УДК 37.04
DOI 10.17513/snt.39616

ОБУЧЕНИЕ ШКОЛЬНИКОВ ВЫПОЛНЕНИЮ РЕКУРСИВНЫХ АЛГОРИТМОВ С ИСПОЛЬЗОВАНИЕМ СИСТЕМ ПРОГРАММИРОВАНИЯ

[1]Козлов С.В., [2]Быков А.А.

[1]ФГБОУ ВО «Смоленский государственный университет», Смоленск, e-mail: svkozlov1981@yandex.ru;

[2]ФГБОУ ВО «Национальный исследовательский университет «МЭИ», филиал, Смоленск, e-mail: alex1by@mail.ru

Изучение дидактической линии алгоритмизации и программирования в школьном курсе информатики предполагает тесную взаимосвязь теоретических положений с практикой их применения. Это обуславливает всестороннее использование возможностей IT-технологий, в частности инструментальных сред, для решения практических задач. В статье обсуждаются особенности обучения школьников анализу и выполнению рекурсивных алгоритмов для различных исполнителей с помощью функциональных методов систем программирования. Для этого предлагается использовать средства языка программирования Python. Этот современный язык обладает широким набором встроенных системных функций, позволяющих обрабатывать числовые и символьные наборы данных. А классические алгоритмические конструкции языка Python, такие как ветвление и цикл, позволяют понятно и эффективно реализовать запись программы для разных исполнителей. В статье приводятся примеры решения двух заданий базового и повышенного уровней сложности, содержащие рекурсивные описания. В первом примере демонстрируется, как записать программы для алгоритмов, в которых действия для исполнителей определены рекурсивными функциональными соотношениями. Во втором примере показано, как реализовать запись программы, определяющей количество траекторий исполнителя по заданным условиям, с помощью рекурсивных вычислений. Авторами в ходе педагогического эксперимента со школьниками IT-классов доказана эффективность использования методов языка программирования Python для решения заданий анализа выполнения рекурсивных алгоритмов.

Ключевые слова: информатика, программирование, IT-технологии, алгоритм, исполнитель, язык Python, программное приложение, образовательный процесс

TEACHING SCHOOLCHILDREN TO PERFORM RECURSIVE ALGORITHMS USING PROGRAMMING SYSTEMS

[1]Kozlov S.V., [2]Bykov A.A.

[1]Smolensk State University, Smolensk, e-mail: svkozlov1981@yandex.ru;

[2]National Research University "MPEI", branch, Smolensk, e-mail: alex1by@mail.ru

The study of the didactic line of algorithmizing and programming in the school course of computer science involves a close relationship of theoretical positions with the practice of their application. This leads to the comprehensive use of the capabilities of IT technologies, in particular tool environments, for solving practical problems. The article discusses the features of teaching schoolchildren to analyze and execute recursive algorithms for various performers using functional methods of programming systems. To do this, it is proposed to use the Python programming language tools. This modern language has a wide range of built-in system functions that allow you to process numeric and character data sets. And the classical algorithmic constructs of the Python language, such as branching and loop, make it possible to clearly and efficiently implement program recording for different performers. This article provides examples of how to solve two basic and advanced complexity tasks that contain recursive descriptions. The first example demonstrates how to write programs for algorithms in which actions for performers are defined by recursive functional relationships. The second example shows how to implement a program record that determines the number of executor paths according to specified conditions using recursive calculations. During a pedagogical experiment with schoolchildren of IT classes, the authors proved the effectiveness of using Python programming language methods to solve tasks of analyzing the execution of recursive algorithms.

Keywords: computer science, programming, IT-technologies, algorithm, executor, Python language, software application, educational process

Изучение курса информатики в профильной школе предусматривает знакомство учащихся с понятием рекурсии и правилами выполнения рекурсивных алгоритмов [1, 2]. При этом ввиду мощного развития IT-технологий является целесообразным обучение школьников не только теоретическому анализу рекурсивных процедур, но и их программной реализации в одной из выбранных инструментальных сред. Такими средами могут выступать как табличные системы анализа данных, представленные, как правило, электронными таблицами, например MS Excel [3, 4], так и языки программирования, например Python [5, 6]. В то же время для этого демонстрация возможностей использования систем программирования является неотъемлемой задачей педагога профильной школы. В условиях цифровой трансформации системы школьного образования [7, 8] учащиеся классов с IT-направленностью должны свободно

владеть навыками составления программ, содержащих рекурсивные алгоритмы.

Однако в практике обучения учителя школ в большей степени ориентируются на использование IT-технологий обработки числовых данных в электронных таблицах. Это обусловлено тем, что для применения систем программирования школьники должны знать понятие функции, уметь создавать собственные функции, описывать их параметры, алгоритмы выполнения и вызывать их из основной программы [9]. Владение этим набором навыков не предусматривает базовый курс школьной информатики, поэтому многие учащиеся, выбравшие для продолжения обучения IT-класс, зачастую не имеют представления об особенностях записи функций в языках программирования. Они умеют реализовывать простейшие программы, содержащие базовые алгоритмические конструкции. В связи с этим учителя не в полной мере раскрывают возможности современных языков программирования для решения класса задач с использованием рекурсии.

Цель исследования – проверка эффективности объяснения школьникам методов выполнения и анализа рекурсивных алгоритмов для формального исполнителя с помощью инструментов языка программирования Python.

Научная новизна состоит в методологическом использовании инструментов среды программирования Python при решении задач выполнения и анализа рекурсивных алгоритмов, которые составлены для формального исполнителя.

Материалы и методы исследования

В профильном курсе информатики в 10–11 классах предусмотрено знакомство учащихся с понятием подпрограммы и правилами организации ее записи в выбранном для изучения языке программирования. Так, в языке программирования Pascal двумя видами подпрограмм служат процедуры и функции, а, например, в среде C# подпрограммы представляют собой методы с набором входных и выходных аргументов. А в языке Python все подпрограммы реализуются в виде функций. В том числе в любой программной оболочке для записи программ на языке Python, как и во всех других современных системах программирования, предусмотрена возможность обращения функции к самой себе, то есть рекурсия [10]. Таким образом, на всех языках программирования можно создать программу с рекурсивным перебором вычисляемых значений.

Обучение школьников умениям записывать собственные рекурсивные алгоритмы требует от них понимания, что в качестве вычисляемого значения функция будет обращаться к самой себе в одной из точек диапазона изменения независимого аргумента [11]. Других существенных отличительных элементов, увеличивающих сложность программной реализации, запись на выбранном языке программирования рекурсивной функции не имеет. Рассмотрим на примере языка Python, каким образом можно разработать и записать собственную рекурсивную функцию и использовать ее для решения задач. Для этого рассмотрим два примера из системы авторских заданий, аналогичных задачам из ЕГЭ по информатике, которые подразумевают применение рекурсивных алгоритмов для решения поставленной задачи для формального исполнителя.

Так, ЕГЭ по информатике в компьютерной форме, как отражение дидактических линий алгоритмизации и программирования школьного курса, содержит как минимум два задания, в которых возможно использовать в решении рекурсивную форму записи функции. Это задания № 16 и № 23. В первом из них школьнику требуется вычислить значение рекурсивной функции в указанной точке. Во втором задании, относящемся к области динамического программирования, – определить количество программ для формального исполнителя, удовлетворяющих заданным условиям. При этом если в первом из них ему необходимо записать только определяющие функцию условия в выбранной системе программирования, то во втором – сначала интерпретировать команды для исполнителя в систему условий, которые задают сложную рекурсивную функцию, а только потом отразить их в программном коде. Таким образом, для решения рассмотренных заданий учащиеся должны овладеть базовыми навыками записи функциональных соотношений в выбранной среде программирования и научиться решать содержательные задачи для формального исполнителя в заданной системе команд.

Рассмотрим первый пример. Алгоритм вычисления значений функции для формального исполнителя задан следующими соотношениями:

$F(n) = 2$, если $n \le 1$;

$F(n) = 2 \cdot F(n–1) + n – 1$, если $n > 1$.

Чему равно значение функции для $n = 23$?

Представим решение данной задачи на языке программирования Python.

```
def f(n):
    if n <= 1:
        return 2
    else:
        return 2 * f(n – 1) + n – 1
print(f(23))
```

Искомое значение функции будет равно 16777192.

Как видно из представленной записи алгоритма на языке программирования Python, строки решения задачи практически в точности повторяют описанные в условии задания характеристические свойства функции. Сначала определяется наименование функции и число параметров, от которых она зависит. Затем с помощью оператора if создается система ветвлений, после чего записываются действия, выполняемые в соответствии с проверяемыми условиями. Для вычисления и возврата значений функции в программе используется команда return. Вывод результатов вызова работы функции при значении n = 23 осуществляется с помощью команды print. Таким образом, применение инструментов записи рекурсивных алгоритмов на языке Python позволяет перевести задание повышенного уровня сложности при «ручном» решении задания к алгоритмическим действиям базового уровня в системе программирования. При этом если говорить о применении для решения данного типа заданий средств электронных таблиц, то в сравнении и с этим подходом запись алгоритма в среде Python также дает преимущество решения задачи. Оптимальность действий состоит в количестве их повторений, иными словами, решение задания в системе программирования Python определяет эффективность по времени выполнения и меньшую неограниченность ресурсами памяти компьютера.

Рассмотрим второй пример. Исполнитель Вычислитель преобразует число, которое записано на экране с помощью трех команд:

1. Прибавить 1.
2. Прибавить 2.
3. Умножить на 3.

Программа для исполнителя – это последовательность его команд. Сколько существует таких программ, которые преобразуют исходное число 3 в число 24, при этом траектория вычислений программы не содержит число 12 и содержит число 14?

Приведем решение данной задачи на языке программирования Python.

Сначала определим функцию, которая зависит от одного параметра – количества программ в указанной точке траектории. Затем интерпретируем команды исполнителя для их записи на языке Python через структуру ветвлений и обращения функции к себе в предыдущих точках. При этом на первом этапе решения реализуем функцию для вычисления значения в точке 14 траектории программы, а затем на втором этапе модифицируем ее для вычисления в точке 24. При таком подходе потребуется лишь заменить начальную точку движения 3 на точку 14, через которую проходит траектория программы, и обнулить все значения до нее. Это более выгодно, чем реализовывать более громоздкую функцию от двух параметров, начальной и конечной точки траектории программы.

Итак, запись функции для траектории программы из начальной точки 3 в точку 14 будет выглядеть следующим образом.

```
def f(n):
    if n < 3:
        return 0
    elif n == 3:
        return 1
    elif n == 12:
        return 0
    elif n % 3 == 0:
        return f(n – 1) + + f(n – 2) + f(n // 3)
    else:
        return f(n – 1) + f(n – 2)
print(f(14))
```

Теперь изменим в первом условии системы ветвлений оператора if число 3 на число 14 и также поступим со вторым условием. Кроме того, остается выполнить вызов функции не в промежуточной точке 14, а в точке 24. Для этого необходимо поменять число 14 на 24 в операторе вывода значений print. Таким образом, по сравнению с первой записью функции f в программу было внесено три изменения. Отметим, что это была замена числовых значений, а конструкция не претерпела никаких изменений.

```
def f(n):
    if n < 14:
        return 0
    elif n == 14:
        return 1
    elif n == 12:
        return 0
    elif n % 3 == 0:
        return f(n – 1) + + f(n – 2) + f(n // 3)
    else:
        return f(n – 1) + f(n – 2)
print(f(24))
```

Для вычисления итогового результата количества возможных программ, которые удовлетворяют условию задачи, необходимо перемножить два полученных результата. На первом шаге было получено значение 36, а на втором после модификации – 89. Результат можно оформить в программе с помощью команды print(36*89). Ответом будет служить число 3204.

Задания такого типа были использованы в ходе педагогического эксперимента при изучении темы «Рекурсивные алгоритмы для формального исполнителя» для приобретения практических умений

их решения и закрепления полученных навыков. Методами исследования выступили констатирующий и формирующий педагогический эксперименты, которые проводились на основе обобщения современных тенденций развития педагогической мысли. Для анализа результатов экспериментальной деятельности были применены методы математической обработки количественных данных. Гипотеза исследования заключалась в том, что использование базовых инструментов языка программирования Python, в частности множественных ветвлений в записи сложных целочисленных функций, при выполнении и анализе рекурсивных алгоритмов для формального исполнителя повышает эффективность обучения школьников.

Результаты исследования и их обсуждение

Педагогический эксперимент по изучению рекурсивных алгоритмов и обучению их выполнению и анализу школьников проводился на базе двух образовательных учреждений: Смоленского физико-математического лицея при МИФИ и средней школы № 6 г. Смоленска. Учащиеся 10 классов этих учреждений обучаются по физико-математическому направлению, информатика входит в перечень профильных дисциплин.

В экспериментальной работе приняли участие 33 школьника. Педагогический эксперимент проводился в два этапа. На первом этапе – констатирующем педагогическом эксперименте – школьники двух классов изучали особенности анализа и выполнения рекурсивных алгоритмов для формального исполнителя с помощью составления вычислительных таблиц сначала «на бумаге», затем в среде электронных таблиц. Они изучали действия формального исполнителя при выполнении команд «Прибавить число *N*» и «Умножить на число *N*». При этом обсуждались как общие правила выполнения данных команд при произвольном натуральном числе *N*, так и специфика проведения вычислений при конкретных значениях числа *N* от 1 до 3. Школьникам были показаны алгоритмы заполнения таблиц данными рекурсивных вычислений в тетради и унификация проводимых действий с использованием записи формул в электронных редакторах. В заключительной части на данном этапе эксперимента было проведено тестирование учащихся. Предложенный им тест включал десять заданий. Первые пять из них необходимо было выполнить без использования компьютерных средств, а вторые пять – в инструментальной среде электронных таблиц.

На втором этапе – формирующем педагогическом эксперименте – школьники решали подобные задания с использованием средств языка программирования Python. Они изучили правила записи функций в данной среде программирования, такие как определение входных параметров функции, составление алгоритма рекурсивных вычислений с помощью ветвлений, передача вычисленных значений в основную программу и вызов функции. Следует заметить, что на данном этапе школьникам предлагались и более сложные по сравнению с предыдущим этапом эксперимента задачи. Усложнение алгоритмов было связано с проверкой делимости при введении дополнительных параметров, таких как наличие или отсутствие в разрядах числа *N* траектории программы заданных цифр или параметров их четности. В то же время это, безусловно, приводило к увеличению числа проверяемых условий в строках программы, но существенным образом не оказывало влияния на общие принципы записи и реализации рекурсивной функции на языке Python.

В заключение данного этапа педагогического эксперимента учащимся также была предложена система из тестовых заданий. В итоговом тесте школьникам предлагалось выполнить десять заданий в программной оболочке среды программирования Python. Данные промежуточного и итогового тестирований отражались в специализированной оболочке, предназначенной для индивидуального и группового тестирования. Для этого применялись средства математического моделирования [12] и диагностики автоматизированного программного приложения «Advanced Tester» [13]. Заметим, что тренировочные задания, сгенерированные в данной программной оболочке, школьники могли выполнять дистанционно при удаленном доступе [14, 15]. С помощью его функциональных инструментов анализировались ответы, которые давали школьники, и представленные варианты решения заданий. Результаты проведенных итоговых диагностик на констатирующем и формирующем этапах педагогического исследования представлены в табл. 1 и 2.

Качественный анализ условий и результатов эксперимента

Как показывают данные об экспериментальной деятельности, представленные в таблицах, количество школьников в группах с высокими уровнями усвоения знаний существенно увеличилось. В этих группах число учащихся выросло более чем в 1,5 раза.

Таблица 1

Результаты констатирующего этапа педагогического эксперимента

Группа	Число школьников, достигших уровня усвоения знаний			Всего
	Высокий	Повышенный	Базовый	
СФМЛ при МИФИ	4	7	7	18
IT-класс школа № 6	2	6	7	15
Всего	6	13	14	33

Таблица 2

Результаты формирующего этапа педагогического эксперимента

Группа	Число школьников, достигших уровня усвоения знаний			Всего
	Высокий	Повышенный	Базовый	
СФМЛ при МИФИ	9	8	1	18
IT-класс школа № 6	8	6	1	15
Всего	17	14	2	33

При этом произошло перераспределение между группами с повышенным и высоким, а также базовым и повышенным уровнями усвоения знаний. Учащихся, продемонстрировавших знания на базовом уровне, почти не осталось, в обеих группах эту категорию составили по одному школьнику. Это говорит о том, что функциональные возможности современных систем программирования таковы, что позволяют упростить исследование условий, которые определены в задании. Такое положение вещей обусловлено тем, что громоздкие ветвления в условиях заданий тестируются командами инструментальной среды. Особенно данное преимущество ощущается при анализе сложных составных условий, в случаях, когда символы дописываются или заменяются не только справа в получаемом числе, но и в его разрядах слева. Школьник может воспользоваться средствами языка программирования и протестировать алгоритм на различных входных данных. Вычисления за него выполняет компьютерная программа, ему остается правильно описать алгоритм с помощью встроенных команд в простейших случаях или написать собственные функции в более сложных задачах. При этом ряд действий, так же как и при решении заданий «вручную», повторяется, но в различных комбинациях, что в программе отражается либо лишь изменением числовых значений, либо в перестановке строк и не изменяет ее структуры. В совокупности это приводит к тому, что школьники могут изучать решения более сложного класса алгоритмических задач, тем самым повышается их интерес к предметной области программирования. Таким образом, гипотеза исследования о повышении эффективности обучения школьников находит свое доказательство.

Заключение

Итак, проведенная экспериментальная работа подтверждает целесообразность использования функциональных методов систем программирования при решении задач анализа и выполнения рекурсивных алгоритмов для различных исполнителей. Школьники осваивают азы составления рекурсивных алгоритмов на языке Python, пополняя свою фундаментальную базу знаний алгоритмических основ новой информацией о реализации функций, которые обращаются сами к себе в точках предыдущих вычислений. При описанном в педагогическом эксперименте подходе они параллельно изучают теоретический материал и видят его практическое применение. Они совершенствуют свои навыки составления алгоритмов и осваивают умения программной реализации заданий, содержащих рекурсивные обращения. Это расширяет круг IT-технологий, которыми владеют школьники. Учащиеся видят результат применения своих знаний на практике решения прикладных задач, в которых инструменты инструментальных сред помогают им в осваивании фундаментальных навыков профессиональной подготовки по программированию. Таким образом, осуществляется всестороннее изучение предметной области информатики от теории к практике применения с использованием современных достижений компьютерных наук.

Список литературы

1. Моркин С.А. Рекурсия в математике и информатике // Проблемы теории и практики обучения математике: сборник научных работ, представленных на Международную научную конференцию «73 Герценовские чтения». 2020. С. 69–73.

2. Яковлев А.В. Рекурсивные алгоритмы // Вестник образовательного консорциума «Среднерусский университет». Информационные технологии. 2021. № 2 (18). С. 37–42.

3. Абрамов Е.В. Решение практических задач с помощью электронных таблиц Excel // Вестник ВИЭПП. 2018. № 1. С. 186–189.

4. Лобанов А.В., Тишина Е.В., Голубев А.А. Решение задач комплексного анализа средствами электронных таблиц (MS Excel) и реализация вычислений на языке Python // Перспективы развития математического образования в эпоху цифровой трансформации. Материалы II Всероссийской научно-практической конференции. Тверь, 2021. С. 112–118.

5. Рослякова Е.А., Химич А.М. Преимущества использования языка программирования Python при изучении раздела «Алгоритмизация и программирования» в школьном курсе информатики // Современные тенденции развития фундаментальных и прикладных наук: материалы Всероссийской с международным участием научно-практической конференции / Под ред. С.А. Коньшаковой. 2018. С. 148–152.

6. Мартынюк Ю.М., Ванькова В.С., Даниленко С.В. Изучение и использование рекурсивных алгоритмов в подготовке учителя информатики // Чебышевский сборник. 2022. Т. 23. № 5 (86). С. 258–268.

7. Козлов С.В., Быков А.А. Организация обучения в профильной школе в условиях цифровизации системы образования // Аксиологические проблемы педагогики. 2020. № 11. С. 102–110.

8. Тимофеева Н.М. О цифровизации образовательного процесса в условиях полного его переноса в онлайн // Системы компьютерной математики и их приложения. 2021. № 22. С. 388–394.

9. Козлов С.В. Особенности обучения школьников информатике в профильной школе // Концепт. 2014. № 1. С. 31–35. URL: http://e-koncept.ru/2014/14006.htm (дата обращения: 23.05.2023).

10. Мартынюк Ю.М., Ванькова В.С., Даниленко С.В. К вопросу об изучении рекурсивных алгоритмов // Университет XXI века: научное измерение: материалы научной конференции научно-педагогических работников, аспирантов, магистрантов ТГПУ им. Л.Н. Толстого. Тула, 2022. С. 210–212.

11. Пирогов В.Ю. Место рекурсивных алгоритмов в преподавании программирования // Современное образование: методология, теория и практика: материалы Международной научно-практической конференции. Шадринск: Шадринский государственный педагогический университет. 2018. С. 114–117.

12. Козлов С.В., Быков А.А. Особенности изучения междисциплинарных тем школьных курсов математики и информатики с помощью методов математического моделирования // Проблемы современного образования. 2021. № 5. С. 250–261.

13. Козлов С.В., Быков А.А. Применение методов математического моделирования для диагностики знаний школьников // Современные наукоемкие технологии. 2021. № 4. С. 157–162.

14. Киселева О.М. Программные средства поддержки удаленного обучения // Вызовы цифровой экономики: тренды развития в условиях последствий пандемии COVID-19: сборник статей IV Всероссийской научно-практической конференции, приуроченной к Году науки и технологий в России. Брянск, 2021. С. 143–146.

15. Senkina G.E., Timofeeva N.M., Kiseleva O.M. Modernization of traditional educational forms in the context of distance learning // Journal of Higher Education Theory and Practice. 2022. Т. 22, № 3. P. 160–165.