УДК 004.8
DOI: http://dx.doi.org/10.21686/1818-4243-2025-6-4-19

К. В. Розов[1], А. В. Подсадников[2], Н. А. Чупин[2]

[1] МБОУ Гимназия № 4, Новосибирск, Россия
[2] Новосибирский государственный педагогический университет, Новосибирск, Россия

Сюжетные задачи с консольной визуализацией в курсе программирования на языке C# как способ активизации творческой самостоятельной работы обучающихся

*Статья посвящена **актуальной проблеме** активизации самостоятельной познавательной деятельности будущих учителей информатики в рамках их профессиональной подготовки в педагогическом вузе. В качестве инструмента для достижения этой цели авторы предлагают использование специально разработанных сюжетных задач с консольной визуализацией, решение которых осуществляется на языке программирования C#.*

***Цель исследования** – рассмотреть особенности использования языка программирования C# в контексте подготовки будущих учителей информатики, выявить преимущества и недостатки данного выбора, а также предложить подход к повышению эффективности образовательного процесса через внедрение сюжетных практико-ориентированных задач с консольным графическим выводом, связанных с моделированием игровых ситуаций и разработкой архитектуры приложений. Такие задачи предназначены прежде всего для их решения наиболее успевающими студентами, способными к программированию обучающимися. Особое внимание уделено вопросам мотивации студентов и развития их творческих способностей.*

***Методология исследования** включает анализ существующих подходов к обучению программированию и тенденций к выбору языка программирования для обучения в школе и педагогическом вузе; выявление возможностей языка C#, его места в профессиональной подготовке педагога и элементов содержания обучения программированию на этом языке; разработку учебных заданий и проведение педагогического эксперимента. Внедрение сюжетных задач с консольной визуализацией направлено не только на повышение познавательной мотивации, но и на комплексное развитие навыков программирования на современном высокоуровневом языке. Важным аспектом является то, что студенты получают опыт работы как в процедурной, так и в объектно-ориентированной парадигмах, что способствует формированию гибкого и глубокого понимания принципов разработки программного обеспечения. Это, в свою очередь, повышает конкурентоспособность будущих учителей на рынке труда, знакомя их с инструментом, востребованным в реальной IT-индустрии.*

***Новизна работы** состоит, во-первых, в целенаправленном рассмотрении языка C#, который, будучи востребованным в профессиональной среде, остается на периферии методических исследований в области преподавания информатики. Во-вторых, в представленных авторами оригинальных методических разработках – системе сюжетных задач, адаптированных для C# с акцентом на консольную визуализацию. Этот подход позволяет обойти ограничения, связанные с дефицитом аудиторного времени, и сфокусироваться на развитии алгоритмического и системного мышления. Разработанный комплекс задач опирается на фундаментальные педагогические принципы, включая принцип постепенного усложнения (Л.С. Выготский), принцип контекстного обучения (А.А. Вербицкий) и принцип продуктивных ограничений (С. Пейперт), что обеспечивает ее методологическую обоснованность.*

***Апробация** предложенного подхода, проведенная в Новосибирском государственном педагогическом университете, подтвердила его эффективность для организации творческой самостоятельной работы обучающихся. Несмотря на то, что задачи выбирало небольшое количество студентов, именно эта целевая группа продемонстрировала значительный прогресс в профессиональном развитии. Наиболее успешные участники не только решили все предложенные задачи, но и в дальнейшем применяли полученный опыт в рамках педагогической практики и курсового проектирования. Таким образом, разработанный комплекс заданий доказал свою практическую значимость и может быть рекомендован для использования в подготовке будущих учителей информатики, а также для работы с мотивированными обучающимися школ, проявляющими повышенный интерес к программированию.*

***Ключевые слова:** программирование, язык программирования, C#, занимательные задачи, сюжетные задачи, практико-ориентированные задачи, самостоятельная работа, геймификация обучения, творческое мышление.*

Konstantin V. Rozov[1], Alexey V. Podsadnikov[2], Nikolay A. Chupin[2]

[1] MBOU Gymnasium No. 4, Novosibirsk, Russia
[2] Novosibirsk State Pedagogical University, Novosibirsk, Russia

Story-Based Tasks with Console Visualization in the C# Programming Course as a Way to Activate Creative Independent Work of Students

The article addresses the relevant problem of activating independent cognitive activity of future teachers of computer science within their professional training at a pedagogical university. As a tool to achieve this purpose, the authors propose the use of specially designed story-based tasks with console visualization, solved using the C# programming language.

*The **purpose of the study** is to examine the features of using the C# programming language in the context of training future teachers of computer science, to identify the advantages and disadvantages of this choice, and to propose an approach to enhancing the effectiveness of the educational process through the introduction of story-based, practice-oriented tasks with console graphical output. These tasks*

are related to modeling game situations and developing application architecture. Such tasks are primarily intended to be solved by the most successful students and capable of programming. Particular attention is paid to the issues of student motivation and the development of their creative abilities.

*The **research methodology** includes an analysis of existing approaches to teaching programming and trends in the choice of programming languages for teaching in schools and pedagogical universities; identifying the capabilities of the C# language, its place in the professional training of teachers, and elements of the content of teaching programming in this language; the development of educational tasks; and conducting a pedagogical experiment. The introduction of story-based tasks with console visualization is aimed not only at increasing cognitive motivation but also at the comprehensive development of programming skills in a modern high-level language. An important aspect is that students gain experience working in both procedural and object-oriented paradigms, which contributes to the formation of a flexible and deep understanding of software development principles. This, in turn, increases the competitiveness of future teachers in the labor market by introducing them to a tool in demand in the real IT industry.*

*The **novelty of the paper** consists, firstly, in a focused consideration of the C# language, which, being in demand in the professional environment, remains on the periphery of methodological research in the field of computer science teaching. Secondly, it lies in the authors' original methodological developments – a system of story-based tasks adapted for C# with an emphasis on console visualization. This **approach** allows for overcoming limitations associated with the shortage of classroom time and focusing on the development of algorithmic and systems thinking. The developed set of tasks is based on fundamental pedagogical principles, including the principle of gradual complication (L. Vygotsky), the principle of contextual learning (A. Verbitsky), and the principle of productive constraints (S. Papert), which ensures its methodological validity.*

The approbation of the proposed approach, conducted at Novosibirsk State Pedagogical University, confirmed its effectiveness in organizing creative independent work of students. Although only a small number of students chose these tasks, this target group demonstrated significant progress in professional development. The most successful participants not only solved all the proposed tasks but also subsequently applied the acquired experience in teaching practice and course projects. Thus, the developed set of tasks has proven its practical significance and can be recommended for the use in the training of future computer science teachers, as well as for working with motivated school students who show an increased interest in programming.

Keywords: programming, programming language, C#, entertaining tasks, story-based tasks, practice-oriented tasks, independent work, gamification of learning, creative thinking.

Введение

Современное педагогическое образование в сфере информатики и информационных технологий сталкивается с рядом вызовов, связанных с необходимостью повышения качества подготовки будущих специалистов. Одним из ключевых аспектов успешной профессиональной подготовки будущих учителей является развитие у них творческих и аналитических способностей. В этой связи особое внимание уделяется выбору подходящих инструментов и методик, способных обеспечить эффективное усвоение материала и стимулировать самостоятельную работу студентов.

В качестве одного из инструментов развития творческих и аналитических способностей, мотивирующих студентов на решение комплексных задач по программированию, предполагающих самостоятельный поиск информации и изучение тем, выходящих за пределы базового курса программирования, являются сюжетные практико-ориентированные задачи с консольной визуализацией, предлагаемые авторами наиболее успевающим студентам в курсе программирования на языке C#. Эти задачи направлены не только на закрепление полученных знаний и формирование устойчивых навыков в области алгоритмизации и программирования, но и на развитие важных качеств личности, таких как критическое мышление, креативность и способность к решению комплексных проблем.

Актуальность предложенного подхода к активизации творческой самостоятельной работы обучающихся по программированию подтверждается исследованиями в области преподавания программирования, в которых подчеркивается роль визуализации, геймификации [1–5] и практико-ориентированности [4, 6] в повышении мотивации. Анализ публикаций свидетельствует, что подавляющее большинство методик обучения основам программирования на универсальном языке программирования ориентировано на применение интерпретируемого языка Python [1, 4, 7–13], что обусловлено прежде всего его низким порогом входа, упрощением внешнего вида программ. Отметим, что сюжетные, практико-ориентированные задачи, отраженные в публикациях [1, 4], предполагают консольный ввод и вывод без использования графики, в т.ч. символьной.

Применение C# в образовательном контексте остается малоизученным, несмотря на его востребованность в профессиональной среде. Однако профессиональная подготовка будущих педагогов требует знакомства с промышленными языками, что подтверждается исследованиями по формированию структуры и содержания цифровых компетенций, формируемых в результате освоения углубленного курса информатики профильными классами [14, 15].

Новизна работы заключается в адаптации сюжетных задач для C# с акцентом на консольную визуализацию, что позволяет сочетать развитие алгоритмического, логического и системного мышления с освоением профессионального инструментария в условиях небольшого количества аудиторных часов, выделяемых на практическую работу студентов, когда на решение задач в области разработки UI (User Interface) средствами Windows Forms или WPF (Windows Presentation Foundation) не остается времени. Предлагаемый подход сознательно использует C#, сочетая преимущества:

- профессиональной релевантности (статическая типизация, ООП);
- образовательной эффективности (консольная визуализация как баланс между наглядностью и сложностью).

О языке C#

C# – объектно-ориентированный язык программирования со статической типизацией. Разработан в период с 1998 по 2001 год программистами компании Microsoft. Язык программирования C# относится к семейству Си-подобных языков, соответственно, его синтаксис близок к синтаксису языков C, C++ и Java. Примечательно, что одним из ведущих разработчиков C# является Андерс Хейльсберг (Anders Hejlsberg) – разработчик языков программирования Turbo Pascal, Object Pascal и среды Delphi, в связи с чем C# унаследовал множество полезных концепций от Delphi [16].

Программы, написанные на C#, выполняются в .NET – виртуальной системе выполнения, вызывающей общеязыковую среду выполнения (CLR) и набор библиотек классов. CLR упрощает разработку приложений и их компонентов, обеспечивая возможность взаимодействия объектов, написанных на разных языках (Visual Basic, C++ и др.), с помощью общеязыковой структуры языка (CLI). Исходный код на языке C# компилируется в промежуточный язык, соответствующий CLI. При выполнении C#-программы сборка (код на промежуточном языке и ресурсы) загружаются в среду CLR, которая выполняет динамическую компиляцию (JIT-компиляцию) промежуточного кода в машинный код для конкретной платформы. Среда CLR выполняет и другие операции, такие как управление ресурсами, автоматическая сборка мусора и обработка исключений.

Язык программирования C# используется для написания приложений различных направлений: настольные приложения для операционной системы Microsoft Windows, серверное программное обеспечение для веб-ресурсов, игры для настольных компьютеров и мобильных устройств и др.

На языке программирования C# можно создавать консольные приложения (приложения без визуального интерфейса) как без применения, так и с применением принципов объектно-ориентированного программирования. Также можно создавать визуальные приложения двух видов: классические оконные визуальные приложения Windows Forms и WPF приложения. Технология WPF предоставляет богатый набор инструментов для обогащения визуализации приложения, в частности, позволяет создавать компоненты нестандартной формы, что отличает приложение WPF от классических визуальных приложений. Для разработки визуальных приложений активно используется IDE Visual Studio, разработанная компанией Microsoft.

Место языка программирования C# в профессиональной подготовке будущего учителя информатики

При обучении программированию в педагогическом университете перед преподавателями встает вопрос: какие языки программирования выбрать для обучения? В рамках подготовки будущих учителей информатики при выборе языка следует ориентироваться на то, какие языки программирования используют в школах. В качестве первого и основного языка программирования в школах все чаще используют Python как на базовом, так и на углубленном уровне курса информатики [1, 7–12]. Язык фактически стал новым стандартом, заменив популярный ранее Pascal. Поэтому, несмотря на неоднозначность использования Python как основного языка программирования для обучения [13], его изучение в современном педагогическом вузе фактически обязательно при подготовке будущих учителей информатики. Но на профильном, углубленном уровне изучения информатики рекомендуется владение обучающимися несколькими языками программирования [14, 17]. Таким образом, в профессиональную подготовку будущего учителя информатики должно быть включено изучение как минимум еще одного языка, кроме наиболее популярного для преподавания в рамках школьного курса информатики.

В Новосибирском государственном педагогическом университете в рамках дисциплины «Программирование» студенты профилей, связанных с информатикой, изучают основы программирования на двух языках в двух разных семестрах. Курс ориентирован на бакалавров направлений подготовки 44.03.01 «Педагогическое образование» и 44.03.05 «Педагогическое образование (с двумя профилями подготовки)», профилей: информатика и ИКТ; математика и информатика; физика и информатика; информатика и иностранный (английский) язык. В качестве первого языка изучается Python, а в качестве второго языка программирования авторами было принято решение использовать язык C#. На выбор именно этого языка программирования повлияли следующие его преимущества как языка для обучения:

- актуальность языка в профессиональной среде и его наличие в перечне языков программирования, которые могут изучаться в школьном курсе информатики согласно федеральным рабочим программам по информатике на уровнях ООО и СОО, и использоваться для решения задач на ОГЭ и ЕГЭ по информатике;
- Си-подобный синтаксис, освоение которого упростит переход на другие востребованные в профессиональной

и академической средах языки со схожим синтаксисом (C, C++, Java, JavaScript и др.) при необходимости;

• объектно-ориентированность и ориентированность на компоненты (в т.ч. визуальные), которая позволит в рамках других дисциплин изучать непосредственно объектно-ориентированное программирование на этом языке с опорой на опыт процедурного программирования и технологий разработки приложений с графическим интерфейсом;

• явная статическая типизация, требующая от обучающихся более внимательного отношения к типам данных, чем при программировании на Python, что способствует дисциплине обучающегося при написании кода и лучшему понимаю им особенностей хранения данных в компьютере;

• богатые возможности работы с консолью по сравнению с Python: перемещение курсора по координатам, изменение цвета фона и символов, считывание нажатий клавиш и др.;

• схожесть консольного ввода с языком Python, заключающаяся в считывании строки и необходимости её преобразования в другие типы данных во многих учебных задачах, что на первых этапах изучения языка позволяет преподавателю проводить аналогии с опорой на опыт программирования обучающихся на Python с целью упрощения понимания ими особенностей преобразования типов.

Кроме того, основы другого востребованного языка программирования в школьной среде – C++, эффективного, в частности, для решения олимпиадных задач, изучаются студентами, будущими учителями информатики, позже в рамках других дисциплин, таких как «Объектно-ориентированное программирование» и «Программирование цифровой микроэлектроники». Общая схожесть синтаксиса и основных операторов для базовых алгоритмических конструкций в языках C++ и C# позволяет преподавателю быстрее включить студентов в практическую работу на этих дисциплинах.

К недостаткам языка C# для обучения можно отнести:

• непопулярность в образовательной среде. Несмотря на наличие языка в перечне языков программирования для обучения алгоритмизации и программированию в школе, сложно найти научно-педагогические публикации и учебно-методические разработки, связанные с применением этого языка для обучения программированию обучающихся школ или студентов педагогических вузов. Наиболее популярными в школьном и педагогическом образовании «профессиональными» языками являются Python (основы программирования, олимпиадное программирование, проектная деятельность – создание игр, искусственный интеллект и др.) и C++ (олимпиадное программирование, программирование плат Arduino, ESP) в связи с их большей универсальностью по отношению к деятельности обучающихся (применение более очевидно и разнообразно с учетом разных возрастов начинающих программистов) [8, 12];

• высокая объектно-ориентированность, что создает методическую сложность обучения программированию на этом языке начинающих, которым может быть сложно сразу понять концепции классов и объектов. Это приводит к тому, что о многих вещах, которые обучающийся постоянно наблюдает в коде, при обучении приходится либо долгое время умалчивать, либо обучать языку только тех, кто имеет уже хорошую «базу», опыт процедурного программирования на другом высокоуровневом языке и готов осваивать объектно-ориентированное программирование.

Таким образом, язык Python в рамках дисциплины «Программирование» выступает базой как с точки зрения основ программирования (как показывает практика, многие студенты программируют достаточно «слабо» в начале изучения дисциплины, что приближает их к обучающимся в школе, которые изучают программирование впервые, а именно «легкость» освоения Python отмечается многими педагогами и исследователями [9, 11], в том числе и авторами, как одно из его ключевых преимуществ перед другими языками), так и с точки зрения инструментария будущего учителя (базовое знание Python для современного учителя информатики обязательно). Язык C# в свою очередь выступает как средство закрепления навыков разработки алгоритмов и программ в новых условиях; как средство расширения представления обучающихся о языках программирования (если Python – интерпретируемый язык с неявной динамической типизацией, то C# – его противоположность: компилируемый язык с явной статической типизацией) и концепциях программирования (если на Python можно было писать код исключительно в процедурном стиле, не затрагивая объектно-ориентированное программирование и не уточняя понятие «метод» при изучении списков, словарей и др., то для программирования на C# понимание сущностей «класс», «метод» уже необходимо как минимум на уровне пользователя, даже без написания своих классов, и это хорошая «естественная» возможность подвести студентов к теме объектно-ориентированного программирования – сам язык является отражением этой концепции).

Содержание обучения программированию на языке C#

Курс изучения программирования на языке C# в рамках дисциплины «Программирование» представляет собой последовательность взаимосвязанных разделов. Каждый раздел состоит из теоретической части и набора практических задач. Несмотря на то, что C# является объектно-ориентированным языком, в котором даже в базовой программе присутствует класс, практические задачи ориентированы на процедурное программирование при написании консольных приложений. В условиях крайне ограниченного количества часов, выделяемых на аудиторную практическую работу в семестре (26 академических часов) и с учетом наличия опыта программирования на языке Python у обучающихся это позволяет показать как сходства, так и различия современных императивных языков программирования, главным образом различия в динамической/статической и неявной/явной типизации, а также показать, что учебные задачи по программированию (в т.ч. школьного уровня) могут быть решены на разных императивных языках, а потому важно не только знать особенности того или иного языка программирования, но и фундаментальные (теоретические) основы алгоритмизации и программирования в принципе (не так важен язык, как умение программировать). Разработка консольных приложений на языке C# без прямого использования объектно-ориентированного программирования на начальном этапе обучения программированию реализуется и при подготовке студентов непедагогических специальностей [18].

Представим разделы учебного курса в порядке их изучения и рассмотрим их основные дидактические единицы.

Первый раздел «*Синтаксис языка программирования C#, структура программы, типы данных, переменные, ввод данных с клавиатуры и вывод данных на консоль*» посвящен изучению отличительных синтаксических особенностей языка программирования C#. В связи с идеологией языка программирования C#, построенной на том, что каждая программа, написанная на нём, является классом, особое внимание уделяется структуре программы. Поскольку C# – это язык программирования со статической типизацией, на первых шагах подробно рассматриваются типы данных и различные способы приведения и конвертирования данных из одного типа в другой. При изучении консольного ввода и вывода данных делается акцент на понимание подключения и использования пространств имён в программе.

Второй раздел «*Работа с числами в C#, арифметические операции, математические функции, случайные числа*» посвящен изучению операций над числовыми данными. В силу особенности работы операций деления в языке C# акцентируется внимание на результатах применения операций как к целым, так и к вещественным числам. Здесь же рассматривается класс *Math*, в котором описываются основные функции для математических вычислений. Также для облегчения дальнейшего тестирования создаваемых программ на данном этапе изучаются функции генерации случайных чисел – класс *Random*. Поскольку при работе со случайными числами обучающиеся сталкиваются с созданием экземпляра класса здесь же рассматривается оператор *new*, который повсеместно используется во время работы с классами в C#.

Раздел «*Логический тип данных. Алгоритм ветвления в C#. Условный оператор. Оператор выбора. Оператор проверки*». При изучении алгоритма ветвления на различных языках программирования как правило в первую очередь решаются задачи на операции отношения с числами. В этом разделе также сделан упор на решение числовых задач. Вначале рассматриваются синтаксические особенности построения логических выражений, затем рассматриваются логические операции для построения сложных выражений. После подробного изучения принципов построения условных выражений изучаются непосредственно условные операторы. Рассматриваются структуры построения алгоритма ветвления: неполное и полное. Далее изучается тернарный условный оператор «? :» и оператор множественного выбора *switch-case*. В завершение раздела рассматриваются различные вложенные конструкции алгоритма ветвления.

Раздел «*Циклические алгоритмы в C#*». В языке C#, как и в других Си-подобных языках, оператор цикла *for* построен таким образом, что он по своей сути уже не является классическим циклом со счётчиком, а является некой универсальной алгоритмической структурой, позволяющей решать все возможные задачи на применение цикла, т.е. является неким универсальным инструментом. Поэтому сначала рассматривается оператор цикла *for*, а затем цикл с предусловием *while*. Далее рассматривается цикл с постусловием do..while. С помощью *for* рекомендуется решать именно задачи на использование классического цикла со счётчиком, а задачи с неизвестным числом повторений – с помощью *while* или *do..while*. Здесь же рассматриваются различные вложенные конструкции: цикл вложен в цикл, ветвление вложено в цикл, цикл вложен в ветвление. Также уделяется внимание операторам *break* и *continue*.

Раздел *«Массивы в C#»*. После изучения циклических операторов можно приступать к работе с различными структурами данных. Именно работа со структурами данных позволяет в большей мере отработать навыки работы с циклами. В этом блоке мы рассматриваем работу с классическими массивами. Рассматриваются одномерные массивы, двумерные массивы, а также массивы массивов – так называемые «ступенчатые» массивы. При работе с массивами решаются задачи заполнения массивов различными способами, обработки, поиска и сортировки элементов массива. Обучающимся запрещено использовать готовые методы массивов (хотя методы рассматриваются на лекциях), они пишут алгоритмы вручную, что способствует развитию у них навыков алгоритмизации. Работа с многомерными и ступенчатыми массивами направлена на отработку навыков написания вложенных алгоритмов.

Раздел *«Строки»*. В связи с тем, что текстовые строки по своей сути представляют собой массив символов, за исключением того, что текстовые строки в отличие от массивов являются неизменяемым типом данных, изучение работы со строками осуществляется сразу же после массивов. При изучении темы акцент делается на четкое понимание отличия символьного типа данных *char* от строкового типа *string*, что особенно важно после изучения языка Python, в котором нет разделения отдельных символов и строк как различных типов данных. Здесь кроме написания собственных алгоритмов также активно применяются готовые методы для работы с символами и строками. В связи с тем, что работа с циклами, как показывает практика, часто вызывает у обучающихся затруднения, отдельное внимание уделяется циклической обработке элементов строки.

Раздел *«Статические методы»*. После изучения основных алгоритмических структур и типов данных мы переходим к изучению темы создания собственных функций. В C# невозможно создать обычную функцию, как это позволяют делать многие другие языки программирования. Все функции в C# создаются только внутри классов, поэтому мы используем соответствующий термин для таких функций – «метод». Студенты здесь более глубоко погружаются в тему представления функций в языках программирования, она является своеобразным мостом между тем, что они знали до этого (обычные функции, с которыми они уже работали, программируя на Python) и тем, что еще предстоит изучить (объектно-ориентированное программирование, методы класса). Однако, поскольку изучение объектно-ориентированного программирования не входит в данный курс, обучающиеся знакомятся только с понятием «статического» метода, который может быть вызван без создания объекта, и пишут исключительно такие методы. В рамках раздела изучаются особенности написания методов, возвращающих и не возвращающих значение, ключевое слово *return*, типы параметров и аргументов (обязательные параметры, аргументы по умолчанию, произвольное количество аргументов (ключевое слово *params*)), области видимости переменных, рекурсия. По сравнению с аналогичной темой при изучении программирования на языке Python, здесь добавляются такие элементы содержания курса как ссылки (ключевые слова *ref* и *out*) и перегрузка методов (возможность создания методов с одинаковыми именами, но различным функционалом, количеством и типом параметров).

Раздел *«Коллекции»*. Работа с различными структурами данных заслуживает отдельного внимания при изучении любого языка программирования. В языке программирования C# реализовано большое количество различных классов для работы с коллекциями данных. В курсе мы рассматриваем такие коллекции как *ArrayList*, *List*, *HashTable*, *SortedList*, *Stack* и *Queue*. В отличие от обычного массива коллекции имеют динамическую длину, в каждой коллекции реализован собственный набор методов для обработки хранимых данных, а также коллекции позволяют хранить в одном наборе данные различных типов. Эта особенность коллекций облегчает работу при решении многих задач с большими наборами данных.

Раздел *«Текстовые файлы»*. Рассматривается работа только с текстовыми файлами (не бинарными). Изучаются различные способы создания текстового файла и записи в него текстовых данных, а также чтение данных из текстового файла. Уделяется внимание работе с текстовыми файлами с различной кодировкой.

Как было отмечено выше, ввиду того, что на аудиторную практическую работу выделено всего 26 академических часов, на занятиях изучение языка C# ограничивается указанными выше темами. При достаточном количестве академических часов для раскрытия потенциала языка программирования C# возможно включение в практическую часть курса следующих тем (разделов).

Раздел *«Обработка исключений»*. Рассматриваются виды возможных ошибок при написании программы: логические ошибки, ошибки времени компиляции (compile time errors) или синтаксические ошибки, ошибки времени выполнения (runtime errors). Основное внимание уделяется ошибкам времени выполнения, по сути являющимися «исключениями». Рассматриваются типы исключений и способы их обработки,

а также возможность получения информации о сработавшем исключении. Кроме того, уделяется внимание генерации собственных исключений.

Раздел *«Основы объектно-ориентированного программирования (ООП)»*. Изучение ООП является очень ресурсоемким по времени, в связи с чем авторами рекомендуется изучение данного раздела в рамках отдельного учебного курса (учебной дисциплины). Но для ознакомления с базовыми принципами ООП и некоторыми особенностями ООП в языке C# в данном разделе предлагается изучение ряда основных концепций.

Первый этап изучения раздела предполагает рассмотрение понятий «класс», «структура» и «объект», а к основным элементам его содержания можно отнести: создание класса и объекта класса; модификаторы доступа; изменение и получение значений полей; понятие инкапсуляции. При изучении методов рассматриваются как методы объекта, так и статические методы. В блоке, посвященном свойствам (методам доступа к полям), уделяется внимание модификаторам доступа в свойствах, автоматическим свойствам. Далее рассматриваются различного вида конструкторы и деструкторы, инициализаторы и деконструкторы.

На втором этапе рассматриваются понятия «наследование» и «полиморфизм», возможности преобразования типа объектов, перегрузка операторов. При изучении наследования рассматривается также наследование конструкторов, возможность запрета наследования. Блок изучения полиморфизма включает в себя следующие элементы содержания: переопределение методов, сокрытие методов, запрет на переопределение методов, переопределение свойств, абстрактные классы и члены классов.

Далее рассматривается возможность использования индексаторов для обращения к данным объекта по индексу, а также понятие значения *null* и операторы *??*, *??=*, *?*, проверка объекта на значение *null*.

На третьем этапе предлагается изучение блоков: делегаты, лямбда-выражения, события. Изучение делегатов включает в себя следующее содержание: объявление делегата, присвоение делегату ссылки на метод, добавление и удаление методов в делегатах, вызов делегата, объединение делегатов, использование делегата в качестве параметра метода, обобщенные делегаты. Блок, посвященный лямбда-выражениям, включает в себя создание лямбды, параметры лямбды, добавление и удаление действий в лямбда-выражении, возвращение результата, лямбда-выражение как аргумент или результат метода. Блок, посвященный событиям, включает определение и вызов события, добавление и удаление обработчика события, управление обработчиками, передача данных события.

Раздел *«Использование ООП для создания пользовательских интерфейсов (UI) и программного обеспечения для стандартных операционных процедур (СОП)»*. В данном разделе рассматривается создание классических оконных приложений Windows Forms с использованием базовых компонентов: форма, кнопка, метка, текстовое поле, поле для изображения, списки, флажки, переключатели, таймер и т.д. Также рассматривается работа с файловой системой. Классические приложения Windows Forms имеют ограничения при работе с прозрачностью, по созданию окон нестандартной формы и др. Для разработки более интерактивных и графически разнообразных интерфейсов рассматривается использование фреймворка WPF (Windows Presentation Foundation). Уделяется внимание работе с диалоговыми окнами и созданию многооконных приложений.

Раздел *«Элементы функционального программирования»*. В этом разделе рассматривается понятие чистой функции, делается акцент на неизменяемые типы данных, изучаются реализация функции *map* (аналог функции из языка Python) при помощи методов расширения LINQ, реализация фильтрации при помощи методов расширения LINQ, реализация функции сокращения (*reduce*) при помощи методов расширения LINQ.

Раздел *«Тестирование программного обеспечения»*. В данном разделе изучается модульное тестирование, интеграционное тестирование, функциональное тестирование, параметризованные тесты и тестовые наборы, автоматизация процесса тестирования. Рассматривается использование ряда фреймворков и инструментов для тестирования: NUnit, MSTest, xUnit, Moq, Entity Framework, Selenium для тестирования веб-интерфейсов, SpecFlow для BDD (Behavior Driven Development).

Учебные материалы, связанные с перечисленными выше разделами, предлагаются студентам для самостоятельного изучения и предоставляются в электронном виде внутри системы управления обучением вуза. Перечень разделов может быть расширен с учетом количества академических часов, выделенных на изучение дисциплины в конкретном учебном заведении.

Проблема «сильных» студентов

Как было отмечено выше, непосредственно перед курсом программирования на языке C# обучающиеся изучали курс программирования на языке Python. В связи с тем, что оба курса представляют изучение основ программирования, т.е.

направлены на формирование и развитие у обучающихся навыков написания и применения алгоритмов для решения прежде всего типовых задач, в том числе школьного уровня, наборы изучаемых разделов в них практически идентичны. Различия курсов заключаются в акцентировании внимания на синтаксисе и возможностях каждого языка для решения определенного типа задач. Такая особенность построения дисциплины «Программирование» приводит к тому, что наборы задач для предшествующего курса программирования на языке Python и текущего курса программирования на языке C# тем или иным образом пересекаются (многие задачи совпадают или включают в себя подзадачи, которые уже решались на другом языке). Возникла проблема: как построить практическую часть курса таким образом, чтобы и наименее успевающие («слабые») обучающиеся могли решать задачи и продолжать развивать свои навыки программирования, и наиболее успевающие («сильные») обучающиеся не потеряли интерес к программированию при решении простых, повторяющихся задач? Наиболее «сильные» студенты, успешно освоившие курс программирования в прошлом семестре, потенциально готовы к самостоятельной, творческой деятельности по программированию, способствующей развитию их профессионально значимых качеств, формированию готовности к самореализации в будущей профессиональной деятельности, но стандартные учебные задачи даже на новом для студентов языке не подходят для организации такой деятельности.

Возможным решением проблемы является разработка для более «сильных» студентов заданий на создание визуальных приложений (приложений с графическим интерфейсом). Но в таком случае обучающимся придется полностью самостоятельно искать и изучать теоретический материал по данному направлению. В виду ограниченного количества часов, выделенных на аудиторную работу в рамках дисциплины, данный подход не позволит студентам полностью освоить весь необходимый материал, поскольку, фактически, обучающиеся будут самостоятельно изучать совершенно другую дисциплину.

Авторами было принято решение создать два набора заданий. Первый набор – это задачи с постепенным увеличением сложности соответственно изучаемому теоретическому материалу. Второй набор представляет собой «занимательные задачи» – сюжетные практико-ориентированные задачи, решение которых требует более глубоких знаний в области программирования, умений и навыков применения изученного ранее материала в комплексе. При этом для решения задач достаточно учебных материалов курса, но с двумя замечаниями: для решения каждой задачи требуется материал сразу нескольких тем, который надо изучить предварительно или в процессе решения, когда студент «почувствует» недостаток знаний; требуется знание команд для работы с консолью (полей и методов класса *Console* и структуры *ConsoleKeyInfo*). Описание команд и примеры их работы демонстрируются преподавателями на практических занятиях и предоставляются в электронном виде.

«Занимательные задачи» в курсе программирования

Под «занимательными задачами» обычно понимаются развивающие задачи, использующие нестандартные формулировки и способы представления данных, вызывающие интерес в связи с их внешней привлекательностью, побуждающие к самостоятельному поиску решения [19]. Отметим, что к формам «занимательных задач» по информатике (и не только), как правило, относят загадки, головоломки, викторины, кроссворды, ребусы [20, 21], но не компьютерные программы, которые требуется написать.

Занимательность предлагаемых авторами задач по программированию состоит главным образом в трёх аспектах: игровая направленность (задачи связаны с моделированием игровых ситуаций, с созданием полноценных игр или их элементов); продумывание и разработка архитектуры приложения; формирование символьной графики для визуализации моделируемых процессов в условиях ограничений консольного вывода. Они также стимулируют самостоятельный поиск обучающимся недостающей информации. Решение может занимать длительное время (более 2 академических часов); нет ограничений по времени выполнения кода и затрачиваемой памяти, хотя эффективность алгоритмов может оцениваться преподавателем; практически не требуется знание каких-либо математических алгоритмов и структур данных кроме массивов (главное отличие от классического «олимпиадного» программирования), задачи направлены прежде всего на построение архитектуры приложения – организацию и структурирование программного кода с помощью подпрограмм (процедурное программирование) и/или с помощью описания собственных типов/моделей (объектно-ориентированное программирование).

Решение задач, основанных на графическом выводе – анимационных историй или игр, повышает осведомленность обучающихся о возможностях изучаемых технологий [5], может способствовать возникно-

вению у студентов творческих идей и познавательной мотивации [5, 22, 23], в частности, к изучению объектно-ориентированного программирования и улучшению понимания эффективности этой парадигмы. Исследования показывают, что изучение основ объектно-ориентированного программирования с использованием визуальных учебных сред и графических устройств, в частности для программирования игр, формирует лучшее понимание планируемых результатов работы программы, назначение и структуру требуемых классов, функционал объектов и способы их взаимодействия между собой [2, 24–26].

Обучающимся в начале курса дается выбор: решать стандартный набор заданий со всей учебной группой или решать набор «занимательных задач». Второй вариант подразумевает выделение большего времени обучающимся для самостоятельной работы. Студенты могут изменить свой выбор в любое время в течение семестра. Первая группа по завершении курса проходит итоговую аттестацию на общих основаниях. Теоретическая часть сдается в виде теста, далее обучающемуся предлагается случайное практическое задание. Вторая группа при полном решении всех задач и их подробном объяснении («защите») получает оценку «отлично» за экзамен. Если же обучающийся из второй группы не решит все задачи или не защитит их решение, тогда он так же сдает экзамен на общих основаниях.

Сюжетные практико-ориентированные задачи

Приведем примеры сюжетных практико-ориентированных задач по программированию, предлагаемых обучающимся для решения на языке C#, включающие формулировки задач и скриншоты консольного вывода, демонстрирующие предполагаемые результаты решения. Все представленные ниже задачи могут быть решены с использованием только процедурной парадигмы программирования, что позволяет студентам включиться в процесс их решения при освоении учебного курса, в который не включены основы объектно-ориентированного программирования. В то же время студенты, уже знакомые с объектно-ориентированным программированием или студенты, желающие освоить эту концепцию, также получают возможность решения предложенных авторами практических задач с использованием соответствующих знаний и навыков.

Рис. 1. Примеры графического отображения полос здоровья и маны в результате выполнения различных команд пользователя

Fig. 1. Examples of graphical display of health and mana bars as a result of executing various user commands

Задача «Полосы здоровья и маны»

Количество здоровья и маны некоторого игрового персонажа хранятся в натуральных числах. Даны максимальные и текущие значения здоровья и маны. В бесконечном цикле выводить полосы здоровья (зелёного цвета) и маны (голубого цвета) длиной L символов, при этом максимальные значения могут быть больше или меньше L. Размер закрашенной области полосы должен соответствовать (быть пропорциональным) текущему значению относительно максимального. После вывода полос запрашивать ввод команды.

Пользователю доступны следующие команды (N – целое число):

- *life* N – установить количество здоровья в значение N;
- *mana* N – установить количество маны в значение N;
- *exit* – завершить работу программы.

Несуществующие команды игнорировать (очищать ввод и не изменять значение здоровья/маны). Если значение N меньше 0, то устанавливать значение 0, а если больше максимального значения, то устанавливать максимальное значение.

Примеры консольного вывода для задачи «Полосы здоровья и маны» показаны на рис. 1.

Задача «Простое пианино»

На экране отображаются белые и чёрные клавиши пианино (одна октава). Под каждой белой клавишей подписана нота. При нажатии кнопок на клавиатуре проигрываются звуки и выделяются цветом клавиши, соответствующие нотам. Например, кнопка A (Ф) соответствует ноте «До» (C), кнопка S (Ы) соответствует ноте «Ре» (D), а кнопка W (Ц) соответствует ноте «До-диез» (C#).

Рис. 2. Примеры графического отображения пианино в исходном состоянии, при нажатии белой клавиши и при нажатии черной клавиши

Fig. 2. Examples of graphic displays of the piano in its initial state, when the white key is pressed, and when the black key is pressed

Примеры консольного вывода для задачи «Простое пианино» показаны на рис. 2.

Задача «Генерация ландшафта»

Дан одномерный массив заданной случайно длины, состоящий из натуральных случайных чисел (максимальное значение не превышает длину массива). Каждое число в массиве – высота фрагмента (столбца) ландшафта. Сверху льётся вода, которая вытекает за края, но остаётся во впадинах. Вывести на экран ландшафт с оставшейся во впадинах водой. Символы для земли и воды должны иметь разный цвет, причём верхний символ земли (в каждом столбце) может отличаться от остальных символов земли и должен иметь цвет, отличный от цвета остальной земли. Предусмотреть возможность изменять все используемые для вывода ландшафта символы через 3 переменные.

Примеры консольного вывода для задачи «Генерация ландшафта» показаны на рис. 3.

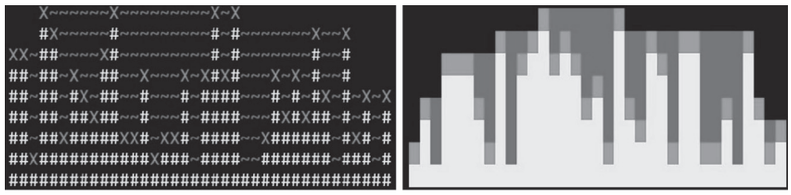

Рис. 3. Примеры графического отображения сгенерированного ландшафта, корректно заполненного водой, при использовании различных наборов символов

Fig. 3. Examples of graphical display of the generated landscape, correctly filled with water, using different character sets

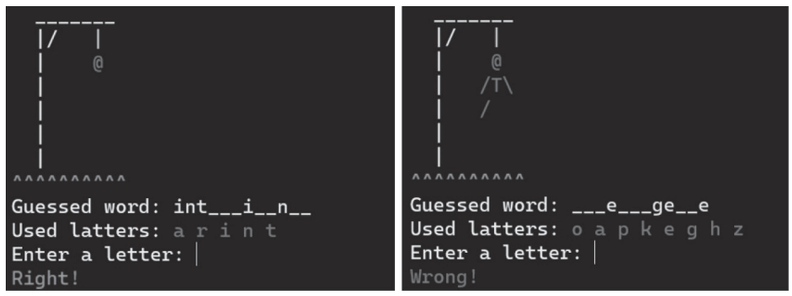

Рис. 4. Примеры отображения пользовательского интерфейса игры в консоли

Fig. 4. Examples of displaying the game user interface in the console

Задача «Игра «Виселица»»

Компьютер загадывает слово (выбирает случайно из заданного списка) и выводит такое количество подчёркиваний, сколько букв в этом слове. Игрок вводит буквы, чтобы отгадать слово. Если буква есть в слове (без учёта регистра), то компьютер вписывает её на своё место в слове (если таких букв несколько, вписываются все). Если буквы нет в слове, появляется очередной элемент виселицы с человечком. Всего элементов 7: верёвка, голова, туловище, правая рука, левая рука, правая нога, левая нога. Если игрок не успел угадать слово до того, как виселица будет нарисована полностью, он проиграл. Если слово отгадано – выиграл.

На экран следует выводить: ранее названые буквы; сообщение о том, угадана ли очередная буква; осуществлять проверку на корректность ввода: введена ли именно буква, причём та, которая ещё не была использована ранее. Если ввод некорректен, программа должна сообщать об этом и повторять запрос ввода.

Дополнительно: загаданное слово загружается из текстового файла.

Примеры консольного вывода для задачи «Игра «Виселица»» показаны на рис. 4.

Задача «Лабиринт с сокровищами»

На основе двумерного массива построить лабиринт, поместить в него игрового персонажа и сокровища. Персонаж может перемещаться на одну клетку вверх, вправо, вниз или влево при нажатии на соответствующую кнопку на клавиатуре, если в направлении движения нет стены. Когда

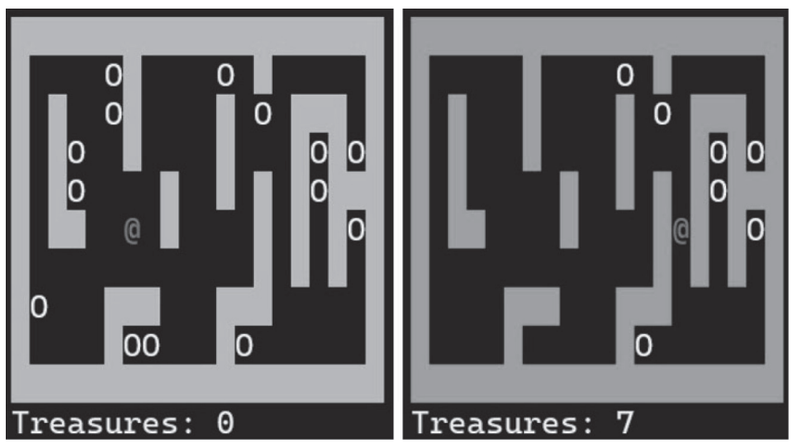

Рис. 5. Примеры графического отображения лабиринта, персонажа, сокровищ и вывода счётчика найденных сокровищ

Fig. 5. Examples of graphic display of the labyrinth, character, treasures and the output of the counter of found treasures

персонаж встаёт на клетку с сокровищем, он его подбирает – сокровище исчезает, а счётчик найденных сокровищ увеличивается на 1.

Примеры консольного вывода для задачи «Лабиринт с сокровищами» показаны на рис. 5.

Задача «Бегущий к цели»

На поле *N*x*M* стоит персонаж. В случайном месте появляется его цель. Персонаж добегает до цели (использовать задержку выполнения программы) и действия повторяются, причём персонаж начинает с той точки, где остановился (где была предыдущая цель). Для вычисления пути персонажа до цели использовать алгоритм Брезенхема для рисования отрезков. Реализовать отображение пути между персонажем и целью.

Примеры консольного вывода для задачи «Бегущий к цели» показаны на рис. 6.

Задача «Дом с привидениями»

Игроку демонстрируются *N* закрытых дверей, за одной из которых находится привидение, и предлагается ввести номер двери (1, 2 и т.д.), в которую он войдет. Если игрок выбрал дверь с привидением, игра заканчивается. Войдя в очередную дверь, игрок попадает в следующую комнату, где действия повторяются (снова выбор из *N* дверей). Дверь, за которой будет находиться привидение, для каждой комнаты выбирается случайно. Цель игры – посетить как можно больше комнат, не встретив привидение. Счётчик посещённых комнат следует выводить на экран. После выбора очередной двери демонстрировать игроку результат (за дверью привидение или свободно) и запрашивать ввод для продолжения игры. Ввод номера двери проверять на корректность – игнорировать любое значение, если было введено не число в пределах диапазона нумерации дверей.

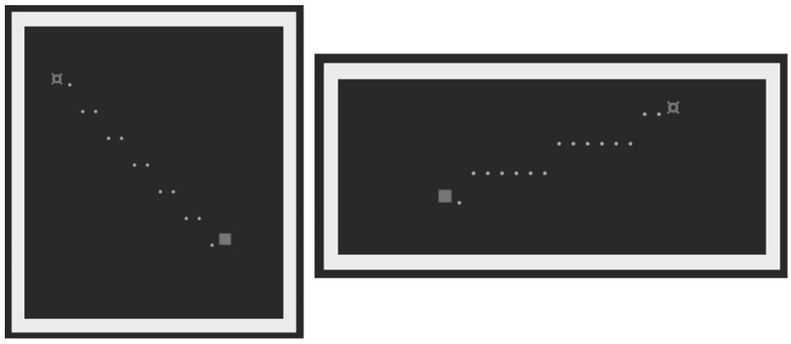

Рис. 6. Примеры графического отображения бегущего персонажа, его цели и пути к ней на полях разного размера

Fig. 6. Examples of graphic display of the running character, the goal and the path to it on fields of different sizes

Дополнительно

- Случайное количество дверей в каждой комнате, но не менее 2-х.
- Несколько жизней. Если игрок встречает привидение, то теряет одну жизнь и может перейти в следующую комнату.
- За дверью без привидения может с небольшим шансом оказаться магический меч. Если у игрока уже есть меч, другой меч появиться не может. Если игрок с мечом встречает призрака, то не проигрывает (не теряет жизнь), однако теряет меч. После этого меч можно найти снова. Наличие меча должно быть обозначено в виде его изображения.

Примеры консольного вывода для задачи «Дом с привидениями» со всеми выполненными дополнительными подзадачами показаны на рис. 7.

Задача «Проверка документов»

Пользователю демонстрируется документ человека, в котором указаны его имя, пол и год рождения. Необходимо проверить документ на соответствие случайно выбранному правилу:

- возраст больше *N* лет;
- возраст меньше *N* лет;
- заданный пол (мужчина/женщина);
- заданный цвет документа (красный/зелёный/синий).

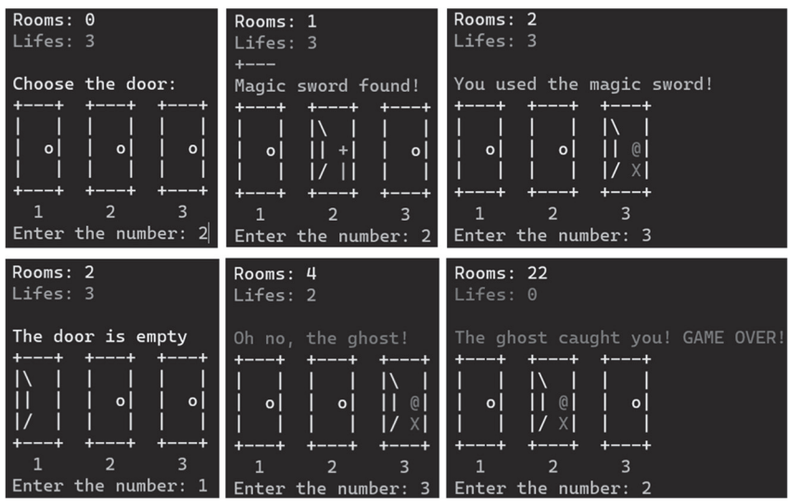

Рис. 7. Примеры отображения графического интерфейса игры «Дом с привидениями» в консоли

Fig. 7. Examples of displaying the graphical interface of the "Haunted house" game in the console

Одно правило действует на 5-10 документов подряд, затем меняется.

Текущий год задаётся случайно и выводится на экран. Возраст человека вычисляется как разность текущего года и года рождения.

Имя генерируется случайно по алгоритму чередования согласных и гласных букв, причём первая буква должна быть прописной. Если имя принадлежит женщине, оно должно оканчиваться на «а». Возраст генерируется от 14 лет. Изображение (фотография) человека в документе меняется в зависимости от пола (2 варианта).

Для каждого документа требуется ответить на вопрос, корректный он или нет (соответствует ли текущему правилу) с помощью ввода символа «y» (да) или «n» (нет) без учёта регистра. Неверный формат ввода обрабатывать, очищая место под верный. Результат должен отображаться в виде надписи «верно»/«неверно». Количество верных ответов пользователя (документ действительно корректный или действительно некорректный) подсчитывается и выводится на экран.

Примеры консольного вывода для задачи «Проверка документов» показаны на рис. 8.

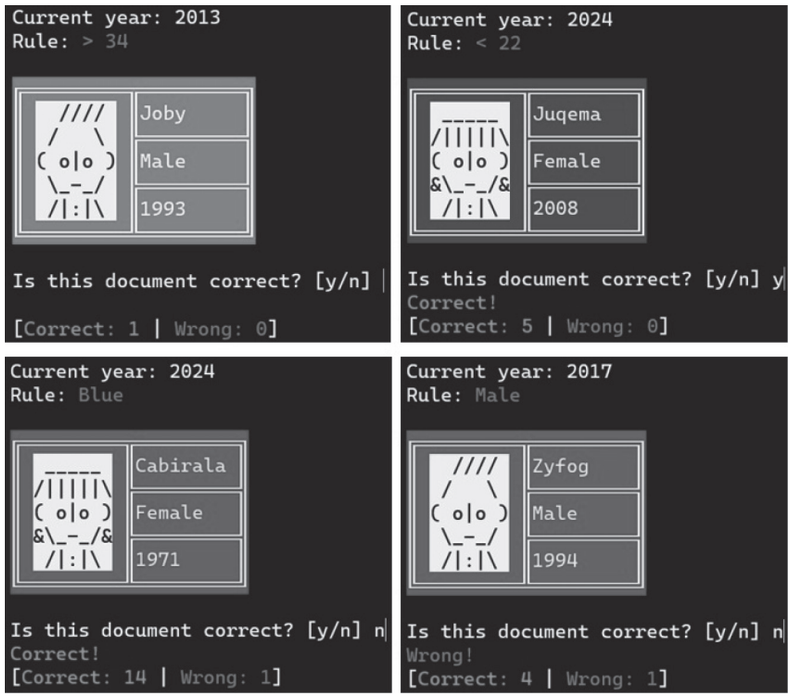

Рис. 8. Примеры отображения графического интерфейса программы «Проверка документов» в консоли

Fig. 8. Examples of displaying the graphical interface of the "Document verification" program in the console

Заключение

Разработанная система «занимательных задач» опирается на три ключевых принципа:

- принцип постепенного усложнения (Л. С. Выготский [27]) – задачи структурированы от простых визуализаций («Полосы здоровья и маны») до комплексных проектов («Дом с привидениями», «Проверка документов»);
- принцип контекстного обучения (А. А. Вербицкий [28]) – каждая задача выступает в качестве проблемной ситуации, при этом моделирует реальную игровую механику, обеспечивая смысловую вовлеченность;
- принцип продуктивных ограничений (С. Пейперт (S. Papert) [29]) – ограничения графических возможностей консоли сознательно используются для стимулирования творческого поиска, ошибки – часть обучения.

Методологическая разработанность и непротиворечивость данных положений, а также анализ отечественного и зарубежного опыта применения сюжетных, практико-ориентированных учебных задач по программированию, отраженного в упомянутых в начале статьи публикациях, позволяют предположить, что предлагаемые авторами сюжетные задачи с консольной визуализацией в курсе программирования на языке C# могут способствовать активизации творческой самостоятельной работы обучающихся.

В рамках апробации в Новосибирском государственном педагогическом университете решение «занимательных задач» выбрало небольшое количество студентов даже среди наиболее «сильных» в учебных группах. Всего в 2023-2024 учебном году задачи решало 4 студента, в 2024-2025 учебном году – 1 студент. Из них только 2 завершили решение всех задач к экзамену и успешно защитили их. Один из этих студентов впоследствии написал и защитил курсовую работу по теме «Разработка занимательных задач по программированию на языке C#», в которой представил собственные разработанные задания, а также использовал из в рамках педагогической практики в 11 классе с углубленным изучением информатики. Это способствовало развитию студента и как будущего педагога, и как программиста.

В целом, несмотря на крайне небольшое количество студентов, выбравших «занимательные задачи» как альтернативу основному курсу, считаем, что задачи были разработаны не зря и их при-

менение оправдано, поскольку их целевая аудитория – студенты с высоким уровнем мотивации к изучению программирования, стремящиеся к самообразованию, готовые активно учиться вне университета – весьма немногочисленна.

Подобные сюжетные практико-ориентированные задачи могут применяться не только для работы со студентами, но и с обучающимися школ, интересующимися программированием и стремящимися выйти за рамки школьной программы, желающими опробовать новый для себя и в то же время актуальный в профессиональной среде язык программирования.

Литература

1. Кольцова К.И. Использование сюжетных задач при обучении программированию на Python // Информатика в школе. 2023. № 2 (180). С. 7–12. DOI: 10.32517/2221-1993-2023-22-1-7-12.
2. Павлов Д.И., Бутарев К.В., Балашова Е.В. О перспективах использования технологий геймификации при раннем обучении объектно-ориентированному программированию // Современные информационные технологии и ИТ-образование. 2018. № 14(4). С. 977–985. DOI: 10.25559/SITITO.14.201804.977-985.
3. Соколова А.Н., Шалагинова Н.В. Сюжетные задачи как средство мотивации школьников при изучении основ программирования // III Международная междисциплинарная конференция «Проблемы теории и практики инновационного развития и интеграции современной науки и образования» (Москва, 16 февраля 2022 г.). М.: Московский государственный областной университет, 2022. С. 178–184.
4. Соколова А.Н., Шалагинова Н.В. Использование практико-ориентированных задач при обучении старшеклассников программированию на Python // IV Международная междисциплинарная конференция «Проблемы теории и практики инновационного развития и интеграции современной науки и образования» (Москва, 15 февраля 2023 г.). М.: Государственный университет просвещения, 2024. С. 276–281.
5. Giannakos M.N., Jaccheri L. From players to makers: An empirical examination of factors that affect creative game development // International Journal of Child-Computer Interaction. 2018. № 18. С. 27–36. DOI: 10.1016/j.ijcci.2018.06.002.
6. Лавина Т.А., Мытникова Е.А., Яруськина Е.Т. Подготовка по программированию бакалавров направления «Программная инженерия» с учетом концепции комплексного подхода к инженерному образованию // Современные наукоемкие технологии. 2023. № 7. С. 160–166. DOI: 10.17513/snt.39712.
7. Завьялова О.А., Маркелов В.К. Возможности онлайн-сред программирования при обучении языку Python в школе // Информатика в школе. 2022. № 3(176). С. 75–82. DOI: 10.32517/2221-1993-2022-21-3-75-82.
8. Козлов О.А., Барышева И.В., Малкина Е.В., Шестакова Н.В. Обучение школьников программированию в рамках предмета «Информатика»: проблемы и возможные решения // Информатика в школе. 2023. № 184(5). С. 67–73. DOI: 10.32517/2221-1993-2023-22-5-67-73.
9. Кривоплясова Е.В., Нефёдова В.Ю., Прилепина А.В. Методика обучения основам программирования на языке Python // Информатика в школе. 2020. № 3(156). С. 24–30. DOI: 10.32517/2221-1993-2020-19-3-24-30.
10. Маркелов В.К., Завьялова О.А. Язык программирования Python как альтернативный инструмент для решения заданий ЕГЭ по информатике // Информатика в школе. 2023. № 2(181). С. 63–72. DOI: 10.32517/2221-1993-2023-22-2-63-72.
11. Панова И.В., Коливнык А.А. Методические аспекты обучения программированию на языке Python в школьном курсе информатики // Информатика в школе. 2020. № 6(159). С. 47–50. DOI: 10.32517/2221-1993-2020-19-6-47-50.
12. Пустыльник Ю.Ю., Чмыхова Е.В., Сальцева А.Д., Додуева С.Ж. Практики изучения языков программирования в школах России: результаты пилотного исследования // Сборник аналитических материалов «Педагогические практики подготовки школьников к олимпиаде по искусственному интеллекту». М.: Институт стратегии развития РАО, 2022. С. 37–52.
13. Сёмин М.С., Федченко Г.М. Обучение программированию на Python в качестве первого языка // Всероссийская научно-практическая конференция «Актуальные проблемы обучения математике, информатике, экономике и естественнонаучным дисциплинам в средней и высшей школе» / Под общ. ред. Н.В. Ермак (Благовещенск, 25 марта 2019 г.). Благовещенск: Благовещенский государственный педагогический университет, 2019. С. 225–228.
14. Самылкина Н.Н. Основные подходы к построению структуры и содержания школьного курса информатики углубленного уровня // Наука и школа. 2019. № 6. С. 171–182. DOI: 10.31862/1819-463X-2019-6-171-182.
15. Самылкина Н.Н. Структура и содержание цифровых компетенций, формируемых в предпрофессиональном обучении // Информатика в школе. 2020. № 4(157). С. 11–19. DOI: 10.32517/2221-1993-2020-19-4-11-19.
16. Хейлсберг А., Торгерсен М., Вилтамут С., Голд П. Язык программирования C#. Классика Computers Science. 4-е изд. СПб.: Питер, 2012. 784 с.

17. Нигматулина Э.А., Пак Н.И. Студент-центрированное обучение программированию в педагогическом вузе // Информатика и образование. 2017. № 2 (281). С. 8–14.

18. Мингалеева Л.Б., Киамова Н.И. Разработка консольных приложений на языке C# при изучении дисциплины «Программирование на языках высокого уровня» // Проблемы современного педагогического образования. 2017. № 55–11. С. 90–97.

19. Беляева М.Б., Ильясова Е.А., Калякина Е.А., Савинкова М.М. Методические рекомендации по использованию занимательных задач на уроках информатики // Вестник педагогических наук. 2023. № 7. С. 168–174.

20. Зубрилин А.А., Зубрилина М.С. Использование игровых элементов во внеурочной деятельности по информатике // Информатика в школе. 2022. № 4 (177). С. 28–35. DOI: 10.32517/2221-1993-2022-21-4-28-35.

21. Сейдаметова С., Исмаилова А.Р. Методические рекомендации по использованию занимательных задач в информатике // Информационно-компьютерные технологии в экономике, образовании и социальной сфере. 2018. № 22(4). С. 101–106.

22. Рогожкина И.Б. Развивающий эффект обучения программированию: психолого-педагогические аспекты // Психология. Журнал Высшей школы экономики. 2012. № 9(2). С. 141–148.

23. Ding A. C. E., Yu C. H. Serious game-based learning and learning by making games: Types of game-based pedagogies and student gaming hours impact students' science learning outcomes // Computers & Education. 2024. Т. 218. DOI: 10.1016/j.compedu.2024.105075.

24. Сидоренко Д.В., Бикмуллина И.И. Игровой метод обучения языку C# // Научно-технический вестник Поволжья. 2023. № 12. С. 107–109.

25. Шкарбан Ф.В. Обучение объектно-ориентированному программированию бакалавров прикладной информатики: реализация модели обучения на основе двух согласованных дисциплин // Вестник Российского университета дружбы народов. Серия: Информатизация образования. 2018. № 15(4). С. 388–397. DOI: 10.22363/2312-8631-2018-15-4-388-397.

26. Corral J.M.R, Balcells A.C., Estevez A.M., Moreno G. J., Ramos M. J. F. A game-based approach to the teaching of object-oriented programming languages // Computers & Education. 2014. № 73. С. 83–92. DOI: 10.1016/j.compedu.2013.12.013.

27. Выготский Л.С. Мышление и речь. Москва; Ленинград: Государственное учебно-педагогическое издательство, 1934. 324 с.

28. Вербицкий А.А. Компетентностный подход и теория контекстного обучения: Материалы к четвертому заседанию методологического семинара 16 ноября 2004 г. М.: Исследовательский центр проблем качества подготовки специалистов, 2004. 84 с.

29. Пейперт С. Переворот в сознании: Дети, компьютеры и плодотворные идеи. М.: Педагогика, 1989. 224 с.

References

1. Kol'tsova K.I. Using Story-Based Tasks in Teaching Python Programming. Informatika v shkole = Computer Science at School. 2023; 2(180): 7–12. DOI: 10.32517/2221-1993-2023-22-1-7-12. (In Russ.)

2. Pavlov D.I., Butarev K.V., Balashova Ye.V. On the Prospects of Using Gamification Technologies in Early Learning of Object-Oriented Programming. Sovremennyye informatsionnyye tekhnologii i IT-obrazovaniye = Modern Information Technologies and IT Education. 2018; 14(4): 977–985. DOI: 10.25559/SITITO.14.201804.977-985. (In Russ.)

3. Sokolova A.N., Shalaginova N.V. Story-based tasks as a means of motivating schoolchildren in learning the basics of programming. III Mezhdunarodnaya mezhdistsiplinarnaya konferentsiya «Problemy teorii i praktiki innovatsionnogo razvitiya i integratsii sovremennoy nauki i obrazovaniya» = III International Interdisciplinary Conference «Problems of Theory and Practice of Innovative Development and Integration of Modern Science and Education» (Moscow, February 16, 2022). Moscow: Moscow State Regional University; 2022: 178–184. (In Russ.)

4. Sokolova A.N., Shalaginova N.V. Using practice-oriented tasks in teaching high school students programming in Python. IV Mezhdunarodnaya mezhdistsiplinarnaya konferentsiya «Problemy teorii i praktiki innovatsionnogo razvitiya i integratsii sovremennoy nauki i obrazovaniya» = IV International Interdisciplinary Conference «Problems of Theory and Practice of Innovative Development and Integration of Modern Science and Education» (Moscow, February 15, 2023). Moscow: State University of Education; 2024: 276-281. (In Russ.)

5. Giannakos M.N., Jaccheri L. From players to makers: An empirical examination of factors that affect creative game development. International Journal of Child-Computer Interaction. 2018; 18: 27–36. DOI: 10.1016/j.ijcci.2018.06.002.

6. Lavina T.A., Mytnikova Ye.A., Yarus'kina Ye.T. Programming training for bachelors in Software Engineering, taking into account the concept of an integrated approach to engineering education. Sovremennyye naukoyemkiye tekhnologii = Modern Science-Intensive Technologies. 2023; 7: 160–166. DOI: 10.17513/snt.39712. (In Russ.)

7. Zav'yalova O.A., Markelov V.K. Possibilities of Online Programming Environments in Teaching Python at School. Informatika v shkole = Computer Science at School. 2022; 3(176): 75–82. DOI: 10.32517/2221-1993-2022-21-3-75-82. (In Russ.)

8. Kozlov O.A., Barysheva I.V., Malkina Ye.V., Shestakova N.V. Teaching Schoolchildren Programming as Part of the Subject «Computer Science»: Problems and Possible Solutions. Informatika v shkole = Computer Science at School. 2023; 184(5): 67–73. DOI: 10.32517/2221-1993-2023-22-5-67-73. (In Russ.)

9. Krivoplyasova Ye.V., Nefodova V.Yu., Prilepina A.V. Methods of Teaching the Basics of Programming in Python. Informatika v shkole = Computer Science at School. 2020; 3(156): 24–30. DOI: 10.32517/2221-1993-2020-19-3-24-30. (In Russ.)

10. Markelov V.K., Zav'yalova O.A. The Python Programming Language as an Alternative Tool for Solving Unified State Exam Tasks in Computer Science. Informatika v shkole = Computer Science at School. 2023; 2(181): 63–72. DOI: 10.32517/2221-1993-2023-22-2-63-72. (In Russ.)

11. Panova I.V., Kolivnyk A.A. Methodological Aspects of Teaching Python Programming in a School Computer Science Course. Informatika v shkole = Computer Science at School. 2020; 6(159): 47-50. DOI: 10.32517/2221-1993-2020-19-6-47-50. (In Russ.)

12. Pustyl'nik Yu.Yu., Chmykhova Ye.V., Sal'tseva A.D., Doduyeva S.Zh. Practices for Studying Programming Languages in Russian Schools: Results of a Pilot Study. Sbornik analiticheskikh materialov «Pedagogicheskiye praktiki podgotovki shkol'nikov k olimpiade po iskusstvennomu intellektu» = Collection of Analytical Materials «Pedagogical Practices for Preparing Schoolchildren for the Artificial Intelligence Olympiad «. Moscow: Institute for Development Strategy, Russian Academy of Education; 2022: 37–52. (In Russ.)

13. Somin M.S., Fedchenko G.M. Teaching Python Programming as a First Language. Vserossiyskaya nauchno-prakticheskaya konferentsiya «Aktual'nyye problemy obucheniya matematike, informatike, ekonomike i yestestvennonauchnym distsiplinam v sredney i vysshey shkole» = All-Russian Scientific and Practical Conference «Actual Problems of Teaching Mathematics, Computer Science, Economics, and Natural Sciences in Secondary and Higher Schools» – Ed. N. V. Ermak (Blagoveshchensk, March 25, 2019). Blagoveshchensk: Blagoveshchensk State Pedagogical University; 2019: 225–228. (In Russ.)

14. Samylkina N.N. Basic Approaches to Building the Structure and Content of an Advanced School Computer Science Course. Nauka i shkola = Science and School. 2019; 6: 171–182. DOI: 10.31862/1819-463X-2019-6-171-182. (In Russ.)

15. Samylkina N.N. Structure and Content of Digital Competencies Developed in Pre-Professional Training. Informatika v shkole = Computer Science at School. 2020; 4(157): 11–19. DOI: 10.32517/2221-1993-2020-19-4-11-19. (In Russ.)

16. Kheylsberg A., Torgersen M., Viltamut S., Gold P. YAzyk programmirovaniya C#. Klassika Computers Science. 4-ye izd. = The C# Programming Language. Computer Science Classics. 4th ed. Saint Petersburg: Piter; 2012. 784 p. (In Russ.)

17. Nigmatulina E.A., Pak N.I. Student-Centered Teaching of Programming in a Pedagogical University. Informatika i obrazovaniye = Computer Science and Education. 2017; 2(281): 8–14. (In Russ.)

18. Mingaleyeva L.B., Kiamova N.I. Developing console applications in C# when studying the discipline «Programming in high-level languages». Problemy sovremennogo pedagogicheskogo obrazovaniya = Problems of modern pedagogical education. 2017; 55-11: 90–97. (In Russ.)

19. Belyayeva M.B., Il'yasova Ye.A., Kalyakina Ye.A., Savinkova M.M. Methodological recommendations for the use of entertaining tasks in computer science lessons. Vestnik pedagogicheskikh nauk = Bulletin of pedagogical sciences. 2023; 7: 168–174. (In Russ.)

20. Zubrilin A.A., Zubrilina M.S. Using game elements in extracurricular activities in computer science. Informatika v shkole = Computer science at school. 2022; 4(177): 28–35. DOI: 10.32517/2221-1993-2022-21-4-28-35. (In Russ.)

21. Seydametova S., Ismailova A.R. Methodological recommendations for the use of entertaining tasks in computer science. Informatsionno-komp'yuternyye tekhnologii v ekonomike, obrazovanii i sotsial'noy sfere = Information and computer technologies in economics, education and social sphere. 2018; 22(4): 101–106. (In Russ.)

22. Rogozhkina I.B. Developmental effect of teaching programming: psychological and pedagogical aspects. Psikhologiya. Zhurnal Vysshey shkoly ekonomiki = Psychology. Journal of the Higher School of Economics. 2012; 9(2): 141–148. (In Russ.)

23. Ding A. C. E., Yu C. H. Serious game-based learning and learning by making games: Types of game-based pedagogies and student gaming hours impact students' science learning outcomes. Computers & Education. 2024: 218. DOI: 10.1016/j.compedu.2024.105075.

24. Sidorenko D.V., Bikmullina I.I. A Game-Based Method of Teaching the C# Language. Nauchno-tekhnicheskiy vestnik Povolzh'ya = Scientific and Technical Bulletin of the Volga Region. 2023; 12: 107–109. (In Russ.)

25. Shkarban F.V. Teaching Object-Oriented Programming to Bachelors of Applied Computer Science: Implementation of a Learning Model Based on Two Coordinated Disciplines. Vestnik Rossiyskogo universiteta druzhby narodov. Seriya: Informatizatsiya obrazovaniya = Bulletin of the Peoples' Friendship University of Russia. Series: Informatization of Education. 2018; 15(4): 388–397. DOI: 10.22363/2312-8631-2018-15-4-388-397. (In Russ.)

26. Corral J.M. R, Balcells A.C., Estevez A.M., Moreno G.J., Ramos M.J. F. A game-based approach

to the teaching of object-oriented programming languages. Computers & Education. 2014; 73: 83–92. DOI: 10.1016/j.compedu.2013.12.013.

27. Vygotskiy L.S. Myshleniye i rech' = Thinking and Speech. Moscow; Leningrad: State Educational and Pedagogical Publishing House; 1934. 324 p. (In Russ.)

28. Verbitskiy A.A. Competence-based approach and the theory of contextual learning: Proceedings for the fourth meeting of the methodological seminar on November 16, 2004. Moscow: Research Center for Problems of the Quality of Specialist Training; 2004. 84 p. (In Russ.)

29. Peypert S. Perevorot v soznanii: Deti, komp'yutery i plodotvornyye idei = Revolution in consciousness: Children, computers and fruitful ideas. Moscow: Pedagogy; 1989. 224 p. (In Russ.)

Сведения об авторах

Константин Владимирович Розов
К.п.н., учитель информатики,
МБОУ Гимназия №4, Новосибирск, Россия
Эл. почта: konstantin_dubrava@mail.ru

Алексей Владимирович Подсадников
Старший преподаватель кафедры информационных систем и цифрового образования Новосибирский государственный педагогический университет, Новосибирск, Россия
Эл. почта: cite2006@mail.ru

Николай Александрович Чупин
К.ф.-м.н., доцент, доцент кафедры информационных систем и цифрового образования, Новосибирский государственный педагогический университет,
Новосибирск, Россия
Эл. почта: chupinna@yandex.ru

Information about the authors

Konstantin V. Rozov
Cand. Sci. (Pedagogical), Computer Science teacher,
MBOU Gymnasium No. 4, Novosibirsk, Russia
E-mail: konstantin_dubrava@mail.ru

Alexey V. Podsadnikov
Senior Lecturer at the Department of Information Systems and Digital Education
Novosibirsk State Pedagogical University,
Novosibirsk, Russia
E-mail: cite2006@mail.ru

Nikolay A. Chupin
Cand. Sci. (Physics and Mathematics), Associate Professor, Associate Professor of the Department of Information Systems and Digital Education,
Novosibirsk State Pedagogical University,
Novosibirsk, Russia
E-mail: chupinna@yandex.ru