УДК 004.43+004.42

# ГИБРИДНАЯ СИСТЕМА ПРОГРАММИРОВАНИЯ ДЛЯ УЧЕБНЫХ ИСПОЛНИТЕЛЕЙ НА PYTHON

**М. В. Райко**[0000-0002-6448-6471]

*НИЦ «Курчатовский институт» – НИИСИ, г. Москва, Россия*

*Московский педагогический государственный университет, г. Москва*

rayko@niisi.ru

***Аннотация***

Рассмотрена методика разработки учебных формальных исполнителей с использованием комбинированного пиктограммно-текстового интерфейса на языке программирования Python. Актуальность исследования обусловлена необходимостью совершенствования подходов к обучению алгоритмизации и программированию в школьном курсе информатики. Представлен разработанный инструментарий для создания формальных исполнителей, сочетающий наглядность пиктограмм с возможностями текстового программирования. Особое внимание уделено практическим аспектам реализации, включая использование встроенных методов Python для обработки графических и текстовых данных.

***Ключевые слова****: формальный исполнитель, визуализация, программирование, пиктограммный интерфейс, Python.*

## ВВЕДЕНИЕ

В современных условиях развития цифровой экономики и перехода к технологическому суверенитету особую актуальность приобретает проблема обучения алгоритмизации и программированию в школьном курсе информатики. Относящиеся к предмету информатика понятия и практики, которыми в обязательном порядке должны овладеть выпускники 9-х и 11-х классов, определяются рядом федеральных документов. С 1 сентября 2025 года вступает в действие приказ Министерства просвещения Российской Федерации от 09.10.2024 № 704, который определяет «проверяемые требования к результатам освоения основной образовательной программы основного общего образования» и «проверяемые элементы содержания» [1]. В этом федеральном документе в раздел «проверяемые

элементы содержания» по предмету информатика (8-й класс) включены элементы двух типов:

с одной стороны, обучаемый должен быть ознакомлен с понятием формального исполнителя и приобрести опыт составления и выполнения алгоритмов управления формальными исполнителями на компьютере с использованием базовых алгоритмических конструкций;

с другой стороны, обучаемый должен ознакомиться с одним из текстовых языков программирования (Python, C++, Java, C#, Школьный Алгоритмический Язык) и поддерживающей этот язык системой программирования (редактор текста программ, транслятор, отладчик).

Для ознакомления учащихся с понятием формального исполнителя и практикой составления программ управления такими исполнителями педагог может использовать одну из свободно распространяемых сред пиктограммного программирования: среду Scratch Junior или отечественную цифровую образовательную среду (ЦОС) ПиктоМир [2].

Однако опыт работы со школьниками, а также студентами педагогических факультетов университетов показывает, что освоенные в таких средах навыки обучаемых по составлению программ с *«использованием циклов и ветвлений»* переносятся на текстовые языки программирования с большими затруднениями. Поэтому возникает идея предоставить школьникам возможность работы с формальными исполнителями на одном из текстовых языков программирования. В качестве такого языка в настоящей статье предложен Python. В качестве минимального набора формальных исполнителей для среды Python выбраны шесть формальных исполнителей: Робот, Вертун, Двигун, Тягун, Паровозик и Ползун которые реализованы нами в среде Python в форме библиотеки, которая названа Moscow Robots (Московские роботы).

Первый из этих формальных исполнителей, Робот, хорошо известен в педагогическом сообществе, так как он впервые появился в российской системе образования в 1990 году в учебнике информатики [3]. Исполнитель Робот приведён как пример формального исполнителя в действующей сегодня Федеральной рабочей программе предмета информатика (базовый уровень), поэтому формальный исполнитель Робот регулярно используется в заданиях ОГЭ по информатике.

## МЕТОДЫ И МАТЕРИАЛЫ

### Пиктограммное программирование

Простейшие задания по освоению базовых управляющих конструкций современных языков программирования эффективнее всего осваиваются новичками в графических средах программирования. В работе [4] был предложен язык Пикто, позволяющий составлять программы управления формальными исполнителями в пиктограммной среде.

Пиктограммы стали важной визуальной особенностью современных графических компьютерных интерфейсов [5]. Эстетическая привлекательность и возможность быстрого распознавания пиктограмм делают их незаменимым элементом цифровых образовательных сред для детей, что позволяет включать образовательный контент на самых ранних этапах развития ребенка, ещё до освоения последним чтения и письма. Универсальность пиктограмм ЦОС ПиктоМир делает его доступным детям вне зависимости от национальной принадлежности [6]. Знакомая визуальная форма, обозначающая конкретный референт (объект или понятие), позволяет ребёнку-интерпретатору легко и быстро понять предъявляемое значение, вне зависимости от языка, и сосредоточиться на изучении и изобретении алгоритмов в цифровой образовательной среде, а не на долгом изучении языка программирования.

Пиктографический язык Пикто – это настоящий язык программирования, используемый в системе ПиктоМир. Редактор-компилятор языка Пикто использует представление LL(1) грамматикой (англ. Left-to-right Leftmost derivation with 1 symbol look-ahead). При этом множество терминалов может быть представлено как в пиктографическом, так и в текстовом виде. Последнее требует разработки системы обозначений для пиктограмм языка.

Однако использование вне среды этого языка затруднялось тем, что ученики, используя вложение управляющих конструкций, обучаемые регулярно, совершали синтаксические ошибки, а диагностика и исправление таких ошибок требовали освоения новичками большого объёма дополнительного материала, не связанного с содержательными алгоритмическими вопросами. Теоретически в среде программирования можно добиться, чтобы любая составленная в нём программа стала автоматически правильной, синтаксически верной. Можно даже

расширить этот язык, позволяющий составлять «безошибочные» программы при использовании арифметических и логических выражений. Например, можно разработать редактор-компилятор, не позволяющий обучаемому совершать синтаксические ошибки. На практике удалось найти более простой подход, который исключает возможность совершения синтаксических ошибок обучаемым. Этот подход, использующий новую версию языка Пикто, представлен в статье [7].

Новый подход состоит в том, чтобы наделить программу управления формальными исполнителями, состоящую из пиктограмм, априорной двумерной структурой и восстанавливать блочную структуру программы по горизонтальным и вертикальным отступам (аналог приема, используемого в языке Python). Оказалось, что такой подход позволяет рассматривать любое расположение пиктограмм в двумерной таблице как синтаксически правильную, исполнимую программу. Если планировать использовать этот подход на начальных этапах составления алгоритмов управления формальными исполнителями, то каждая процедура библиотеки Moscow Robots снабжена не только уникальным текстовым именем, но и уникальной пиктограммой, и еще две дополнительные программы включены в разрабатываемую библиотеку:

а) простейший редактор, который размещает пиктограммы в прямоугольной таблице;

б) компилятор, который превращает любую такую таблицу в корректную программу на Python, которая может быть выполнена в стандартной среде Python.

Наличие а) и б) позволяет обучаемому решать задачи управления формальными исполнителями в два этапа: сначала пиктограммными средствами создается и отлаживается алгоритм движения робота в игровой обстановке, а затем происходит переход в «настоящую» текстовую среду программирования на языке Python, в которой обучаемый может добавить работу с числовыми или символьными данными для завершения решения задачи.

**Библиотека Moscow Robots**

Набор абстрактных исполнителей[1] Moscow Robots включает 6 движущихся роботов и 4 неподвижных устройства, моделирующих одиночные числовые и логические переменные. В набор роботов входят Вертун, Двигун, Тягун, Ползун, Робот и Паровозик, которые действуют на клетчатом игровом поле (карте) и выполняют задания по программе, созданной учащимся.

На карте в зависимости от типа размещенного на ней исполнителя (робота) могут присутствовать разделяющие клетки стенки, а также различные грузы и пометки клеток. Неподвижные устройства располагаются вне карты и могут быть добавлены составителем задания к роботу любого типа.

В рамках настоящего исследования предложено текстовое представление пиктограммных команд роботов.

Далее рассмотрена практика работы в библиотеке Moscow Robots на примере робота Вертун.

**Робот Вертун**

По легенде робот Вертун отвечает за ремонт полей-клеток космодрома, представленного в виде матричной клеточной модели. Клетки бывают трёх видов и цветов, каждый цвет несет свой функционал: голубая – обычная клетка, серая – поломанная, требующая починку командой закрасить, синяя – клетка закрашенная (после ремонта).

Вертун умеет выполнять команды-приказы:

*вперёд* – сделать, если возможно, шаг вперёд;

*налево* – повернуться на 90° налево;

*направо* – повернуться на 90° направо.

При невозможности выполнить команду *вперёд* Вертун сигнализирует об этом и перестаёт выполнять дальнейшие команды (отказ).

Робот Вертун умеет выполнять команды-вопросы (с обратной связью), которые возвращают логический ответ «да» или «нет» в зависимости от клетки, в

[1] Под абстрактным исполнителем здесь понимается исполнитель с фиксированной функциональностью, с возможностью внешнего управления работой в некоторой обстановке.

которой в данный момент находится робот, и обстановки рядом с ней: впереди стена?, впереди свободно?, клетка голубая?, клетка серая?, клетка синяя?

В ЦОС ПиктоМир к каждой из таких логических команд-вопросов прилагалась команда-отрицание: клетка голубая?, клетка не голубая?, дающая противоположный результат прямому вопросу и наоборот. В производственных языках программирования необходимость такой двойственности отсутствует, так как логическое отрицание легко осуществляется средствами самого языка («за скобками»): **не** в КуМире или **not** в Python.

Для языка Python эти команды заменяются на их латинские варианты:

step_forward, turn_left, turn_right, fix_cell для команд-приказов;

way_clean, way_blocked, cell_normal, cell_broken, cell_fixed для команд-вопросов.

В текстовой кодировке, разработанной для библиотеки Moscow Robots, команды Вертуна преобразовались в следующие:

rfwd – *вперёд* (по этой команде робот делает шаг вперёд из середины одной клетки в середину соседней по направлению движения), rright – *направо* (по этой команде робот поворачивается направо на угол 90°, не выходя за пределы клетки, в которой находится), rleft – налево (по этой команде робот поворачивается налево на угол 90°, не выходя за пределы клетки, в которой находится), <rclear> – вперёди свободно?, <!rclr> – вперёди не свободно?, а также некоторое количество команд, специфичных для отдельных типов роботов, как то rfix – закрасить текущую клетку, <rcnor> – клетка нормальная, <rcbro> – клетка поломана, <rcfix> – клетка починена, вместе с их отрицаниями и аналогами для использования в циклах.

Обстановки для роботов представлены текстовыми файлами в формате JSON.

Описание обстановки на примере обстановки для Вертуна (рис. 1):

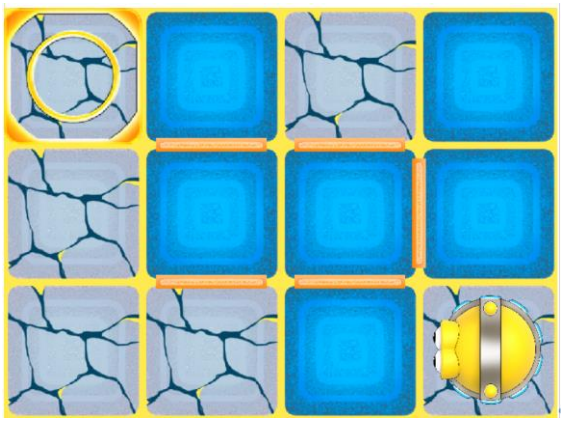

Рис. 1. Обстановка к заданию с Вертуном

```
{
    "robot": {
        "kind": 0,
        "x": 3,
        "y": 2,
        "dir": 3,
        "fpos": [0, 0]
    },
    "field": {
        "sx": 4,
        "sy": 3,
        "cells": [
            ["b", " ", "b", " "],
            ["b", " ", " ", " "],
            ["b", "b", " ", "b"]
        ],
        "vfences": [
            "   ",
            "  |",
            "   "
        ],
        "hfences": [
            " -- ",
            " -- "
        ]
    }
}
```

- Раздел robot содержит
  - целочисленное поле kind, кодирующее тип робота; 0 соответствует Вертуну.
  - целочисленные поля x и y, задающие начальное положение робота; верхней левой клетке соответствует координата x = 0, y = 0.
  - целочисленное поле dir, задающее начальную ориентацию робота; 0 – вверх, 1 – вправо, 2 – вниз, 3 – влево;
  - целочисленную пару координат fpos, задающую требуемое финальное положение робота.
- Раздел field, задающий состояние клеток и других элементов поля
  - целочисленные поля sx, sy, задающие размер поля в клетках по горизонтали и вертикали соответственно;
  - двумерный строковый массив cells, задающий закодированные состояния каждой клетки. Для вертуна элементы кодировки такие: пробел “ ” – нормальная клетка, “b” – разбитая клетка, “p” – закрашенная (отремонтированная) клетка;
  - строковый массив vfences размером sy, задающий в строках длины sx – 1 расположение вертикальных стенок внутри поля, при наличии таковых;
  - строковый массив hfences размером sy – 1, задающий в строках длины sx расположение горизонтальных стенок внутри поля, при наличии таковых.
- В выходных обстановках могут быть дополнительные поля верхнего уровня:
  - целочисленное поле alive, отвечающее за успешное функционирование робота;
  - целочисленное поле ok, отвечающее за успешное выполнение задания.

Обстановка, представленная на рис. 1, формируется библиотекой Moscow Robots с использованием графических файлов формата PNG, хранящихся в составе самой библиотеки.

**Редактор-компилятор программ, использующих библиотеку Moscow Robots**

Редактор-компилятор принято отождествлять с интегрированной средой разработки (англ. – Integrated Development Environment, IDE), которая объединяет в себе как минимум функции редактора программ и компилятора. Удобство такой среды состоит в запуске и(или) компиляции программы (программного проекта) одной кнопкой, когда не требуется покидать редактор и вручную запускать, выходя из среды разработки. В настоящее время такой инструмент допускает фоновую компиляцию программы, например, когда разработчик закончил исправлять один файл проекта и перешел к редактированию другого. Такой подход позволяет сократить время запуска проекта, поскольку большинство файлов с программным кодом уже скомпилировано автоматически. Однако, если вернуться к появлению IDE как производственного средства программиста [8], важнейшей целью являлось то, что отдельные функции среды разработки объединены общей основой: грамматикой языка программирования. То есть программист при написании программы имел поддержку от IDE в виде интерактивной диагностики, поддерживающей программный код в синтаксически-корректном виде. Компиляция программы следовала принципу, известному из управления производством и логистикой, «точно вовремя» (англ. Just-In-Time, JIT).

В 1990-е годы создание учебных интерактивных сред программирования, компьютерных практикумов требовало решения проблемы сократить время на компиляцию практически до нуля, чтобы повысить эффективность обучения. Даже в школьном алгоритмическом языке программирования, часто называемым сейчас языком КуМир [9], изменение всего одной строки в глобальных объявлениях потенциально ведёт к необходимости перекомпиляции сотен строк программы. При разработке этой среды программирования были предприняты специальные методы, такие, например, как обратные ссылки на использованные объекты, которые позволили построить реальный редактор-компилятор, когда процесс компиляции состоял в непрерывной поддержке полного соответствия текста учебной программы скомпилированному коду [10].

Несмотря на взрывной рост производительности вычислительной техники, в системах КуМир, ПиктоМир и ПиктоМир К используется тот же подход и сегодня.

Получив задание на составление программы, использующей библиотеку Moscow Robots, обучаемый может выбрать одну из двух возможностей: составить программу на языке Python с помощью текстового редактора или создать эту программу в пиктограммной форме с помощью прилагаемого редактора программ, использующего безошибочный двумерный синтаксис языка Пикто, описанный в работе [7]. Редактор-компилятор позволяет составить программу для любого Робота, располагая пиктограммы команд этого робота и ряд служебных пиктограмм в клетках таблицы ширины 6 и (предполагаемой) высоты 12. После составления программы ученик может в той же среде преобразовать программу в представление для языка Python, использующее библиотеку Moscow Robots.

Далее рассмотрено решение учебной задачи с использованием языка Пикто на примере задания для Вертуна.

Задание. Робот находится в обстановке, приведенной выше (рис. 1).

Требуется: составить программу управления Вертуном, закрашивающую все сломанные клетки (и только их) и приводящую робота в требуемое финальное положение (0, 0).

В традиционном пиктограммном синтаксисе программа может быть представлена следующим образом:

Эта же программа в безошибочном двумерном синтаксисе представляется в более компактной форме:

Редактор-компилятор переводит её в текстовое представление:

```
(6) [rclear] rfwd <rcbro> rfix
_ rright
```

а также в Python-представление:

```
import moscow_robots as mr


pitcher = 0
memory = 0
fl0 = False
fl1 = False

def proc_main():
    for _ in range(6):
        while mr.path_clear():
            mr.step_forward()
            if mr.cell_broken():
                mr.fix_cell()
        mr.turn_right()
    pass

def proc_A():
    pass

def proc_B():
    pass

def proc_C():
    pass

def proc_D():
    pass

with mr.GameVertun("a.json") as mr:
         proc_main()
```

В составе Moscow Robots есть три неподвижных «робота», имитирующих две целочисленные (pitcher и memory) и две логические (fl0, fl1) переменные. Заготовки для использования этих роботов включаются в любую программу, создаваемую с помощью указанной библиотеки. Аналогично включаются заготовки-заголовки четырёх подпрограмм без параметров proc_A, proc_B,

proc_C и proc_D. Эта программа должна быть выполнена в среде Python, что и будет решением задачи, рис. 2.

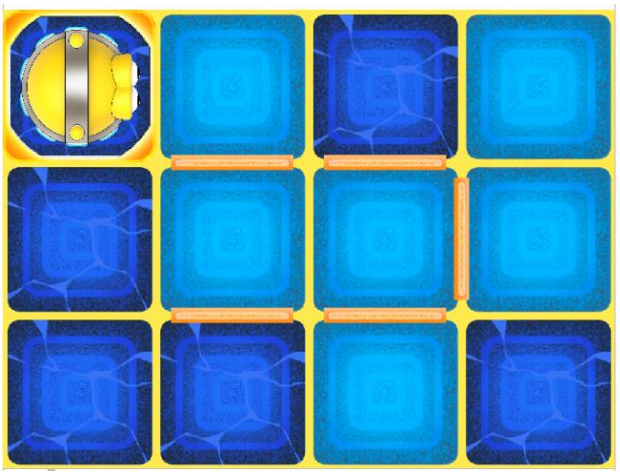

Рис. 2. Результат выполнения программы

Выполнение задания на базе библиотеки Moscow Robots может потребовать реализации некоторого алгоритма движения робота и сбора информации по маршруту с использованием переменных. В этом случае обучаемый с помощью редактора-компилятора сможет сначала составить программу движения по требуемому маршруту, а затем в обычном текстовом редакторе добавить работу с переменными языка Python.

## РЕЗУЛЬТАТЫ ЭКСПЕРИМЕНТАЛЬНЫХ ИССЛЕДОВАНИЙ

Практика работы в начальной школе и на педагогических факультетах университетов показала, что при обучении азам программирования новичков многоязыковый подход более эффективен, чем моноязыковый.

Разработанная модель обучения с использованием библиотеки Moscow Robots апробирована в многоязыковом курсе «Азы программирования» для студентов по сдвоенным специальностям – учитель информатики и учитель начальной школы. Для этого курса был разработан многоязыковый практикум, включающий более 1000 заданий разной степени сложности со встроенными механизмами автоматической проверки правильности решений. Все эти задания касались трех образовательных сред программирования: ПиктоМир -> ПиктоМир-К -> КуМир [11].

Использование дополнительно библиотеки Moscow Robots позволило не только повысить эффективность освоения предмета, но и добавить привлекательность, вариативность содержания и средств обучения этого многоязыкового курса программирования.

В рамках апробации было создано более 30 задач для управления виртуальными роботами ПиктоМир, реализованных в указанной (библиотеке Python).

Включение заданий и библиотеки Moscow Robots в практические занятия было положительно оценено студентами, изучающими курс, позволяя им обрести уверенность в своих способностях при изучении не только образовательных, но и производственных языков программирования.

Алгоритмическая схожесть большинства задач для разных ЦОС позволяет разделить алгоритмические, синтаксические и интерфейсные трудности и сосредоточиться на них в различное время при изучении материала. Это привело к росту эффективности работы студентов при самостоятельном составлении программ в рамках данного курса.

В общей сложности в рамках этого курса, состоящего из 72 полуторачасовых занятий, в 2022–2023 учебном году было предложено около 550 заданий, а в 2023–2024 учебном году – 600 заданий.

**ЗАКЛЮЧЕНИЕ**

Основываясь на результатах проведённого исследования, полученных в 2023–2024 учебном году, можно сделать вывод о том что комбинация пиктограммного и текстового программирования учебных исполнителей на языке Python в многоязыковых системах программирования дает ряд преимуществ обучаемым:

- повышает эффективность работы студентов на вводных курсах программирования, что объясняется тем, что учащиеся знакомятся как с образовательными, так и с профессиональными языками и средами программирования;

- помогает им лучше понимать материал и применять его на практике;

- включение производственной среды программирования, дополненной библиотекой Moscow Robots, повышает мотивацию студентов и побуждает их продолжать изучать программирование.

**Благодарности**

Работа выполнена в рамках темы государственного задания НИЦ «Курчатовский институт» – НИИСИ по теме № FNEF-2024-0001 (1023032100070-3-1.2.1).

**СПИСОК ЛИТЕРАТУРЫ**

1. О внесении изменений в некоторые приказы Министерства просвещения Российской Федерации, касающиеся федеральных образовательных программ начального общего образования, основного общего образования и среднего общего образования // Приказ Министерства просвещения Российской Федерации от 09.10.2024 № 704. Зарегистрирован 11.02.2025. № 81220. http://publication.pravo.gov.ru/document/0001202502120007

2. *Бесшапошников Н.О., Кушниренко А.Г., Леонов А.Г., Райко М.В., Собакинских О.В.* Цифровая образовательная среда «ПиктоМир»: опыт разработки и массового внедрения годового курса программирования для дошкольников // Информатика и образование. 2020. № 10. С. 28–40. https://doi.org/10.32517/0234-0453-2020-35-10-28-40

3. *Кушниренко А. Г., Лебедев Г. В., Сворень Р. А.* Основы информатики и вычислительной техники: учеб. пособие для 10–11-х классов общеобразовательных учреждений. М.: Просвещение; 1990. 224 с. Режим доступа: https:// www.niisi.ru/kumir/books/1.pdf

4. *Бесшапошников Н.О., Леонов А.Г.* Пиктограммный язык программирования «Пикто» // Вестник кибернетики. 2017. № 4 (28). С. 173–180.

5. *Пирс Ч. С.* Что такое знак? // Вестник Томского государственного университета. Философия. Социология. Политология. 2009. № 3. С. 88–95.

6. Раскин Д. Интерфейс: новые направления в проектировании компьютерных систем. М.: Символ-Плюс. 2005.

7. *Кушниренко А.Г., Леонов А.Г., Поликарпов С.А.* Безошибочный двумерный пиктограммный синтаксис в учебной среде программирования для дошкольников // Доклады Российской академии наук. Математика, информатика, процессы управления. 2023. Том 511. № 1. С. 13–19. https://doi.org/10.31857/S2686954323700169

8. *Teitelbaum T., Reps, T.* The Cornell program synthesizer: a syntax-directed programming environment // Communications of the ACM. 1981. Vol. 24. № 9. P. 563–573. https://doi.org/10.1145/358746.358755

9. *Леонов А.Г.* Тенденции объектно-ориентированного программирования в разработке системы КуМир // Программные продукты и системы. 2012. № 4. С. 53.

10. *Леонов А.Г., Эпиктетов М.Г.* Компоненты операционной системы для конструирования педагогических программных продуктов // Аннотированный библиографический указатель «Депонированные научные работы» ВИНИТИ РАН. 1988. № 4278-В88.

11. *Леонов А.Г., Райко М.В.* Знакомство с Цифровой Образовательной Средой «ПиктоМир-К». От знаков к тексту. (Учимся и играем) // Материалы всероссийской научно-практической конференции «STEAM образование: от дошкольника до выпускника ВУЗа». 2022.

## A COMBINATION OF PICTOGRAPHIC AND TEXT PROGRAMMING WHEN CREATING LEARNING EXECUTORS IN PYTHON

**M. V. Rayko**[0000-0002-6448-6471]

*NRC «Kurchatov Institute» – SRISA*, *Moscow*, *Russia*

*Moscow Pedagogical State University, Moscow, Russia*

rayko@niisi.ru

***Abstract***

The article discusses the methodology of developing educational algorithm executors using a combined pictographic-text interface in the Python programming language. The relevance of the research is due to the need to improve approaches to teaching algorithms and programming in the school course of computer science. The author presents the toolkit for creating algorithm executors, combining visibility of icons with text programming capabilities. Particular attention is paid to the practical

aspects of implementation, including the use of builtin Python routines for processing graphics and text.

***Keywords****: algorithm executor, visualization, programming, pictographic interface, Python.*

**REFERENCES**

1. On Amendments to Certain Orders of the Ministry of Education of the Russian Federation Concerning Federal Educational Programs for Primary General Education, Basic General Education, and Secondary General Education // Order of the Ministry of Education of the Russian Federation № 704 dated 09.10.2024. Registered 11.02.2025. № 81220. http://publication.pravo.gov.ru/document/0001202502120007

2. *Besshaposhnikov N.O., Kushnirenko A.G., Leonov A.G., Raiko M.V., Sobakinskikh O.V.* Digital educational environment PictoMir: Experience of development and mass implementation of an annual programming course for preschoolers // Informatics and Education. 2020. № 10. P. 28–40.
https://doi.org/10.32517/0234-0453-2020-35-10-28-40

3. *Kushnirenko A. G., Lebedev G. V., Svoren R. A*. Fundamen tals of informatics and computer engineering: Textbook for 10–11th grade educational institutions. Moscow, Prosveshhenie; 1990. 224 p. (In Russian.) Available at:
https://www. niisi.ru/kumir/books/1.pdf]

4. *Besshaposhnikov N.O., Leonov A.G.* Pictographic programming language Pikto // Proceedings in Cybernetics. 2017. № 4 (28). S. 173–180.

5. *Peirce C.S.* What Is a Sign? // The Essential Peirce. Vol. 2. P. 4–10. Peirce Edition Project. https://peirce.indianapolis.iu.edu/

6. *Raskin J.* The Humane Interface: New Directions for Designing Interactive Systems. 2000. Addison-Wesley Professional.

7. *Kushnirenko A.G., Leonov A.G., Polikarpov S. A.* Error-free 2D pictogrammic syntax in a programming learning environment for preschool children // Proceedings of the Russian Academy of Science. Mathematics, informatics, control processes. 2023. Vol. 511. № 1. P. 13–19. https://doi.org/10.31857/S2686954323700169

8. *Teitelbaum T.,Reps, T.* The Cornell program synthesizer: a syntax-directed programming environment // Communications of the ACM. 1981. Vol. 24. № 9. P. 563–573. https://doi.org/10.1145/358746.358755

9. *Leonov A.G.* Object-oriented programming trends in the development of the KuMir programming environment // Software products and systems. 2012. № 4. P. 53.

10. *Leonov A.G., Epiktetov M.G.* Operating system components for designing educational software products // Annotated bibliographic index "Deposited scientific papers" of VINITI RAS. 1988. № 4278-B88.

11. *Leonov A.G., Rayko M.V.* Introduction to the PictoMir-K digital educational environment. From icons to text. (Learn and play) // Proceedings of the All-Russian scientific and practical conference "STEAM education: from preschooler to university graduate". 2022.

**СВЕДЕНИЯ ОБ АВТОРЕ**

***РАЙКО Миля Вячеславовна***. Научный сотрудник, старший преподаватель кафедры ДПО НИЦ «Курчатовский институт» – НИИСИ, преподаватель Института детства МПГУ.

Области исследований: информатика, информационные системы, робототехнические системы, технологии в образовании.

***Milya V. RAYKO.*** Researcher, senior lecturer of the Department of Additional Professional Education of the NRC «Kurchatov Institute» – SRISA, teacher of the Institute of Childhood of Moscow State University.

Research areas: computer science, information systems, robotic systems, technologies in education.

email: rayko@niisi.ru

ORCID: 0000-0002-6448-6471

*Материал поступил в редакцию 5 апреля 2025 года*