DOI: 10.32517/2221-1993-2023-22-3-61-73

В. К. Маркелов, О. А. Завьялова
Шуйский филиал Ивановского государственного университета, г. Шуя, Ивановская область, Россия

ВОЗМОЖНОСТИ ИГРОВЫХ ДИАЛОГОВЫХ ПРОГРАММ ДЛЯ ОБУЧЕНИЯ РЕШЕНИЮ ТИПОВЫХ ЗАДАЧ РАЗДЕЛА «ПРОГРАММИРОВАНИЕ» В ШКОЛЬНОМ КУРСЕ ИНФОРМАТИКИ

Аннотация

Сегодня в школе обучению программированию уделяется недостаточно внимания, так как оно вызывает у учащихся множество трудностей, связанных с непониманием абстрактного и сложного для усвоения нового материала. При этом слабый уровень мотивации к изучению и в дальнейшем к преподаванию раздела «Программирование» является одной из проблем подготовки будущих учителей информатики. В качестве средства мотивации как для обучающихся, так и для будущих учителей информатики могут использоваться игровые диалоговые программы (игры, в которых игрок вступает в диалог с компьютером). Такие программы могут применяться учителями для отработки навыков у обучающихся по решению типовых задач раздела «Программирование».

В статье рассматриваются возможности применения различных разновидностей игровых диалоговых программ (текстовые квесты, игры со случайными числами. игра в кости, быки и коровы) для обучения решению типовых задач раздела «Программирование». В частности, данные игры могут использоваться в рамках изучения алгоритмов обработки данных с использованием ветвлений и циклов, алгоритмов поиска максимума и минимума, алгоритмов выделения цифр из натурального числа. При этом следует отметить, что эти программы могут быть адаптированы для обучения решению типовых задач раздела «Программирование» на базе других учебных языков программирования, таких как Pascal, Java, C++ и т. д.

Ключевые слова: программирование, игровые диалоговые программы, методика обучения программированию, Python, информатика.

1. Введение

Проблемы преподавания раздела «Программирование» в рамках школьного курса информатики обусловлены существенными изменениями курса информатики, к числу которых относится переход на новые федеральные государственные образовательные стандарты. Согласно Примерной рабочей программе основного общего образования по информатике, составленной на основе требований к результатам освоения основной

Контактная информация

Маркелов Валерий Константинович, магистрант 2-го года обучения по направлению подготовки 44.04.01 «Педагогическое образование» (профиль «Информационные технологии в профессиональной деятельности педагога»), Шуйский филиал Ивановского государственного университета, г. Шуя, Ивановская область, Россия; *адрес:* 155908, Россия, Ивановская область, г. Шуя, ул. Кооперативная, д. 24; *e-mail:* v.a.l.e.m.a.r.k@yandex.ru

Завьялова Ольга Алексеевна, канд. пед. наук, доцент кафедры математики, информатики и методики обучения, Шуйский филиал Ивановского государственного университета, г. Шуя, Ивановская область, Россия; *адрес:* 155908, Россия, Ивановская область, г. Шуя, ул. Кооперативная, д. 24; *e-mail:* ooolga30@yandex.ru

V. K. Markelov, O. A. Zavyalova
Shuya branch of Ivanovo State University, Shuya, Ivanovo Region, Russia

THE POSSIBILITIES OF GAME DIALOG PROGRAMS FOR TEACHING THE SOLUTION OF TYPICAL TASKS OF THE "PROGRAMMING" SECTION OF THE SCHOOL INFORMATICS COURSE

Abstract

Today, not enough attention is paid to teaching programming in schools, as it causes many difficulties for students associated with a misunderstanding of the abstract and difficult to assimilate new material. At the same time, a weak level of motivation to study and, in the future, to teach the section "Programming" is one of the problems of training future informatics teachers. As a means of motivation for both students and future informatics teachers, game dialogue programs (games in which the player enters into a dialogue with the computer) can be used. Such programs can be used by teachers to develop students' skills in solving typical problems in the "Programming" section.

The article discusses the possibilities of using various types of game dialogue programs (text quests, games with random numbers, dice, bulls and cows) for learning to solve typical problems in the "Programming" section. In particular, these games can be used in the study of data processing algorithms using branches and loops, algorithms for finding the maximum and minimum, algorithms for extracting digits from a natural number. At the same time, it should be noted that these programs can be adapted for learning to solve typical problems of the "Programming" section based on other educational programming languages, such as Pascal, Java, C++, etc.

Keywords: programming, game dialogue programs, programming teaching methods, Python, informatics.

образовательной программы основного общего образования, раздел «Алгоритмы и программирование» является одним из тематических разделов, определяющих структуру основного содержания учебного предмета «Информатика» [8].

В соответствии с обновленным ФГОС основного общего образования на основе требований к освоению предметных результатов по учебному предмету «Информатика» на базовом уровне можно выделить следующие типовые задачи раздела «Программирование» [7]:

- алгоритмы для управления исполнителями (Черепаха, Чертежник);
- алгоритмы обработки данных с использованием ветвлений и циклов;
- алгоритмы проверки делимости одного числа на другое;
- алгоритмы проверки натурального числа на простоту;
- алгоритмы выделения цифр из натурального числа;
- алгоритмы поиска максимума и минимума;
- алгоритм суммы числовой последовательности.

Обучение школьников программированию довольно часто вызывает у них множество трудностей, связанных с непониманием абстрактного и сложного для усвоения нового материала [5]. Поэтому успеваемость учащихся обычно снижается и интерес к изучению программирования падает, что негативно влияет на мотивацию к изучению информатики в целом. В том числе это связано с тем, что программирование является одним из самых сложных разделов школьного курса информатики для изучения и восприятия обучающимися [4]. Также слабый уровень мотивации к изучению и в дальнейшем к преподаванию раздела «Программирование» является одной из проблем подготовки будущих учителей информатики, для решения которой требуется модернизация существующих методик преподавания программирования [3].

Чтобы мотивировать обучающихся к изучению программирования, развивать в процессе обучения аналитическое и алгоритмическое мышление, нужно использовать задания, которые вызывают у школьников наибольший интерес: дети любят играть в игры и заинтересованы в том, чтобы разобраться, как они устроены [2]. Под *алгоритмическим мышлением* понимается совокупность мыслительных действий и приемов, нацеленных на решение задач, в результате которых создается алгоритм, являющийся специфическим продуктом человеческой деятельности [9]. Поэтому в качестве средства мотивации как для обучающихся, так и для будущих учителей информатики могут использоваться игровые диалоговые программы.

Особенность игровой диалоговой программы состоит в том, что результат ее работы напрямую зависит от того, как пользователь (игрок) взаимодействует с этой программой, т. е. от того, какие данные игрок вводит в программу (какой устанавливается «диалог» между игроком и игровой программой) [6]. Такие программы могут использоваться учителями как для мотивации обучающихся к обучению программированию, так и для отработки у обучающихся навыков решения типовых задач раздела «Программирование».

2. Разновидности игровых диалоговых программ для решения типовых задач раздела «Программирование»

Рассмотрим возможности использования различных разновидностей игровых диалоговых программ для решения типовых задач раздела «Программирование». Рабочие примеры данных программ на базе языка программирования Python доступны для скачивания и тестирования посредством онлайн-среды программирования Trinket.io по следующей ссылке:
https://sites.google.com/view/game-dialogue-programs/main

2.1. Текстовые квесты

Одной из наиболее простых, но в то же время интересных и творческих разновидностей игровых диалоговых программ, которую можно использовать для обучения программированию, является текстовый квест. Под *текстовым квестом* понимается игра, сюжет которой напрямую зависит от действий игрока. Обучающимся будет интересно придумать собственную историю и возможные варианты того, как она будет развиваться.

2.1.1. Программа «Прогулка»

В качестве примера рассмотрим одну из таких историй.

В ней мальчик Вася в прекрасный солнечный день решает прогуляться, и перед ним встает выбор: отправиться в парк или к реке. В зависимости от совершенного выбора у истории будут различные варианты развития событий или последствия — «концовки». Данная история выглядит следующим образом.

Начало истории. Проснувшись рано утром, Вася выглянул в открытое окно. Ясное и безоблачное небо распахивалось над головой, как ковер, привлекая внимание красотой и яркостью. Лучи солнца освещали все вокруг и обогащали цвета природы. Трава, деревья и цветы блестели на солнце. Птицы чирикали и пели свои песни, совершенно не обращая внимания на прохожих на земле. Воздух был свежим и легким, наполняя легкие чистым и приятным запахом свежей зелени.

Поэтому Вася решил прогуляться и задумался, куда отправиться: в парк или к реке?

1 — отправиться в парк;

2 — отправиться к реке.

Концовка № 1. Летний парк был настоящим укрытием от солнечной жары. Это было незабываемое место, где можно расслабиться и насладиться воздухом цветущей природы. Петя поразился красоте вокруг: воздух был наполнен ароматами душистых цветов. Парк был в своем расцвете, а зеленые деревья были облиты ярким солнцем. Насладившись прогулкой, Петя отправился навестить своих друзей.

Концовка № 2. Когда Петя добрался до реки, он сразу же заметил красоту природы, которой он окружен. Река была глубокой и узкой, а вода в ней — прозрачной и искристой. На берегу реки были белые камни, которые блестели на солнце. Он решил пройти вдоль реки, чтобы насладиться красотами природы и провести время на свежем воздухе. Счастливый и довольный своей прогулкой, Вася вернулся домой, соскучившись по своим любимым игрушкам, которые ждали его дома.

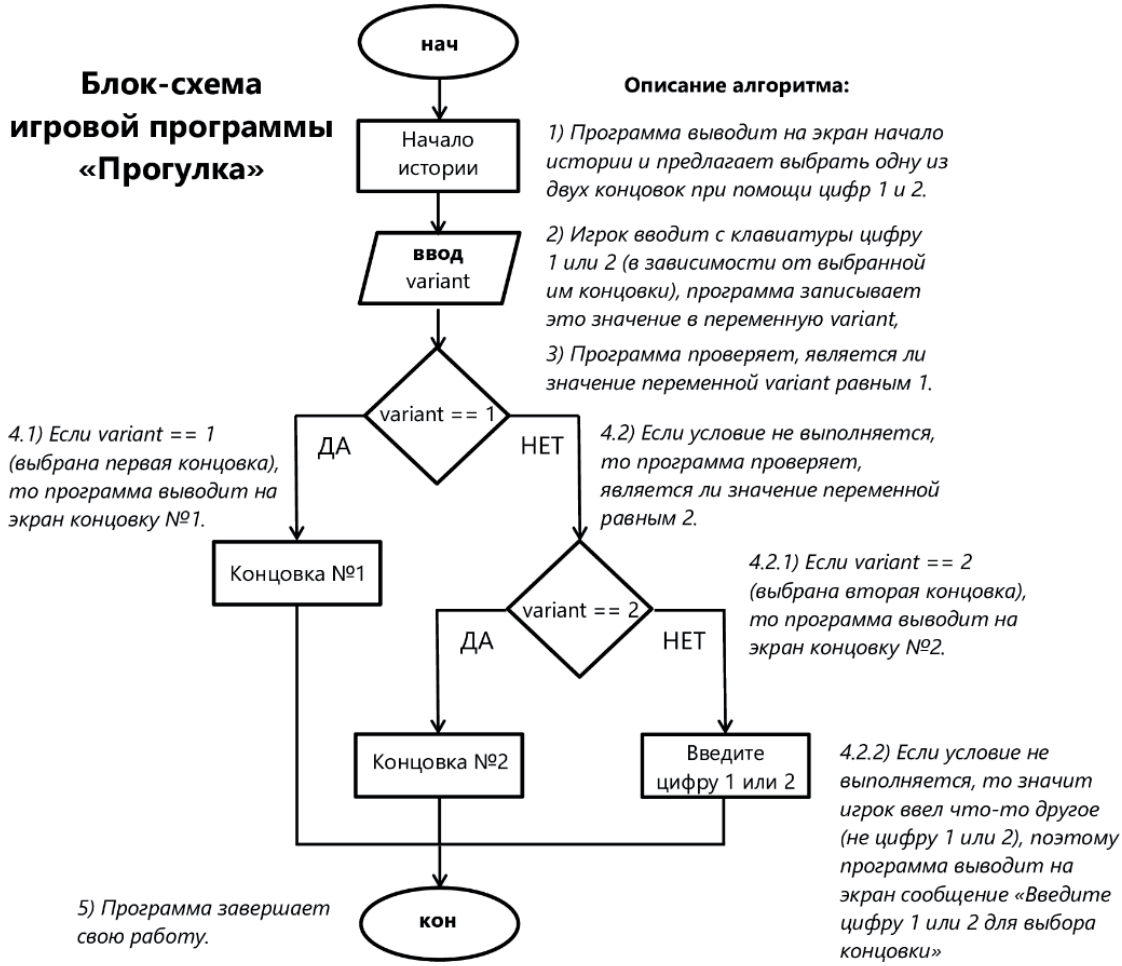

Рис. 1. Блок-схема игровой диалоговой программы «Прогулка»

Блок-схема игровой диалоговой программы «Прогулка» представлена на рисунке 1, а также доступна по ссылке:

https://drive.google.com/file/d/10qGLkdB7HCA2Du9QRwptvok7dnMxrmkv

Данная программа выводит на экран начало истории и предлагает игроку выбрать один из двух вариантов дальнейшего развития событий при помощи цифр 1 и 2 (1 — отправиться в парк, 2 — отправиться к реке). Затем игрок вводит с клавиатуры цифру 1 или 2, чтобы перейти к концовке истории, а программа выводит на экран концовку в зависимости от выбранного варианта. Если игрок ввел другую цифру или символ, то программа выведет на экран сообщение: «Введите цифру 1 или 2 для выбора концовки» и завершит свою работу.

На рисунке 2 представлена демонстрация работы игровой диалоговой программы «Прогулка» на языке Python в онлайн-среде программирования Trinket.io.

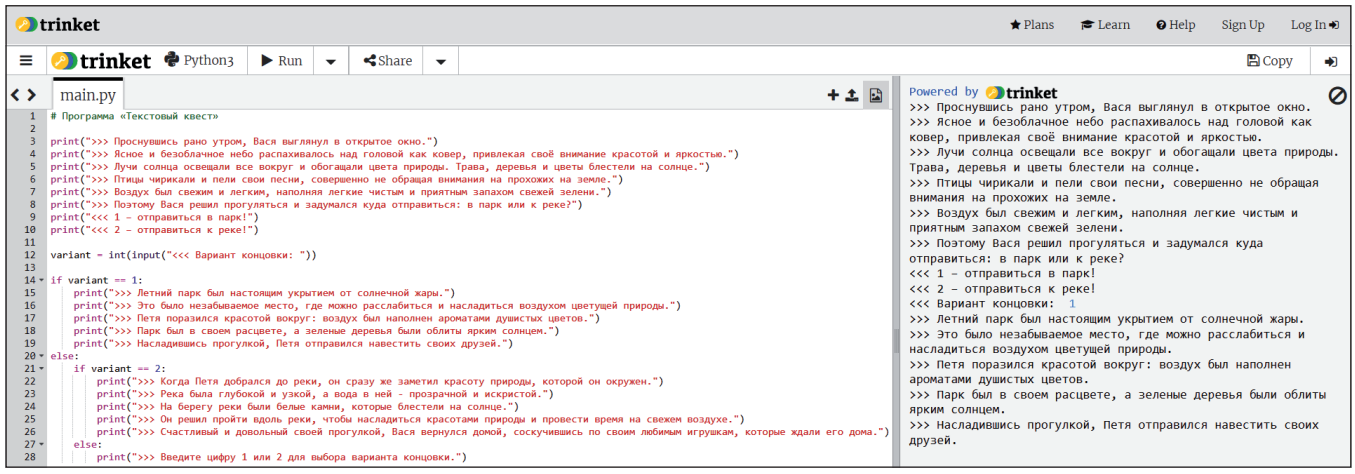

Рис. 2. Демонстрация работы игровой диалоговой программы «Прогулка» в онлайн-среде программирования Trinket.io

Аналогичную игровую диалоговую программу можно использовать при изучении полного условного оператора, а также простых и вложенных условий. Обучающиеся придумывают начало истории (в качестве истории можно взять случай из жизни), а затем два возможных варианта ее развития — «концовки», а после того как придуманы история и варианты ее окончания, учащиеся приступают к разработке программы на учебном языке программирования (Pascal или Python), используя готовую блок-схему [5].

2.1.2. Программа «Таинственный остров»

Более сложным примером текстового квеста является игровая программа, в которой игроку необходимо выбрать один из нескольких вариантов развития событий, которые могут привести к нескольким «концовкам».

В качестве примера такой игры рассмотрим игровую диалоговую программу «Таинственный остров» про исследователя, который приплывает на тропический остров в поисках сокровищ. Исследователь может «перемещаться» по различным частям острова, используя цифры 1, 2, 3 и 4 (1 — северная часть острова, 2 — восточная, 3 — западная, 4 — южная). При этом в зависимости от того, какую часть острова он решит исследовать, будут получены различные концовки.

Таким образом, история для данной игровой диалоговой программы выглядит следующим образом.

Начало истории. Далекий тропический остров таит в себе множество тайн и загадок. Исследователь приплывает на остров, чтобы изучить его и отыскать сокровище, которое находится где-то на острове. Исследователь может перемещаться по различным частям острова, используя цифры 1, 2, 3, 4:

1 — перейти в северную часть острова;
2 — перейти в восточную часть острова;
3 — перейти в западную часть острова;
4 — перейти в южную часть острова.

Концовка № 1. Исследователь направился в северную часть острова, где отыскал легендарное сокровище острова! После возвращения из путешествия наш герой стал богатым и знаменитым.

Концовка № 2. В восточной части острова исследователь случайно столкнулся с пиратами. Пираты захватили несчастного исследователя в плен и забрали на свой корабль в длительное плавание.

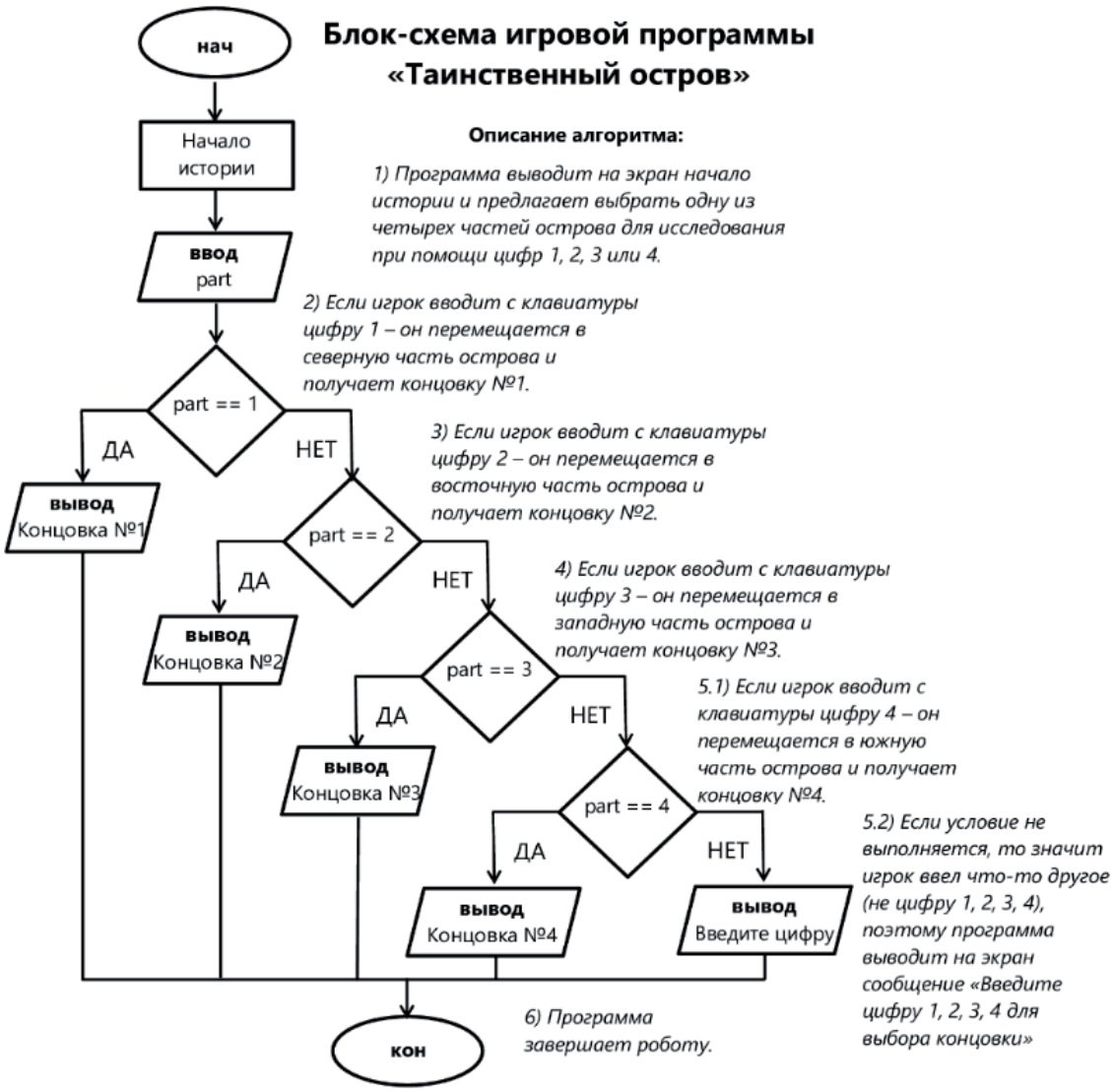

Рис. 3. Блок-схема игровой диалоговой программы «Таинственный остров»

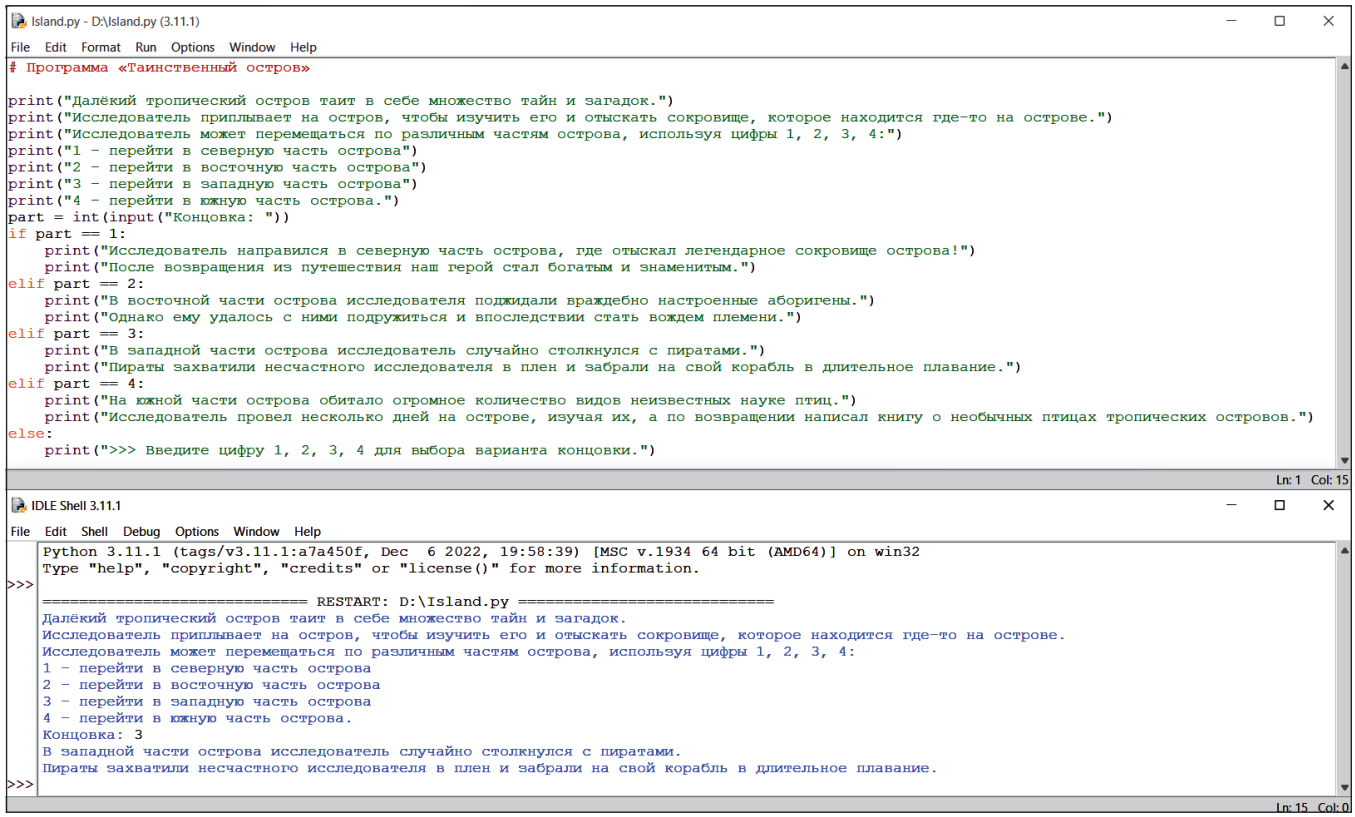

Рис. 4. Демонстрация работы игровой диалоговой программы «Таинственный остров» в среде программирования IDLE Python

Концовка № 3. В западной части острова исследователя поджидали враждебно настроенные аборигены. Однако ему удалось с ними подружиться и впоследствии стать вождем племени.

Концовка № 4. На южной части острова обитало огромное количество птиц неизвестных науке видов. Исследователь провел несколько дней на острове, изучая их, а по возвращении написал книгу о необычных птицах тропических островов.

Блок-схема игровой диалоговой программы «Таинственный остров» представлена на рисунке 3 и доступна по ссылке:

https://drive.google.com/file/d/1KMdHVbroMA7iOFtaOvYFw-G4n8ILUkLq

Так же как и игровая программа «Прогулка», данная программа выводит на экран начало истории, но теперь предлагает игроку выбрать один из нескольких вариантов дальнейшего развития событий (в какую часть острова отправиться исследователю) при помощи цифр 1, 2, 3, 4. Затем игрок вводит с клавиатуры цифру 1 или 2, чтобы перейти к концовке истории, а программа выводит на экран концовку в зависимости от выбранного варианта. Если игрок ввел другую цифру или символ, то программа выведет на экран сообщение: «Введите цифру 1, 2, 3, 4 для выбора концовки» и завершит свою работу.

На рисунке 4 представлена демонстрация работы игровой программы «Таинственный остров» в среде программирования IDLE Python. Данная программа подойдет как для изучения простых и вложенных условий, так и для знакомства обучающихся с каскадным условным оператором в Python, благодаря наличию нескольких «концовок» в игре. При этом сначала можно рассмотреть вариант программы с применением вложенных условий, а потом переделать ее с использованием каскадного условного оператора, показав, что обновленный вариант «читается» проще и позволяет лучше понять, как работает программа.

2.1.3. Создание истории с помощью чат-бота ChatGPT

Если обучающиеся хотят почувствовать себя писателями и написать интересную историю для своего текстового квеста, они могут воспользоваться возможностями чат-бота ChatGPT, отправив ему текстовый запрос: «напиши историю про <идея истории>». Полученный вариант истории может быть неидеален, а также содержать логические, орфографические и синтаксические ошибки, поэтому он должен быть переработан обучающимися совместно с учителем.

Отметим, что история, представленная в игровой диалоговой программе «Прогулка», была создана с использованием возможностей чат-бота ChatGPT.

По состоянию на июнь 2023 года доступ к чату-боту ChatGPT ограничен в Российской Федерации, однако его возможностями можно воспользоваться при помощи ботов ChatGPT в мессенджере Telegram, которые основаны на API OpenAI.

Примерами таких ботов являются:

- ChatGPT 3.5 | Telegram bot (https://t.me/GPT4Telegrambot);
- CHAT GPT FREE bot (https://t.me/GPT_chat_robot);
- ChatGPT Bot (https://t.me/chatsgpts_bot).

Данные боты являются бесплатными для использования, но могут иметь ограничения по скорости и количеству запросов.

Для доступа к ChatGPT также могут использоваться и другие сервисы, разработанные с применением OpenAI API, например: https://chat-gpt.org/ru

2.2. Игры со случайными числами

Одним из типов игровых диалоговых программ, которые также можно использовать для решения типовых задач раздела «Программирование», являются игры, в которых используются случайные числа. Для работы со случайными числами в Pascal для этого используются функция *random* и процедура *randomize*, а в Python – модуль *random*.

2.2.1. Программа «Бросок игральных кубиков»

Простейшей игровой диалоговой программой с использованием случайных чисел является игровая программа «Бросок игральных кубиков».

В данной игре программа, выступающая в качестве игрока, «бросает» два игральных кубика, а игрок в свою очередь должен попытаться угадать сумму чисел, выпавших на кубиках. Компьютер определяет значения игральных кубиков (от 1 до 6) случайным образом, следовательно, сумма чисел, выпавших на кубиках, может принимать значения от 2 до 12.

Блок-схема игровой диалоговой программы «Бросок игральных кубиков» представлена на рисунке 5 и доступна по ссылке:

https://drive.google.com/file/d/1JW1UnAGaiT2qtQ0El5fNR-gs40UDE17b

Данная игровая программа может использоваться для знакомства обучающихся с возможностями работы со случайными числами в рамках изучения программирования (функция *random* в языке Pascal и модуль *random* в языке Python). Кроме того, ее можно использовать в рамках изучения простых условий и полного условного оператора в соответствующем учебном языке программирования.

Демонстрация игровой диалоговой программы «Бросок игральных кубиков» в онлайн-среде программирования Online Python (https://www.online-python.com) представлена на рисунке 6.

2.2.2. Программа «Угадай большее»

Другим вариантом игры с использованием игральных кубиков может быть игровая диалоговая программа «Угадай большее».

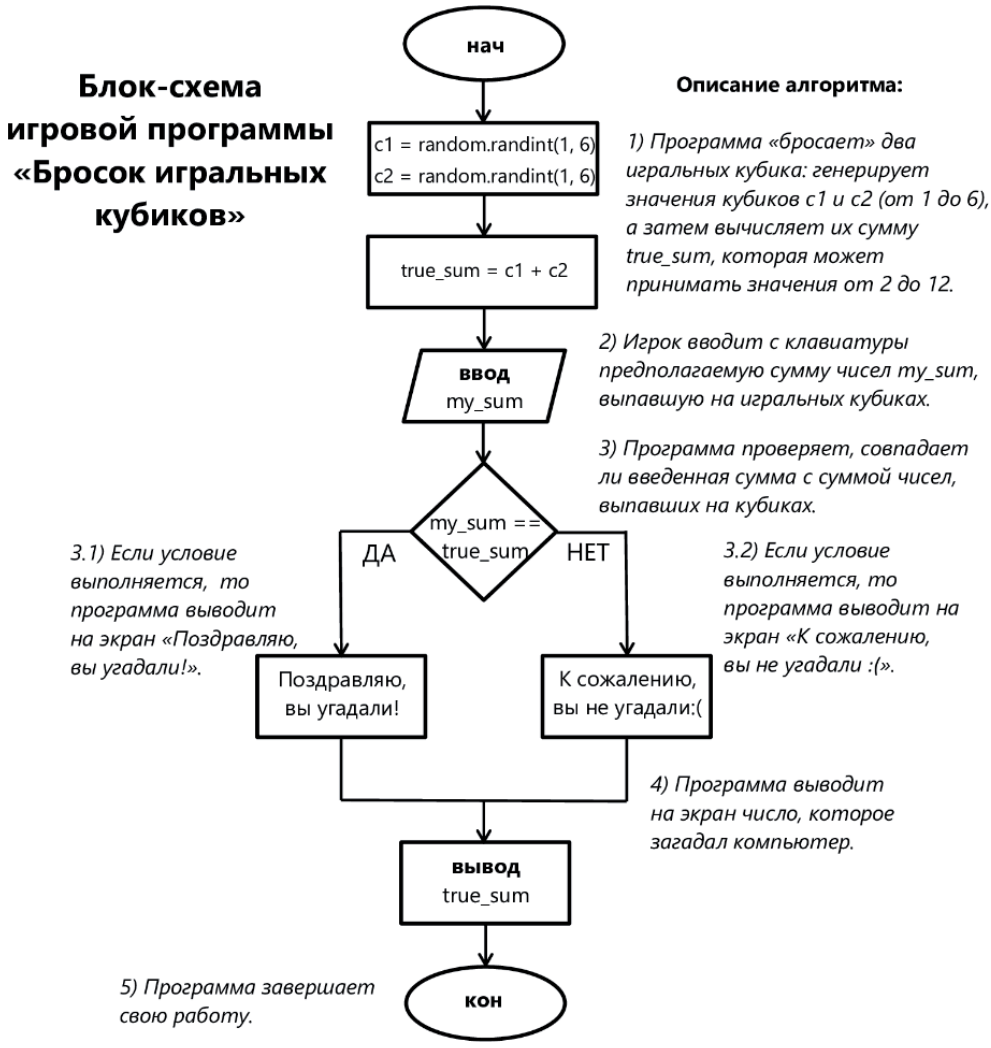

Рис. 5. Блок-схема игровой диалоговой программы «Бросок игральных кубиков»

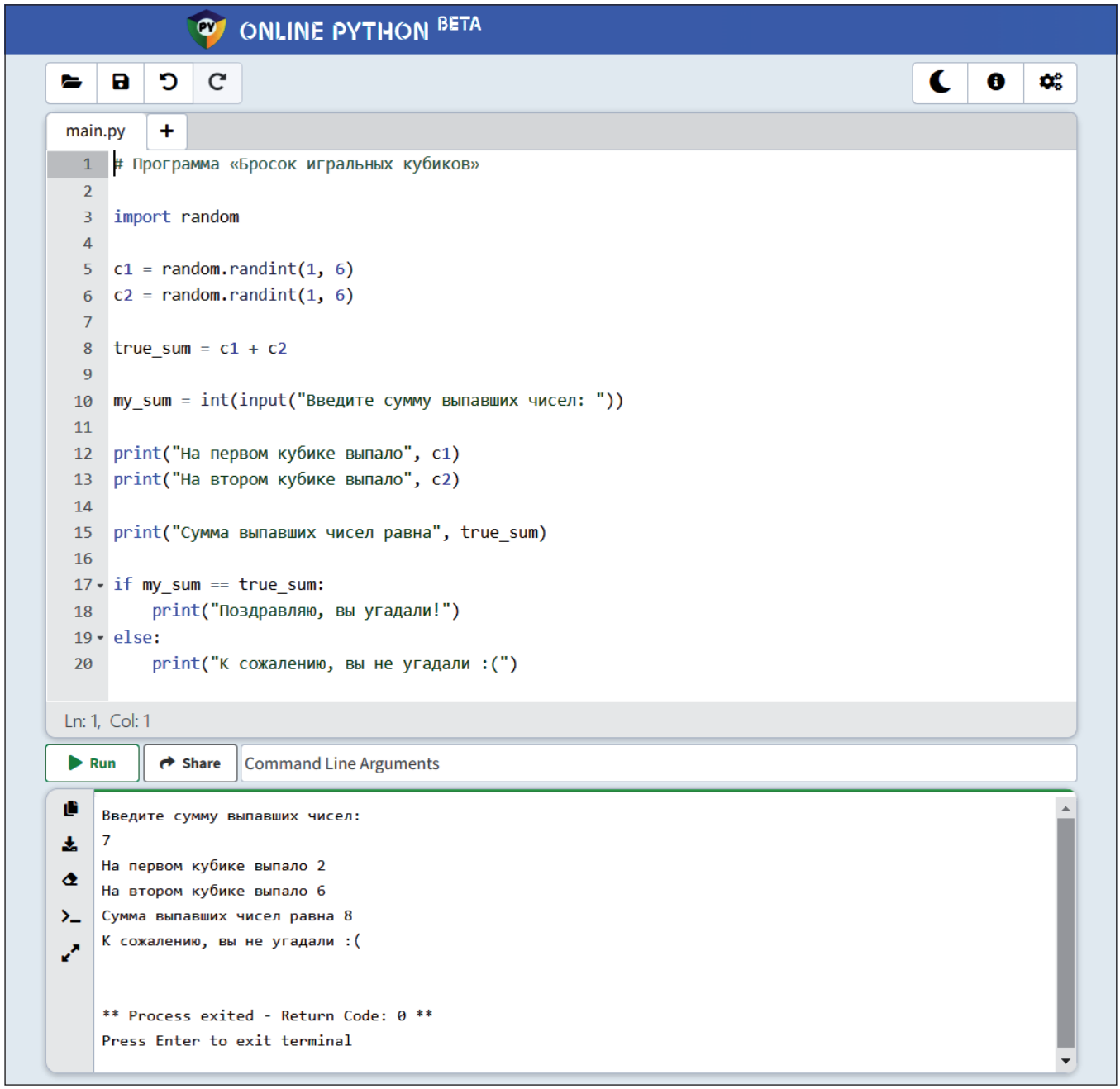

Рис. 6. Демонстрация игровой диалоговой программы «Бросок игральных кубиков» в онлайн-среде программирования Online Python

В данной игре компьютер «бросает» три игральных кубика, а игрок должен угадать, какое наибольшее число выпало на кубике. Для того чтобы шанс угадать наибольшее число не был слишком высоким, можно использовать нестандартные кубики, которые могут принимать больше значений, например, от 1 до 9.

Блок-схема игровой диалоговой программы «Угадай большее» представлена на рисунке 7 и доступна по ссылке:

https://drive.google.com/file/d/1ep8ObXmHDb1KkzewSBTnAZaKFlvefvme

Для написания этой игровой программы необходимо реализовать алгоритм поиска максимума из трех чисел, являющийся одной из типовых задач раздела «Программирование» Данный алгоритм может быть реализован как с использованием неполного условного оператора, так и при помощи каскадного условного оператора и вложенных условий.

В качестве дополнительного задания можно предложить обучающимся упростить игру, дополнив программу следующим образом — перед тем как игрок введет предполагаемое наибольшее число на кубиках, программа выведет на экран:

- сумму чисел, полученных после бросков трех игральных кубиков;
- наименьшее число, полученное на игральном кубике.

Таким образом, дополненный вариант программы можно использовать для формирования умения реали-

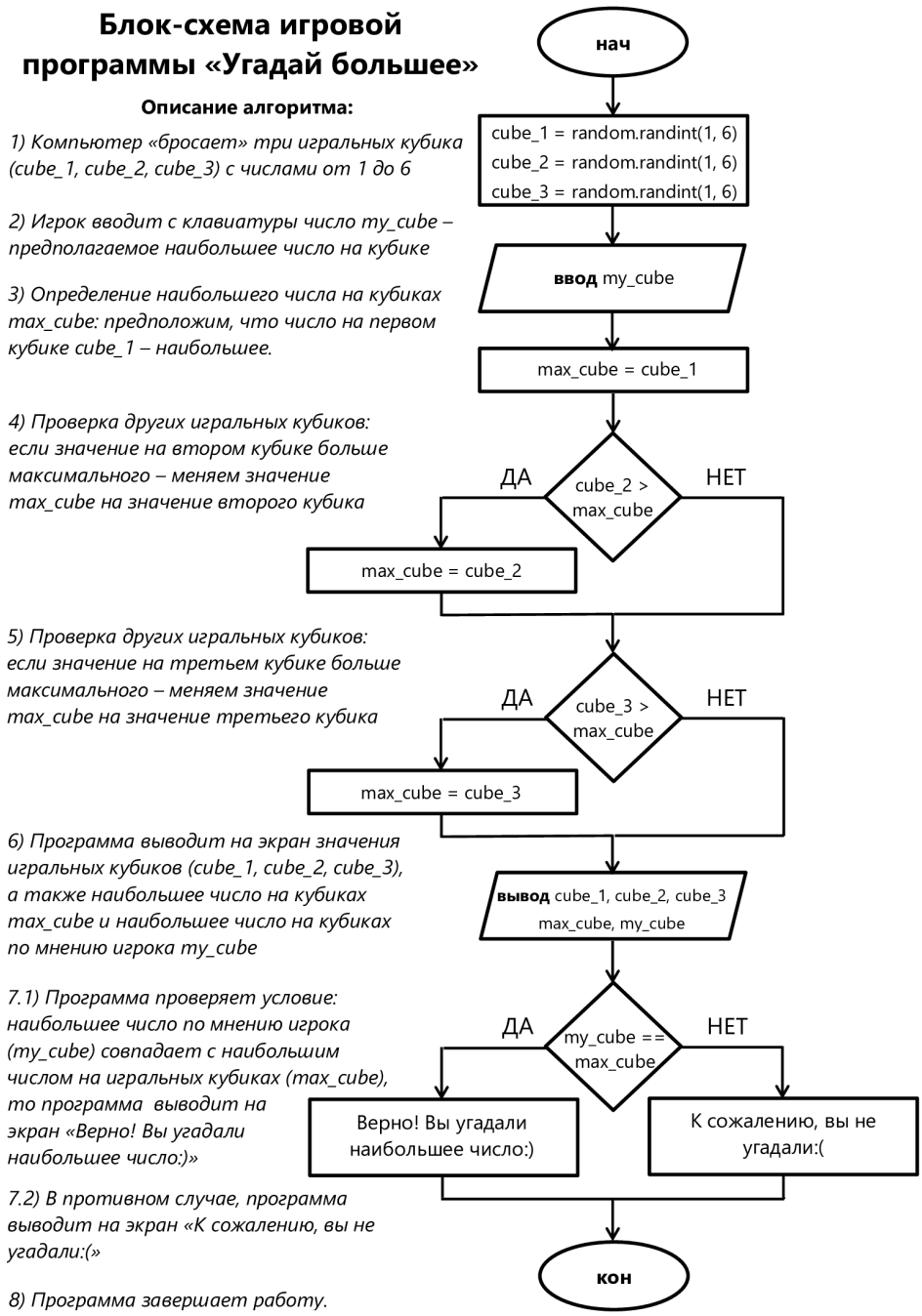

Рис. 7. Блок-схема игровой диалоговой программы «Угадай большее»

зоваться алгоритмы поиска максимума и минимума из трех чисел.

Демонстрация работы игровой диалоговой программы «Угадай большее» в онлайн-среде программирования Online Python Compiler (https://www.programiz.com/python-programming/online-compiler/) представлена на рисунке 8.

2.3. Игра в кости

Еще одной разновидностью игровых диалоговых программ, которые можно использовать в рамках обучения программированию, являются игры в кости.

В качестве примера такой игры рассмотрим игровую диалоговую программу «Кости», которая основана на одной из традиционных монгольских игр. «Шагай» — это традиционная монгольская игра в кости, при этом в Монголии насчитывается до 60 видов игры в «Шагай» [1]. В свою очередь, шагай — это специально обработанные кости нижних надкопытных суставов свиней, овец, коз, косуль и некоторых других животных [10].

Более подробно рассмотрим правила игры «Кости». В начале игры у каждого игрока есть по 10 косточек. В начале каждого раунда компьютер и игрок по очереди ставят от 1 до 5 косточек. Компьютер предполагает, сколько должно быть косточек у него в сумме с игроком, и называет эту сумму. Задача игрока состоит в том, чтобы определить, сколько будет косточек у него в сумме с компьютером. Тот, кто оказался ближе к истинной сумме, в конце раунда забирает косточки противника. Если и игрок, и компьютер угадали истинную сумму косточек

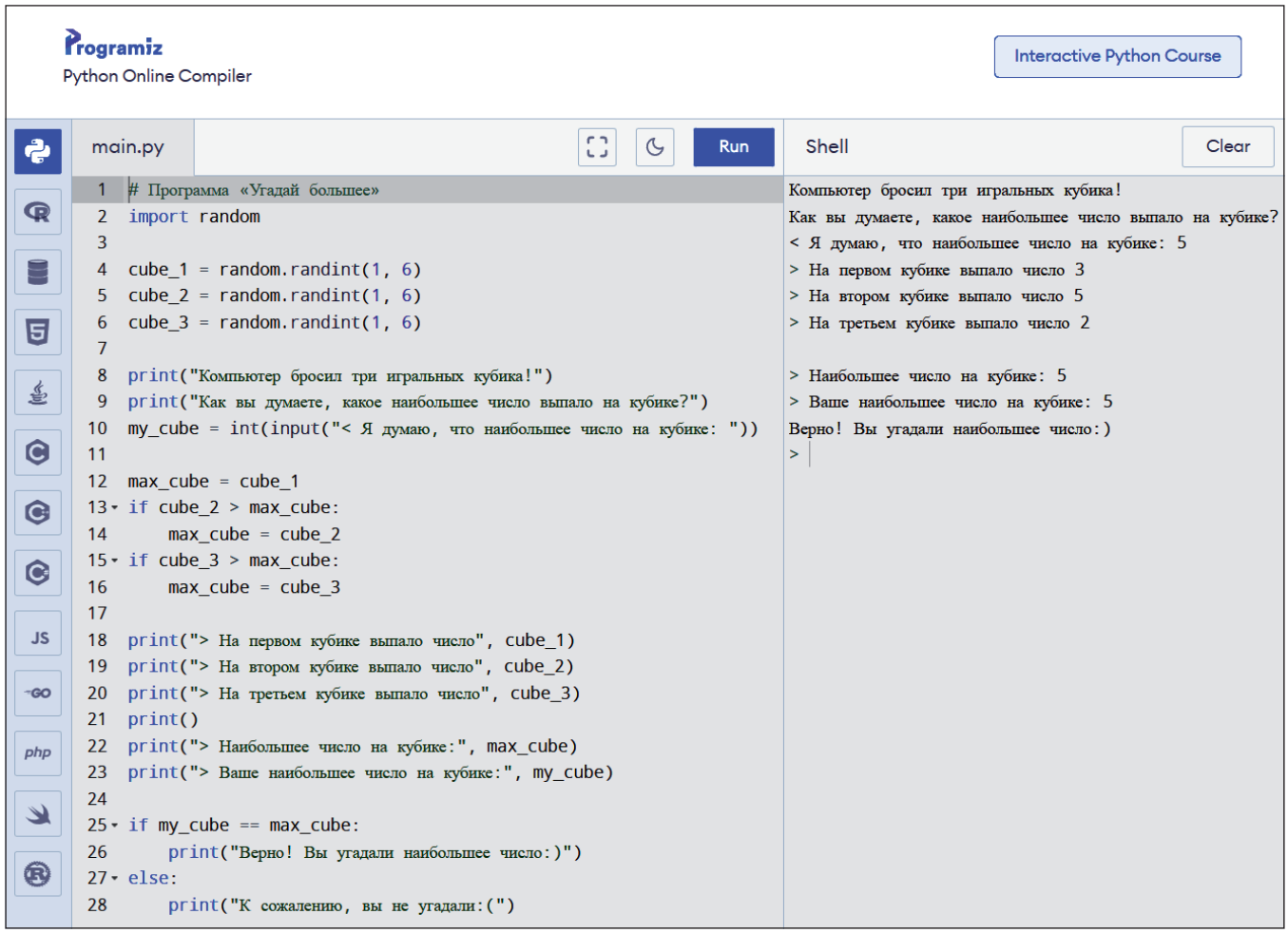

Рис. 8. Демонстрация работы игровой диалоговой программы «Угадай большее» в онлайн-среде программирования Online Python Compiler (Programiz)

или не угадали ее на одинаковое количество косточек, в раунде объявляется ничья. В игре побеждает тот, кто заберет все косточки противника.

Блок-схема игровой диалоговой программы «Кости» доступна по следующей ссылке: https://drive.google.com/file/d/1KTGDv1d4A1mghS2DauWOFAMctMyq6ick

Рассмотрим более подробно **алгоритм работы программы «Кости» на базе языка Python.**

1) В начале игры как у игрока, так и у компьютера есть по 10 косточек, или «шагай». Переменная *player_points* = 10 — количество косточек у игрока, *computer_points* = 10 — количество косточек у компьютера, а переменная *turns* = 0 — номер текущего раунда игры.

2) Игра разделена на раунды, каждый из которых представляет собой итерацию цикла с предусловием. Игровой цикл продолжается до тех пор, пока один из игроков не заберет все косточки у другого игрока, т. е. пока *player_points* > 0 и *computer_points* > 0. При этом в начале каждой итерации значение переменной *turns* (номер текущего раунда игры) увеличивается на 1, а на экран выводится количество косточек как у игрока, так и у противника.

3) Компьютер при помощи функции *random.randint*() «предполагает», сколько косточек может поставить игрок в текущем раунде, поэтому проверяется условие: является ли количество косточек игрока (*player_points*) больше 5. Если условие выполняется, то игрок может поставить от 1 до 5 косточек, а если условие не выполняется (у игрока осталось меньше 5 косточек), то игрок может поставить от 1 до *player_points* косточек.

4) Аналогичным образом при помощи функции *random.randint*() компьютер «предполагает», сколько косточек он может поставить в текущем раунде, поэтому проверяется условие: является ли количество косточек компьютера (*computer_points*) большим 5. Если условие выполняется, то компьютер может поставить от 1 до 5 косточек, а если условие не выполняется (у игрока осталось меньше 5 косточек), то компьютер может поставить от 1 до *computer_points* косточек.

Примечание: данные условия нужны для того, чтобы компьютер не мог поставить больше косточек, чем есть у игрока (например, если у игрока осталось 4 косточки, а компьютер ставит 5 косточек, чего не может быть), а также чтобы компьютер не мог поставить свои косточки «в долг», т. е. поставить больше косточек, чем у него есть (например, если у компьютера 1 косточка, а он ставит 4 косточки, если компьютер проиграет, то у него останется –3 косточки, чего также не может быть).

5) Вычисляется предполагаемая компьютером сумма косточек игрока и компьютера (*computer_sum*) и выводится на экран, чтобы помочь игроку приблизительно угадать количество косточек, которое мог поставить компьютер.

6) Игрок вводит с клавиатуры количество косточек, которые он хочет поставить (*pr_player*), а также предполагаемое количество косточек, которые поставил компьютер (*pr_computer*).

7) Вычисляется предполагаемая игроком сумма косточек игрока и компьютера (*player_sum*) и реальная сумма косточек игрока и компьютера (*true_sum*), т. е. сумма косточек, которые поставил игрок (*pr_player*), и косточек, которые поставил компьютер (*cr_computer*). Затем все полученные суммы косточек (*player_sum*, *computer_sum*, *true_sum*) выводятся на экран.

8) Сравниваются расстояния (модули разности) между реальной суммой косточек и предполагаемыми суммами косточек игрока и компьютера. При этом количество косточек победителя увеличивается на количество косточек, которые поставил проигравший, а проигравший теряет эти косточки. Вычисления производятся по следующим правилам:

- если предполагаемая сумма косточек игрока ближе к реальной сумме (ее модуль разности меньше), то побеждает игрок (на экран выводится сообщение «Я победил в раунде!»);
- если предполагаемая сумма косточек игрока ближе к реальной сумме (ее модуль разности меньше), то побеждает компьютер (на экран выводится сообщение «Я проиграл в раунде!»);
- в противном случае (модули разности между реальной суммой и суммами игрока и компьютера совпадают) в раунде объявляется ничья (на экран выводится сообщение «Ничья!»).

Например, реальная сумма косточек игрока и компьютера равна 8. Игрок предположил, что у него будет 5 косточек, а у компьютера — 2 косточки (сумма косточек игрока и компьютера равна 7). Компьютер же предположил, что у игрока будет 1 косточка, а у него — 3 косточки (сумма косточек игрока и компьютера равна 4). В данном случае игрок выигрывает, поскольку он оказался ближе к реальной сумме, чем компьютер, и поэтому забирает у него 3 косточки (поскольку компьютер «поставил» 3 косточки). А если бы игрок проиграл, то компьютер «забрал» бы у него 5 косточек (поскольку игрок «поставил» 5 косточек).

9) После выхода из цикла с предусловием игровой цикл заканчивается — на экран выводится количество косточек у игрока (*player_points*) и количество косточек у компьютера (*computer_points*).

10) Определяется победитель: к концу игры один игрок забирает все кости у другого игрока, поэтому:

- если количество косточек игрока (*player_points*) больше 0, то побеждает игрок (на экран выводится сообщение «Игрок победил!»);
- в противном случае (количество косточек у компьютера (*computer_points*) больше 0) побеждает компьютер (на экран выводится сообщение «Компьютер победил!»).

Примечание: чтобы игра была честной, игрок должен ставить от 1 до 5 косточек. Поэтому в качестве **дополнительного задания** можно предложить обучающимся доработать программу так, чтобы проверялось это условие и игрок не мог продолжить игру до тех пор,

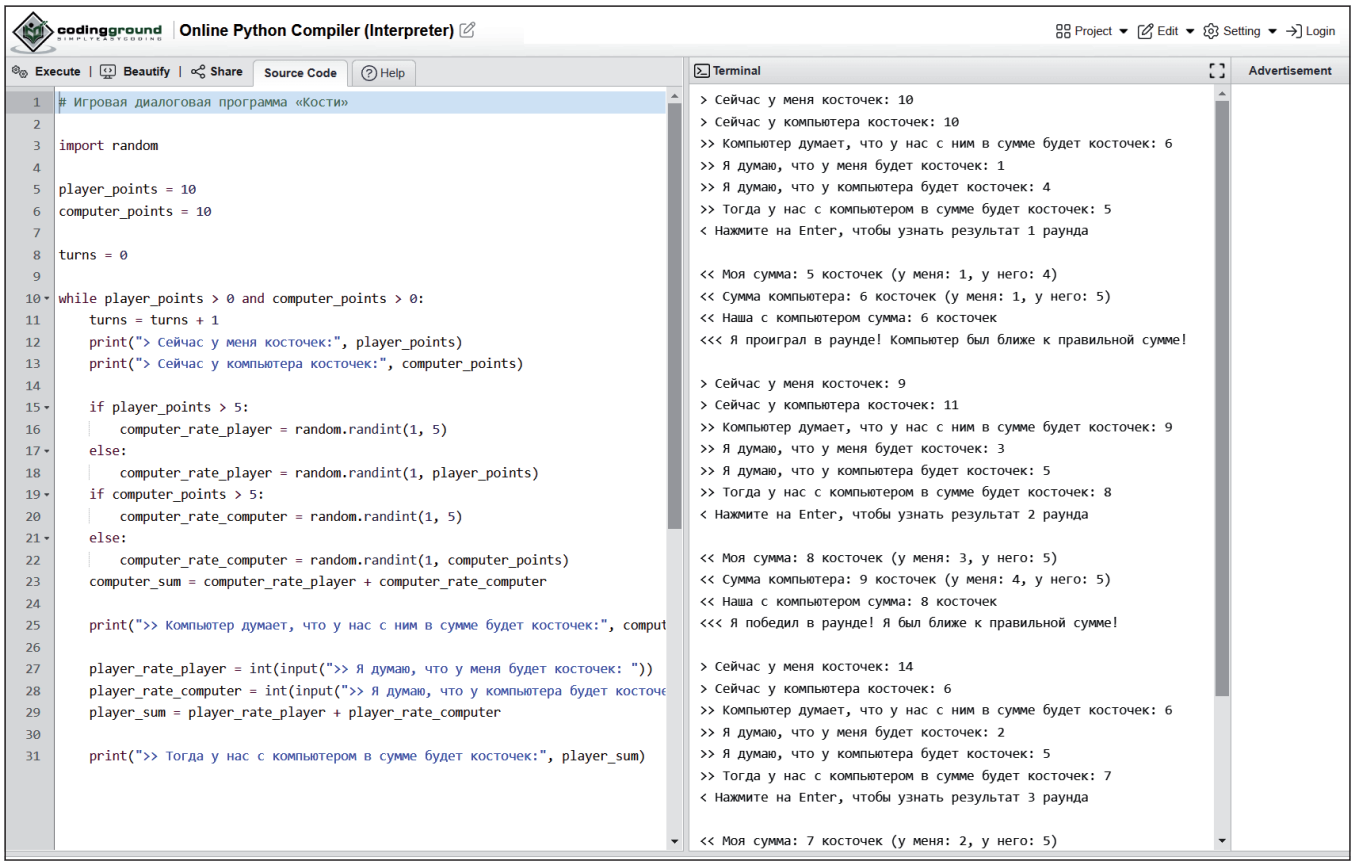

Рис. 9. Демонстрация работы игровой диалоговой программы «Кости» в среде программирования Online Python Compiler (TutorialsPoint)

пока не введет корректное количество косточек (от 1 до 5 или от 1 до *x*, если количество косточек игрока $x < 5$).

Данную программу можно использовать как в рамках изучения простых и сложных условий, а также вложенного и каскадного условного оператора в языке Python, так и в рамках изучения цикла с предусловием. Кроме того, для выполнения дополнительного задания необходимо использовать цикл с предусловием, который будет выполняться, пока значения переменных *pr_player* и *pr_computer* не лежат в диапазоне от 1 до 5.

Демонстрация работы игровой диалоговой программы «Кости» в среде программирования Online Python Compiler (https://www.tutorialspoint.com/execute_python3_online.php) представлена на рисунке 9. Также программа доступна в данной среде программирования по ссылке:

http://tpcg.io/_K13XHK

2.4. Быки и коровы

К игровым диалоговым программам, которые можно использовать при обучении решению типовых задач раздела «Программирование», также относится игровая программа «Быки и коровы».

Цель данной игры заключается в том, чтобы угадать трехзначное число, «загаданное» компьютером, при помощи подсказок, которые связаны с цифрами числа («коровы» и «быки»). У игрока есть 10 попыток, чтобы угадать это число, при этом компьютер может помочь игроку двумя следующими подсказками:

- «корова» — цифра числа угадана, но находится не на своем месте;
- «бык» — цифра числа угадана и находится на своем месте.

Блок-схема игровой диалоговой программы «Быки и коровы» доступна по ссылке: https://drive.google.com/file/d/1tTmE3IofMj86jRpx-tsapTpkAh5_Oypw

Рассмотрим **алгоритм работы данной программы** более подробно.

1) Программа генерирует цифры случайного трехзначного числа, причем первая цифра может принимать значения от 1 до 9 (если первая цифра будет нулем, то получится двузначное число), а вторая и третья цифры могут принимать значения от 0 до 9.

2) Необходимо убедиться в том, что цифры сгенерированного случайного числа не повторяются. Для этого создаются два цикла с предусловием:

- в первом цикле заново генерируется вторая цифра числа (*secret*2) до тех пор, пока она совпадает с первой цифрой числа (*secret*1);
- во втором цикле заново генерируется третья цифра числа (*secret*3) до тех пор. пока она совпадает с первой (*secret*1) или второй (*secret*2) цифрой числа.

3) Вычисляется случайное трехзначное число *secret* из цифр этого числа (*secret*1, *secret*2, *secret*3) по формуле *secret* = 100 * *secret*1 + 10 * *secret*2 + *secret*3, где *secret* — это случайное трехзначное число, а *secret*1, *secret*2, *secret*3 — его цифры.

4) В начале игрового цикла количество попыток угадывания числа увеличивается на 1, а затем определяются переменные *cows* = 0 — количество «коров» в предполагаемом числе и *bulls* = 0 — количество «быков» в предполагаемом числе.

5) Начинается игровой цикл, каждая итерация которого представляет собой попытку угадывания числа, загаданного компьютером. Поэтому в начале цикла количество попыток угадывания числа (*attempt*) увеличивается на 1, а затем определяются переменные *cows* = 0 — количество «коров» в предполагаемом числе и *bulls* = 0 — количество «быков» в предполагаемом числе.

6) Игрок вводит с клавиатуры предполагаемое число *number*, которое загадал компьютер.

7) При помощи арифметических операций «деление нацело» и «деление по остатку» выделяются цифры предполагаемого числа *number*, которое загадал компьютер (*number*1, *number*2, *number*3).

8) Подсчитывается количество «быков» и «коров» для предполагаемого числа *number*, которое загадал компьютер:

- если первая цифра числа, загаданного компьютером (*secret*1), совпадает с первой цифрой числа, предполагаемого игроком (*number*1), то количество «быков» (*bulls*) увеличивается на 1; в противном случае, если первая цифра числа, загаданного компьютером (*secret*1), совпадает со второй или третьей цифрой числа, предполагаемого игроком (*number*2 или *number*3), то количество «коров» (*cows*) увеличивается на 1;
- если вторая цифра числа, загаданного компьютером (*secret*2), совпадает со второй цифрой числа, предполагаемого игроком (*number*2), то количество «быков» (*bulls*) увеличивается на 1; в противном случае, если вторая цифра числа, загаданного компьютером (*secret*2), совпадает с первой или третьей цифрой числа, предполагаемого игроком (*number*1 или *number*3), то количество «коров» (*cows*) увеличивается на 1;
- если третья цифра числа, загаданного компьютером (*secret*3), совпадает с третьей цифрой числа, предполагаемого игроком (*number*3), то количество «быков» (*bulls*) увеличивается на 1; в противном случае, если третья цифра числа, загаданного компьютером (*secret*3), совпадает с первой или второй цифрой числа, предполагаемого игроком (*number*1 или *number*2), то количество «коров» (*cows*) увеличивается на 1.

9) Если число, загаданное компьютером (*secret*), совпадает с числом, загаданным игроком (*number*), на экран выводится сообщение: «Поздравляем! Вы угадали число!» и цикл досрочно завершается, иначе на экран выводится количество «коров» и «быков».

10) После окончания игрового цикла проверяется условие: различается ли число, загаданное компьютером (*secret*), с последним числом, предполагаемым игроком (*number*). Если условие выполняется, программа выводит на экран сообщение: «К сожалению, вы не угадали, компьютер загадал число *secret*», где *secret* — число, загаданное компьютером.

Демонстрация работы игровой диалоговой программы «Быки и коровы» в среде программирования Repl.it (https://repl.it/) представлена на рисунке 10.

Данную игровую программу можно использовать в рамках изучения алгоритма выделения цифр из натурального числа, который является одной из типовых задач раздела «Программирование», а также при изучении

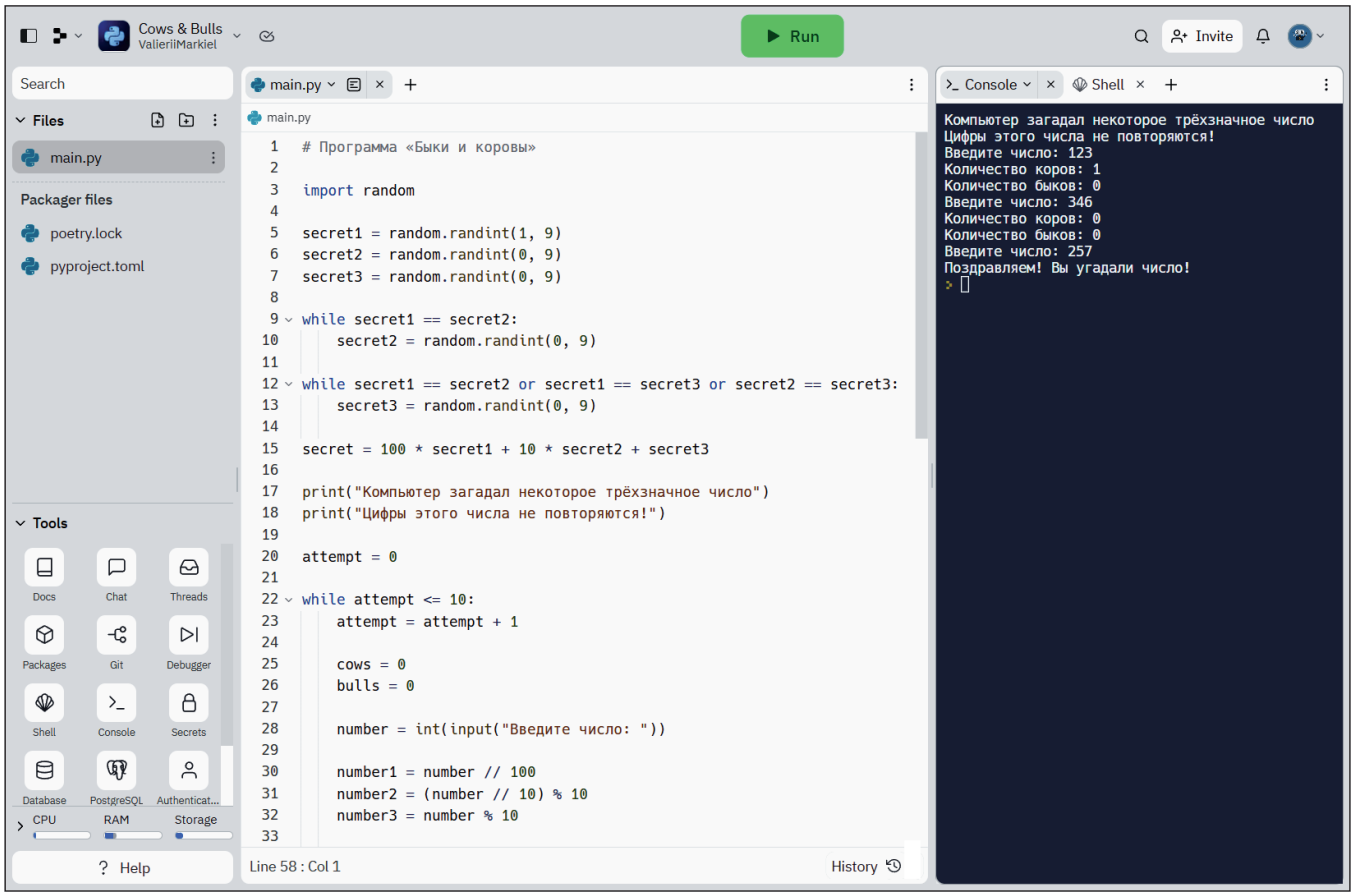

Рис. 10. Демонстрация работы игровой диалоговой программы «Быки и коровы» в среде программирования Repl.it

цикла с предусловием, полного и каскадного условного оператора. При этом в качестве дополнительного задания можно предложить обучающимся написать аналогичную программу для четырехзначного числа, сняв ограничение на количество попыток угадывания числа.

3. Заключение

Возможности использования рассмотренных игровых диалоговых программ при обучении решению типовых задач раздела «Программирование» можно обобщить при помощи таблицы.

Таким образом, различные разновидности игровых диалоговых программ могут использоваться для обучения решению типовых задач раздела «Программирование». В частности, примеры игровых диалоговых программ, которые были рассмотрены в статье, могут использоваться в рамках изучения алгоритмов обработки данных с использованием ветвлений и циклов, алгоритмов поиска максимума и минимума, алгоритмов

Таблица

Возможности использования игровых диалоговых программ при обучении решению типовых задач раздела «Программирование»

№ п/п	Игровые диалоговые программы	Типовые задачи раздела «Программирование» в соответствии с ФГОС основного общего образования
1	Прогулка	Алгоритм обработки данных с использованием ветвлений
2	Таинственный остров	Алгоритм обработки данных с использованием ветвлений
3	Бросок игральных кубиков	Алгоритм обработки данных с использованием ветвлений
4	Угадай большее	Алгоритмы поиска максимума и минимума
5	Кости	Алгоритм обработки данных с использованием ветвлений. Алгоритм обработки данных с использованием циклов
6	Быки и коровы	Алгоритм обработки данных с использованием ветвлений. Алгоритм обработки данных с использованием циклов. Алгоритм выделения цифр из натурального числа

выделения цифр из натурального числа. При этом, несмотря на то что данные игровые диалоговые программы написаны на языке Python, при помощи соответствующих им блок-схем эти программы могут быть адаптированы для обучения решению типовых задач раздела «Программирование» на базе других учебных языков программирования (Pascal, Java, C++).

Список источников

1. *Бизьяагийн С.* Педагогический потенциал монгольских народных игр «Шагай»: дис. ... канд. пед. наук. Улан-Удэ, 1998. 23 с.

2. *Жемчужников Д. Г.* Формирование мотивации школьников к изучению программирования на основе межпредметной интеграции // Вестник Российского университета дружбы народов. Серия: Информатизация образования. 2011. № 1. С. 46–48. EDN: NDAELX.

3. *Завьялова О. А., Маркелов В. К.* Преподавание программирования в школе как барьер в профессиональном выборе будущего учителя информатики // Научный поиск: личность, образование, культура. 2022. № 2 (44). С. 31–38. EDN: CJGDQK. DOI: 10.54348/SciS.2022.2.5

4. *Латанская И. В., Громова С. Ф.* Развитие содержательной линии «Алгоритмизация и программирование» в школьном курсе информатики // XXIII Всероссийская студенческая научно-практическая конференция Нижневартовского государственного университета. Материалы конференции (Нижневартовск, 06–07 апреля 2021 года). Ч. 10. Нижневартовск: Нижневартовский государственный университет, 2021. С. 361–368. EDN: DJDAAY.

5. *Малова А. И., Куликова Н. Ю.* Использование визуальных сред разработки компьютерных игр при обучении алгоритмизации и программированию // Образование и проблемы развития общества. Сборник научных статей Международной научно-методической конференции (Курск, 03 октября 2019 года). Курск: Юго-Западный государственный университет, 2019. С. 18–21. EDN: YCCRYC.

6. *Маркелов В. К., Завьялова О. А.* Игровые диалоговые программы как инструмент мотивации к изучению программирования в школе // Современные тренды образования. Материалы IV Всероссийской (национальной) педагогической научно-практической конференции (Шуя, 15–17 декабря 2021 года). Шуя: Ивановский государственный университет, Шуйский филиал, 2022. С. 81–89. EDN: HLVJLL.

7. Приказ Минпросвещения России от 31.05.2021 № 287 «Об утверждении федерального государственного образовательного стандарта основного общего образования». https://www.consultant.ru/document/cons_doc_LAW_389560/

8. Примерная рабочая программа основного общего образования. Информатика. Базовый уровень (для 7–9 классов образовательных организаций) (Одобрена решением федерального учебно-методического объединения по общему образованию, протокол 3/21 от 27.09.2021 г.). https://fgosreestr.ru/uploads/files/dcca994c21165f0d49d4baf4a7e008c0.pdf

9. *Чебурина О. В.* Формирование алгоритмического мышления в обучении программированию игр // Наука и перспективы. 2017. № 2. С. 75–79. EDN: ZDUTXJ.

10. *Шохирев В. В.* Народные игры бурят-монгольского населения Сибири // Совершенствование профессиональной и физической подготовки курсантов, слушателей образовательных организаций и сотрудников силовых ведомств. Сборник статей XXII Всероссийской научно-практической конференции (Иркутск, 01 октября 2020 года). Иркутск: Восточно-Сибирский институт Министерства внутренних дел Российской Федерации, 2020. С. 464–466. EDN: PPKJKF.