

Все задания обучающиеся выполняли в среде программирования PyCharm на языке Python. В конце обучения им была предложена контрольная работа, которая состояла из пяти заданий, соответствующих упражнениям на разных этапах обучения. Задания были сгенерированы в среде программного комплекса «SmartTeacher» [15]. Проверка выполнения заданий, как в ходе текущей диагностики, так и в итоговом тестировании, осуществлялась автоматически интеллектуальными средствами этой системы. Результаты итоговой диагностики подтвердили уровень учебных достижений, которые демонстрировали обучающиеся в ходе экспериментальной работы.

Так, 14 чел. достигли высокого уровня усвоения учебного материала, 15 чел. – продвинутого уровня и 4 чел. – базового уровня. Это позволяет говорить о том, что результаты обучения отражают нормальный закон распределения со смещением наблюдаемых показателей в область высоких значений и образованием там плато. Гипотеза исследования состояла в том, что обучение поиску элементов, их пар, троек и четверок в текстовых файлах с использованием логических переменных дает возможность сформировать комплексные знания о логическом типе данных и особенностях его применения в современных языках программирования, таких как Python. Обучающиеся получают необходимую практику совершенствования своих навыков программирования при вы-

боре компонентов, удовлетворяющих условиям простых и сложных запросов.

Качественный анализ условий и результатов эксперимента. Результаты экспериментальной работы позволяют утверждать, что обучающиеся усвоили правила записи выражений при проверке условий и овладели приемами использования логических переменных для выделения отдельных критериев поиска. В зависимости от ситуации они научились оценивать сложность формируемой записи и применять методы ее оптимизации. Для этого при обработке элементов последовательности, исследовании их пар, троек и четверок обучающиеся разбивали запрос на отдельные части и формировали сложное условие. При этом в записи составных частей они прибегали к логическим переменным, которые обеспечивали простоту записи и читаемость программного кода. Такой подход раскрывает в полном объеме потенциал оперирования логическим типом данных. Кроме того, это создает необходимую базу знаний о конструкции пользовательских запросов при работе с базами данных информационных систем.

Задания, которые выполняли обучающиеся, также учитывали ошибки, типичные для экзаменуемых на ЕГЭ по информатике. Это позволяет выработать у них строгость и наглядность написания программного кода, когда условия поиска содержат разнообразные критерии отбора элементов. Также отметим, что это позволило дополнительно изучить функциональные инструменты языка программирования Python, выработать подходы к оформлению структуры программы, отражающей алгоритмическую логику решения. Это позволяет утверждать, что гипотеза экспериментального исследования находит свое подтверждение в проделанной работе.

Заключение

Таким образом, осуществленная в ходе эксперимента работа демонстрирует целесообразность использования логических переменных для записи логических выражений при проверке условий для их оптимизации. Формирование у обучающихся наглядных представлений о структуре программного кода при записи логических условий позволяет им сократить время отладки алгоритма и повысить его эффективность. Это дает возможность привить будущим IT-специалистам на этапе обучения в профильных классах целостное восприятие компонентов условий обработки данных и показать им обоснованность применяемых методов на практике.

Список литературы

1. Колобов А. Н. Особенности обучения элементам математической логики в средней школе // Мир науки, культуры, образования. 2023. № 4 (101). С. 106–108. DOI: 10.24412/1991-5497-2023-4101-106-108.

2. Козлов С. В., Быков А. А. Особенности изучения междисциплинарных тем школьных курсов математики и информатики с помощью методов математического моделирования // Проблемы современного образования. 2021. № 5. С. 250–261. DOI: 10.31862/2218-8711-2021-5-250-261.

3. Попов В. С. Изучение законов алгебры логики в симуляторе логических схем // Информатика в школе. 2025. Т. 24. № 6. С. 45–51. DOI: 10.32517/2221-1993-2025-24-6-45-51.

4. Козлов С. В., Быков А. А. Обучение анализу логических выражений с использованием языка программирования Python // Современные наукоемкие технологии. 2025. № 4. С. 121–126. URL: https://top-technologies.ru/ru/article/view?id=40375 (дата обращения: 20.02.2026). DOI: 10.17513/snt.40375.

5. Булах Д. А., Казеннов Г. Г., Лапин А. В. Использование модификации алгоритма работы сетей Петри для функционального моделирования логических схем, представленных на вентильном уровне // Известия высших учебных заведений. Электроника. 2017. Т. 22. № 4. С. 379–385. DOI: 10.24151/1561-5405-2017-22-4-379-385.

6. Любивая Т. Г. Математическая логика как средство формализации задач искусственного интеллекта // Школа университетской науки: парадигма развития. 2020. № 2 (36). С. 48–50. EDN: RZXXVF.

7. Иванова А. В., Митющенко Е. В. Урок информатики в системно-деятельностном подходе по теме «Элементы алгебры логики» // Информатика в школе. 2023. № 2 (181). С. 13–24. DOI: 10.32517/2221-1993-2023-22-2-13-24.

8. Меньшиков В. В. Урок на тему «Элементы схемотехники. Логические схемы» // Информатика в школе. 2022. № 4 (177). С. 56–69. DOI: 10.32517/2221-1993-2022-21-4-56-69.

9. Попов В. С., Алефиренко Е. А., Черницына Л. Ю. Компетенции для вычислений в электронных таблицах: нахождение минимальных и максимальных значений по условию, функции МИНЕСЛИ, МАКСЕСЛИ // Вестник МГПУ. Серия: Информатика и информатизация образования. 2025. № 1 (71). С. 90–100. DOI: 10.24412/2072-9014-2025-171-90-100.

10. Потехина Е. В. Поддержка курса математической логики средствами информационных технологий // Мир науки, культуры, образования. 2018. № 3 (70). С. 329–330. EDN: XUNFCP.

11. Кондратьева В. А. Обучение основам программирования на языке Python в школьном курсе информатики // Вестник МГПУ. Серия: Информатика и информатизация образования. 2021. № 1 (55). С. 8–16. DOI: 10.25688/2072-9014.2021.55.1.01.

12. Каракозов С. Д., Маняхина В. Г. Python как базовый язык обучения программированию в школе // Информатика в школе. 2020. № 1 (154). С. 26–30. DOI: 10.32517/2221-1993-2020-19-1-26-30.

13. Киселева О. М. Использование математических методов для формализации элементов образовательного процесса // Концепт. 2013. № 2. С. 51–57. URL: http://e-koncept.ru/2013/13032.htm (дата обращения: 13.03.2026).

14. Быков А. А., Козлов С. В. Интеллектуальное формирование наборов тестовых заданий с использованием web-платформы “smartteach” // Естественные и технические науки. 2025. № 5 (204). С. 147–149. EDN: GQUHIM.

15. Быков А. А., Козлов С. В., Исаков М. А. Использование web-платформы “smart teach” как инструмента построения образовательных карт // Естественные и технические науки. 2024. № 10 (197). С. 171–173. EDN: BHYICU.

Конфликт интересов: Авторы заявляют об отсутствии конфликта интересов.

Conflict of interest: The authors declare that there is no conflict of interest.