DOI: 10.32517/2221-1993-2024-23-3-5-14

В. С. Попов
победитель Всероссийского конкурса «Эксперименты, исследования, проекты по информатике» в номинации «Эксперименты, исследования, проекты по информатике: теория и практика», Московский государственный технический университет имени Н. Э. Баумана (национальный исследовательский университет), г. Москва, Россия;
Тверской государственный университет, г. Тверь, Россия

ЭМПИРИЧЕСКОЕ ИССЛЕДОВАНИЕ ВРЕМЕНИ ВЫПОЛНЕНИЯ РЕКУРСИВНЫХ ФУНКЦИЙ И РАЗВИТИЕ STEM-КОМПЕТЕНЦИЙ*

Аннотация

В статье представлен подход к изучению рекурсивных функций на углубленном уровне освоения курса информатики в X—XI классах на примере функций вычисления факториала и обобщенных функций Фибоначчи различного порядка. Являясь объектом исследования в рассматриваемой в статье исследовательской работе, эти функции предоставляют усредненные экспериментальные данные времени выполнения, которые в дальнейшем визуализируются, анализируются и аппроксимируются. В процессе такого анализа соотносятся теоретические оценки вычислительной сложности и полученные экспериментальные данные времени выполнения рекурсивных функций в зависимости от переменных аргументов. Показаны примеры двумерной визуализации времени выполнения функции вычисления факториала средствами языка программирования Python и трехмерной визуализации времени выполнения обобщенных функций Фибоначчи различного порядка в редакторе электронных таблиц. Выделены развиваемые STEM-компетенции, изучаемые теории, методы, принципы и концепции в науке, технологиях, инженерии и математике. Объектами научной новизны в данной работе являются: демонстрация нелинейной вычислительной сложности рекурсивного алгоритма вычисления факториала в Python при больших аргументах и выявление причин такого поведения данного алгоритма, основанное на контрпримере; написание рекурсивной обобщенной функции вычисления чисел рядов Фибоначчи с различным порядком как пример реализации принципа DRY; предложенные подходы к углубленному изучению рекурсии и знакомству обучающихся с теорией вычислительной сложности.

Ключевые слова: рекурсия, рекурсивная функция, измерение времени выполнения кода, тайминг кода, сложность алгоритмов, визуализация данных, аппроксимация, компьютерный эксперимент, STEM, компетенция, компетентность.

* Материалы к статье можно скачать на сайте ИНФО: http://infojournal.ru/journals/school/school_03-2024/

Контактная информация

Попов Владислав Сергеевич,
старший преподаватель кафедры «Информационные системы и телекоммуникации», факультет «Информатика и системы управления», Московский государственный технический университет имени Н. Э. Баумана (национальный исследовательский университет), г. Москва, Россия; *адрес:* 105005, Россия, г. Москва, 2-я Бауманская ул., д. 5, стр. 1;
аспирант Института педагогического образования и социальных технологий, Тверской государственный университет, г. Тверь, Россия; *адрес:* 170100, Россия, г. Тверь, ул. Желябова, д. 33;
e-mail: popov_vlad@mail.ru

V. S. Popov
Bauman Moscow State Technical University, Moscow, Russia;
Tver State University, Tver, Russia

EMPIRICAL RESEARCH OF RECURSIVE FUNCTIONS' EXECUTION TIME AND DEVELOPMENT OF STEM COMPETENCIES

Abstract

The article presents an approach to studying recursive functions at an in-depth level of studying informatics course in grades 10–11 using the example of functions for calculating factorial and generalized Fibonacci functions of various orders. Being the object of study in the research work considered in the article, these functions provide average experimental runtime data, which are further visualized, analyzed and approximated. In the process of such analysis, theoretical estimates of computational complexity are compared with the obtained experimental data on the execution time of recursive functions depending on the variable arguments. Examples of 2D visualization of the execution time of the factorial calculation function using the Python programming language and 3D visualization of the execution time of generalized Fibonacci functions of various orders in a spreadsheet editor are shown. Developed STEM competencies, studied theories, methods, principles and concepts in science, technology, engineering and mathematics are highlighted. The objects of scientific novelty in this work are: demonstration of the nonlinear computational complexity of the recursive algorithm for calculating the factorial in Python with large arguments and identifying the reasons for this behavior of this algorithm, based on a counterexample; writing a recursive generalized function for calculating the numbers of Fibonacci series with different orders as an example of the implementation of the DRY principle; proposed approaches to in-depth study of recursion and acquaintance of students with the theory of computational complexity.

Keywords: recursion, recursive function, code execution time, code timing, computational complexity, data visualization, approximation, computer experiment, STEM, competency, competence.

1. Введение

Рекурсия — это одна из тем предмета «Информатика», которые наиболее трудно усваиваются обучающимися. Авторы научно-методических публикаций подчеркивают низкий уровень выполнения учебных заданий по данной теме [2, с. 1887; 13, с. 3], обусловленный неспособностью формально исполнить рекурсивный алгоритм и построить верную последовательность рекурсивных вызовов [2, с. 1887], а также недостаточной сформированностью у обучающихся фундаментальных знаний в области математики, теории алгоритмов и программирования [13, с. 6; 32, с. 465]. Тем не менее этот непривычный для многих способ организации вычислений лежит не только в основе нескольких крайне полезных парадигм языков программирования — как минимум логического [24] и функционального [27] программирования, широко применяемых при создании современных интеллектуальных систем [7, 26], — но и *в самой основе компьютерных вычислений* [8, с. 34–36; 22, с. 179; 23, с. 130; 25, с. 210; 33, с. 189]. Изучение рекурсии на углубленном уровне развивает профессиональный кругозор [27, с. 126], интеллектуальные способности [4, с. 13] и вычислительное мышление [37], формирует информационные и ИКТ-компетенции [3, с. 370; 4, с. 13], а также компетенции обучающихся в области программирования [27, с. 126–130].

Федеральные рабочие программы по информатике для X—XI классов образовательных организаций, обязательные для реализации в школах с 1 сентября 2023 года, в тематическом разделе «Алгоритмы и программирование» включают рекурсию как в виде программного содержания, так и в качестве видов деятельности обучающихся [34, 35]. Даже на базовом уровне изучения информатики в XI классе рассматриваются рекурсивные алгоритмы: ученики должны пояснять сущность рекурсивного алгоритма, находить рекурсивные объекты в окружающем мире, определять результат работы простого рекурсивного алгоритма [34, с. 33–34]. На углубленном уровне дополнительно должны изучаться фракталы как рекурсивные объекты, рекурсивные процедуры и функции, стек рекурсивных вызовов, рекурсивные алгоритмы обхода дерева, динамическое программирование, реализованное через вычисление рекурсивной функции [35]. Также на углубленном уровне изучения информатики федеральная рабочая программа предлагает две практические работы, связанные с рекурсией, — «Рекурсивные подпрограммы» и «Вычисление рекурсивных функций с помощью динамического программирования» [35], и приведенная в данной статье работа может быть их частью. Федеральные рабочие программы как для базового, так и для углубленного уровня содержат введение в теорию сложности: например, раздел «Алгоритмы и программирование» федеральной рабочей программы для углубленного уровня включает в качестве содержания образования оценку сложности вычислений, времени работы программ, асимптотической сложности алгоритмов, а также примеры алгоритмов различной сложности [35, с. 13]. При этом на углубленном уровне изучения информатики ученики должны пояснять понятия «сложность алгоритма», «эффективность алгоритма», давать оценку сложности известных алгоритмов.

На рассматриваемом в данной статье занятии **на основе методологии науки [21] и более частной методологии проведения научного эксперимента [5, 30] предлагается углубленное изучение алгоритмов поиска факториала [9] и *n*-го числа последовательности Фибоначчи [11], являющихся базовыми алгоритмами при знакомстве с рекурсией [25, с. 211].** *Ученики не только вспомнят базовые и узнают нетривиальные примеры рекурсивных и циклических алгоритмов и программ, но также на практике познакомятся с теорией сложности алгоритмов и измерением времени выполнения программного кода, усреднением при проведении научного эксперимента и аппроксимацией, построением двумерных и трехмерных графиков для визуализации полученных данных.* На основе обобщенных функций Фибоначчи различного порядка [14; 36; 40, с. 89–103] обучающиеся познакомятся с одним из главных принципов разработки программного обеспечения — *Don't Repeat Yourself* [41]. *На занятии проявляются не только естественные межпредметные, но и внутрипредметные связи: изучаются рекурсия и теория сложности, экспериментально полученные данные аппроксимируются и выводятся в виде графика вместе с аппроксимирующей кривой, приемы программирования органично дополняются навыками работы в редакторе электронных таблиц.* Рассматривается поистине «постмодернистский компот» [28] из научных методов, приемов, принципов и средств программирования, с которым уже работают и будут работать выпускники будущего [1] для решения самых разнообразных задач.

Некоторые тезисы о вычислениях и их эффективности [12, с. 2] зачастую принимаются учениками на веру, т. е. в качестве очередного *мифа* — и здесь внимательный читатель легко найдет *педагогическое противоречие*, разрешению которого способствует *формирование научной картины мира и соответствующих компетенций*. Через проведение компьютерных экспериментов и последующую корректную обработку и оценку полученных данных ученик пройдет еще несколько шагов к главной *неопозитивистской задаче воспитания*, а именно к *формированию рационально мыслящего человека* [29; 31, с. 171].

Цель занятия: формирование STEM-компетенций обучающихся на примере углубленного изучения рекурсивных функций.

Возраст обучающихся: X—XI класс; занятие рассчитано на школьников, обучающихся на уровне среднего общего образования и имеющих базовые представления о рекурсии.

Описываемая исследовательская работа может быть адаптирована к разным типам учебных занятий в рамках как основного, так и дополнительного образования.

2. Рекурсия: нетривиальные эксперименты, доступные каждому

2.1. Время выполнения рекурсивной функции вычисления факториала: тайминг, усреднение, визуализация, аппроксимация

В качестве первого этапа изучения времени выполнения и вычислительной сложности рекурсивных функций предлагается рассмотреть функцию вычисления факториала.

```
import time
def fact(n):
    if n == 0:
        return 1
    if n > 0:
        return n * fact(n - 1)
for i in range(1001):
    start_time = time.time()
    f = fact(i)
    end_time = time.time()
    print('%.15f' % (end_time - start_time), i, f)
```

Листинг 1

Рекурсивная функция вычисления факториала, *вывод значений факториала для аргумента n от 0 до 1000*, *вычисление времени выполнения рекурсивной функции* показаны в листинге 1.

Для *тайминга кода* использована функция *time* модуля *time*, время выполнения выбранных фрагментов программного кода вычисляется как разность измеренного времени перед выполнением и после выполнения заданного фрагмента (*end_time – start_time*).

В качестве данных при выводе для каждого аргумента *n* показаны:

- время выполнения рекурсивной функции с точностью до 15 знаков после запятой;
- аргумент *i* (параметр *n*) рекурсивной функции вычисления факториала;
- значение рекурсивной функции вычисления факториала для заданного аргумента.

Результат выполнения приведенной программы может оказаться удивительным для учеников старших классов, поскольку измеренное время выполнения рекурсивной функции не возрастает линейно: в качестве примера приведен фрагмент вывода программы, в котором *time*(*fact*(29)) = 0.000997304916382 при меньших значениях времени выполнения для функций с меньшим и бóльшим аргументами.

График зависимости времени выполнения от аргумента рекурсивной функции вычисления факториала также показывает нелинейное увеличение времени выполнения функции с увеличением ее аргумента (рис. 1 и листинг 2). Подобный вид функции можно объяснить целым рядом причин — от многоядерности современных процессоров и многозадачности современных операционных систем до кеширования на уровне процессора и работы интерпретатора.

```
0.000000000000000 27 108888694504183521607680000000
0.000000000000000 28 3048883446117138605015040000000
0.000997304916382 29 884176199373970195454361600000
0.000000000000000 30 265252859812191058636308480000000
0.000000000000000 31 8222838654177922817725562880000000
```

Фрагмент вывода для листинга 1

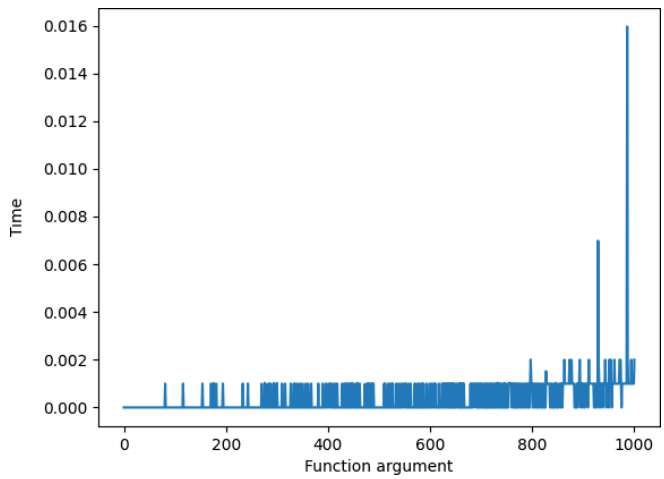


Рис. 1. Время выполнения рекурсивной функции факториала fact(n) в секундах в зависимости от аргумента n

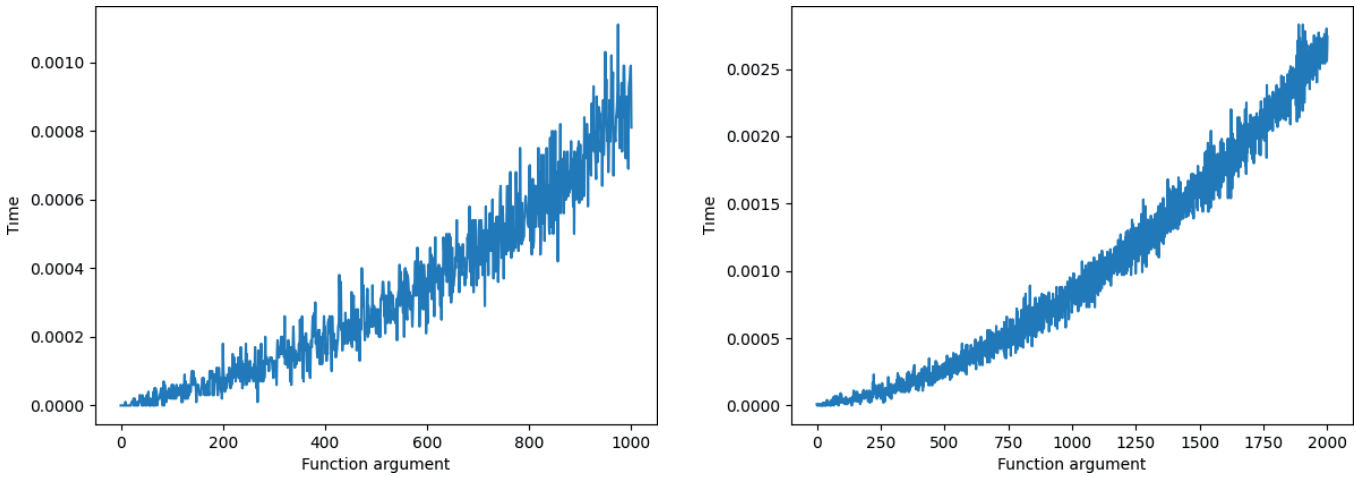
```
import time
import matplotlib.pyplot as plt
def fact(n):
    if n == 0:
        return 1
    if n > 0:
        return n * fact(n - 1)
lst = []
for i in range(1001):
    start_time = time.time()
    f = fact(i)
    end_time = time.time()
    lst.append(end_time - start_time)
    print('%.15f' % lst[-1], i, f)
plt.plot(lst)
plt.xlabel("Function argument")
plt.ylabel("Time")
plt.show()
```

Листинг 2

Первым сущностным проблемным вопросом *является снижение систематической погрешности измерений как степени близости результатов измерений к среднему значению.* Статистически более достоверную и ясную картину функциональной зависимости времени выполнения функции от аргумента функции дает *многократное проведение компьютерного эксперимента*, программный код которого приведен в листинге 2. *Усреднение по ансамблю* [15, с. 17–19], вычисляемое по формуле (1), показано на рисунке 2 и реализовано в листинге 3.

$$\langle T \rangle = \frac{1}{N}(T_1 + T_2 + \ldots + T_N). \quad (1)$$

Доказано, что *вычислительная сложность рассмотренной рекурсивной функции нахождения факториала составляет O(n)* [19]. При этом по графику (см. рис. 2) видно, что экспериментально найденная зависимость носит нелинейный характер. ***Второй важный проблемный вопрос*** *— о рассогласовании экспериментально полученной зависимости и теоретически вычисленной линейной зависимости.* Для ответа на этот вопрос можно предложить множество *гипотез* (здесь ува-



а *б*

Рис. 2. Время выполнения (Time) рекурсивной функции вычисления факториала fact(n) в секундах в зависимости от аргумента n (Function argument):
а) усреднение по ансамблю для N_iterations = 1000; $T_{F(0)}$, …, $T_{F(1000)}$; примерное время выполнения 5 мин;
б) усреднение по ансамблю для N_iterations = 1000; $T_{F(0)}$, …, $T_{F(2000)}$; примерное время выполнения 33 мин

```
import time
import matplotlib.pyplot as plt
import sys
sys.setrecursionlimit(5000)
def fact(n):
    if n == 0:
        return 1
    if n > 0:
        return n * fact(n - 1)
N = 2001
lst = [0] * N
N_iterations = 1000
for k in range(N_iterations):
    for i in range(N):
        start_time = time.time()
        f = fact(i)
        end_time = time.time()
        lst[i] += (end_time - start_time) / N_iterations
    if k % 10 == 0:
        print(k)
plt.plot(lst)
plt.xlabel("Function argument")
plt.ylabel("Time")
plt.show()
print(*lst)
```

Листинг 3

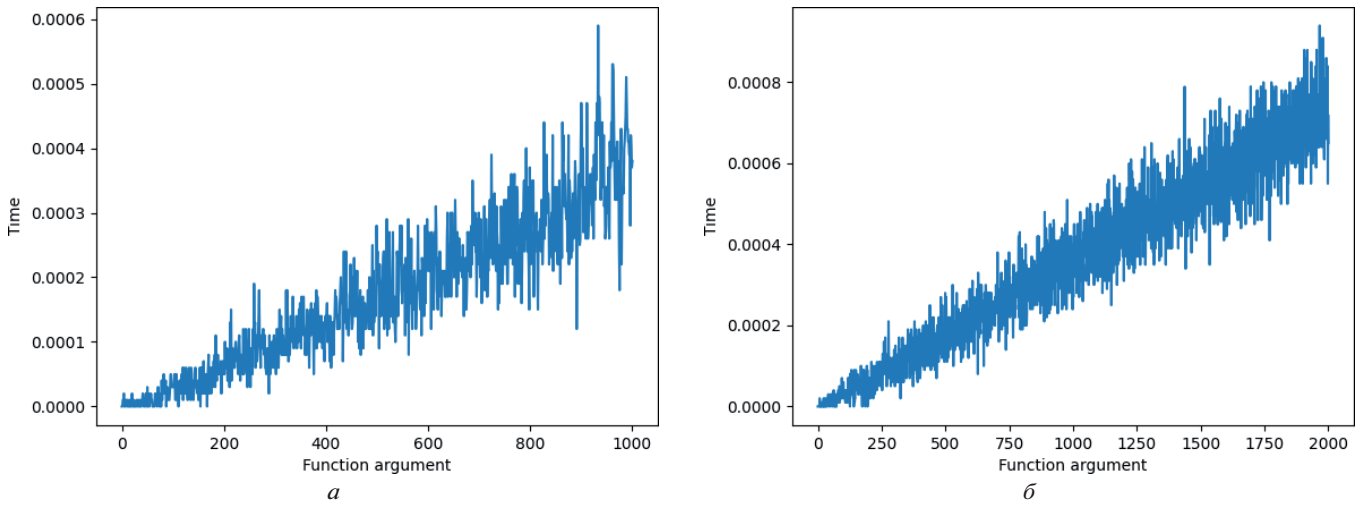


Рис. 3. Время выполнения функции суммирования (измененной рекурсивной функции) в секундах в зависимости от аргумента n:
а) усреднение по ансамблю для Niterations = 1000; $T_{F(0)}$, ..., $T_{F(1000)}$; примерное время выполнения 3 мин;
б) усреднение по ансамблю для Niterations = 1000; $T_{F(0)}$, ..., $T_{F(2000)}$; примерное время выполнения 12 мин

жаемые коллеги остановятся и подумают, чем именно может быть обусловлено подобное рассогласование теории и эмпирики). Следует подчеркнуть, что анализ сложности рекурсивных алгоритмов в общем случае представляет собой весьма нетривиальную задачу [13, с. 5].

В качестве инструмента для ответа на данный вопрос возможно использование *измененной функции вычисления факториала* (или уже-не-факториала), в которой операция умножения заменена на операцию сложения (строка функции `return n * fact(n - 1)` изменена на `return n + fact(n - 1)`). Соответствующие графики зависимости усредненного времени выполнения измененной функции от ее аргумента показаны на рисунке 3. В качестве **дополнительного задания** учитель может предложить ученикам подумать над вопросом, чем эта функция отличается от рекурсивной функции суммирования натуральных чисел.

Сравнивая полученные графики зависимости времени выполнения от аргумента для содержащей операцию умножения функции факториала и содержащей операцию сложения функции суммирования (см. рис. 2 и 3 соответственно), *становится очевидным, что тип вычислительной сложности — полиномиальный или линейный — определяется выбранной операцией*. Ответом на поставленный выше проблемный вопрос является *гипотеза о том, что для вычисления факториала при больших аргументах в языке программирования Python используется длинная арифметика, реализующая алгоритм умножения Карацубы, который, в свою очередь, имеет сложность* $O(n^{\log_2 3})$ [20]. Например, 1000! является числом из 2568 цифр, и «обычная» 32- или даже 64-битная арифметика не справится с подобными вычислениями, что оказывает непосредственное влияние на итоговую сложность функции вычисления факториала.

Полезным является **знакомство с возможностями аппроксимации данных в частности и с математическим моделированием в целом**. Реализация аппроксимации полученных данных времени выполнения рекурсивной функции вычисления факториала в зависимости от аргумента функции с помощью библиотеки *numpy* приведена в листинге 4, а ее результаты представлены на рисунке 4. Получена аппроксимирующая функция:

```
0.0008190808928166944 +
0.001352063425170358 x**1 +
0.0005178936431975428 x**2 -
1.3893495459160382e-05 x**3
```

```
from numpy.polynomial import Polynomial
import matplotlib.pyplot as plt
y_data = list(map(float, open('data-2b.txt').read().split()))
x_data = [i for i in range(0, 2001)]
degrees = (0, 1, 2, 3)
p = Polynomial.fit(x_data, y_data, degrees)
print(p)
y_approx = [p(i) for i in range(2001)]
plt.plot(x_data, y_data, color='gray', label='data')
plt.plot(x_data, y_approx, color='red', label='approx')
plt.xlabel("Function argument")
plt.ylabel("Time")
plt.legend()
plt.show()
```

Листинг 4

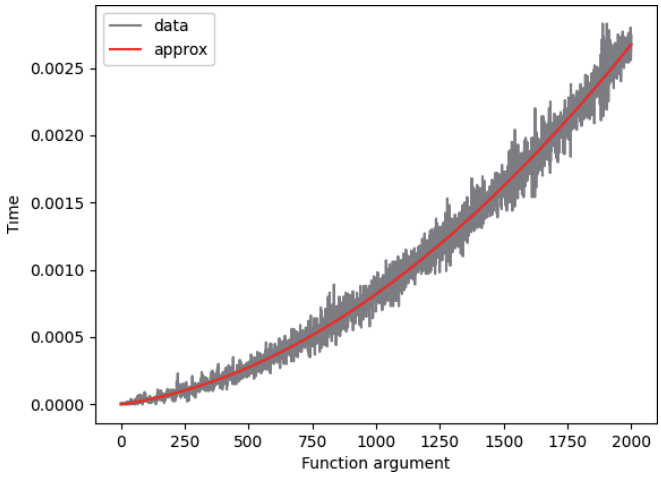


Рис. 4. Исходные данные (соответствующие рис. 2, б) и график аппроксимирующей их функции

2.2. Время выполнения рекурсивных функций Фибоначчи порядка *n*: тайминг, усреднение, визуализация

Полезным является анализ времени выполнения рекурсивных функций, рекуррентная формула в которых для вычисления *n*-го значения опирается не на одно, а на несколько предыдущих значений. В таком случае можно произвести анализ зависимости времени выполнения рекурсивной функции не только от значений ее аргумента, но и от количества прямых вызовов на каждой итерации, получив в качестве результата функцию зависимости времени от аргумента и порядка функции.

Например, для **рекурсивных функций Фибоначчи порядка *n*** можно измерить время выполнения рекурсивных функций:

- Фибоначчи (формула 2) (каждое число является суммой двух предыдущих);
- трибоначчи (формула 3) (каждое число является суммой трех предыдущих);
- тетраначчи (формула 4) (каждое число является суммой четырех предыдущих);
- пентаначчи (каждое число является суммой пяти предыдущих);
- гексаначчи (каждое число является суммой шести предыдущих);
- гептаначчи (каждое число является суммой семи предыдущих);
- октаначчи (каждое число является суммой восьми предыдущих);
- ноначчи (каждое число является суммой девяти предыдущих).

Данные рекуррентные последовательности, иногда также называемые *n-рекуррентностями Фибоначчи* [36, с. 1073], несмотря на продолжительную историю их исследования, до сих пор вызывают научный интерес и являются как объектом научных исследований, так и средством для этих исследований [6, 16, 17, 18, 36].

$$F = \begin{cases} 0 \ if \ n = 0 \\ 1 \ if \ n = 1 \\ F(n-1) + F(n-2) \ if \ n > 1 \end{cases} \quad (2)$$

$$F = \begin{cases} 0 \ if \ n = 0 \ or \ n = 1 \\ 1 \ if \ n = 2 \\ F(n-1) + F(n-2) + F(n-3) \ if \ n > 2 \end{cases} \quad (3)$$

$$F = \begin{cases} 0 \ if \ n = 0 \ or \ n = 1 \ or \ n = 2 \\ 1 \ if \ n = 3 \\ F(n-1) + F(n-2) + F(n-3) + F(n-4) \ if \ n > 3 \end{cases} \quad (4)$$

На примере записи рекурсивных функций для вычисления *n*-го числа Фибоначчи, трибоначчи и последующих функций Фибоначчи других порядков крайне полезным является **демонстрация принципа программирования *Don't Repeat Yourself* (DRY) — «не повторяйся»**, призывающего не дублировать программный код [41].

Листинг 5 демонстрирует обобщенную рекурсивную функцию вычисления чисел Фибоначчи высокого порядка. В функции *fibn* аргумент *n* — номер числа в последовательности для вычисления, аргумент *order* — порядок обобщенной функции Фибоначчи.

Полезной иллюстрацией принципа DRY будет соотнесение рекурсивных функций для вычисления элементов рядов Фибоначчи (2), трибоначчи (3), тетраначчи (4) и обобщенной рекурсивной функции (листинг 5).

Формирование двумерного списка времени выполнения функции вычисления чисел Фибоначчи различного порядка с усреднением по ансамблю показано в листинге 6.

Полученные данные усредненного времени выполнения функции можно легко **представить в виде трехмерного графика в редакторе электронных таблиц**. Для этого следует:

- скопировать данные в редактор электронных таблиц;
- задать подписи строк и столбцов данных (рис. 5);
- построить трехмерный график по полученным данным (рис. 6).

```
def fibn(n, order):
    if n < order - 1:
        return 0
    if n == order - 1:
        return 1
    else:
        sum = 0
        for i in range(order):
            sum += fibn(n - i - 1, order)
        return sum
```

Листинг 5

```
A = []
for i in range(10):
    A.append([0] * 28)
N_iterations = 10
for k in range(N_iterations):
    for order in range(2, 10):
        for i in range(28):
            start_time = time.time()
            f = fibn(i, order)
            end_time = time.time()
            A[order][i] += (end_time - start_time) / N_iterations
for i in range(10):
    print(*A[i])
```

Листинг 6

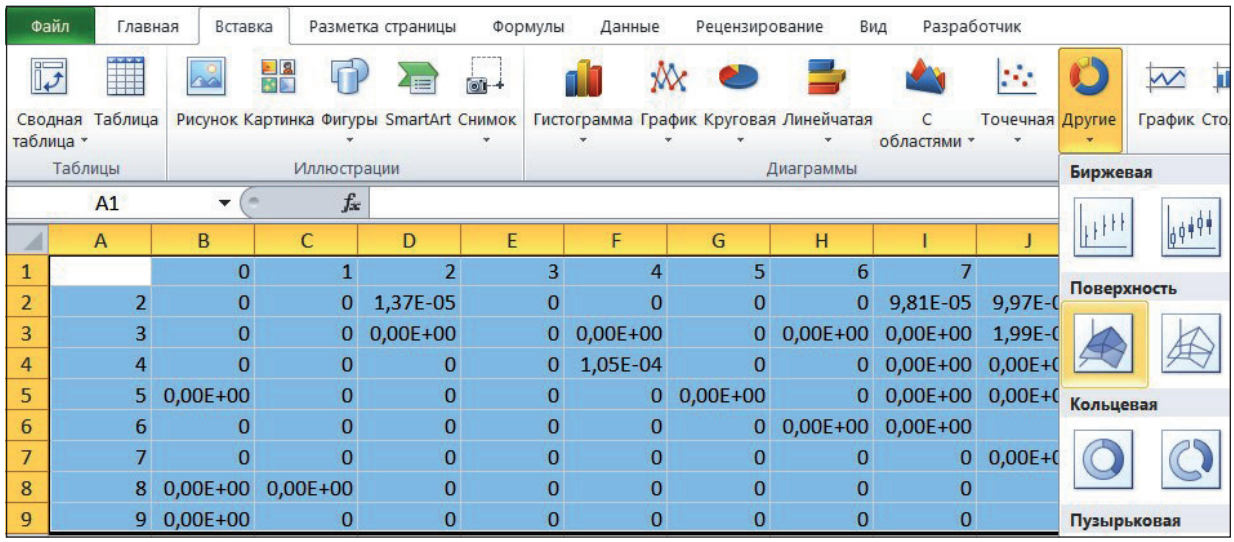


Рис. 5. Данные и подписи данных в электронной таблице Excel, вставка диаграммы ***Поверхность*** *для создания 3D-графика*

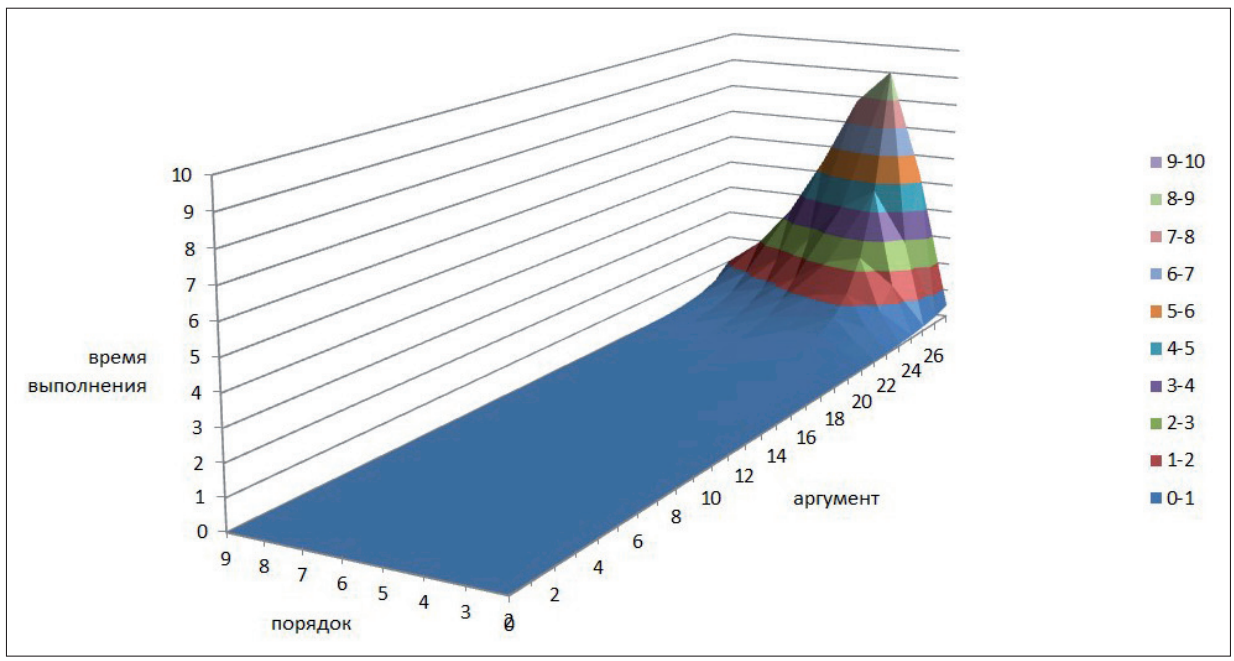


Рис. 6. Трехмерный график зависимости времени выполнения в секундах функций Фибоначчи различных порядков от порядка и аргумента функции

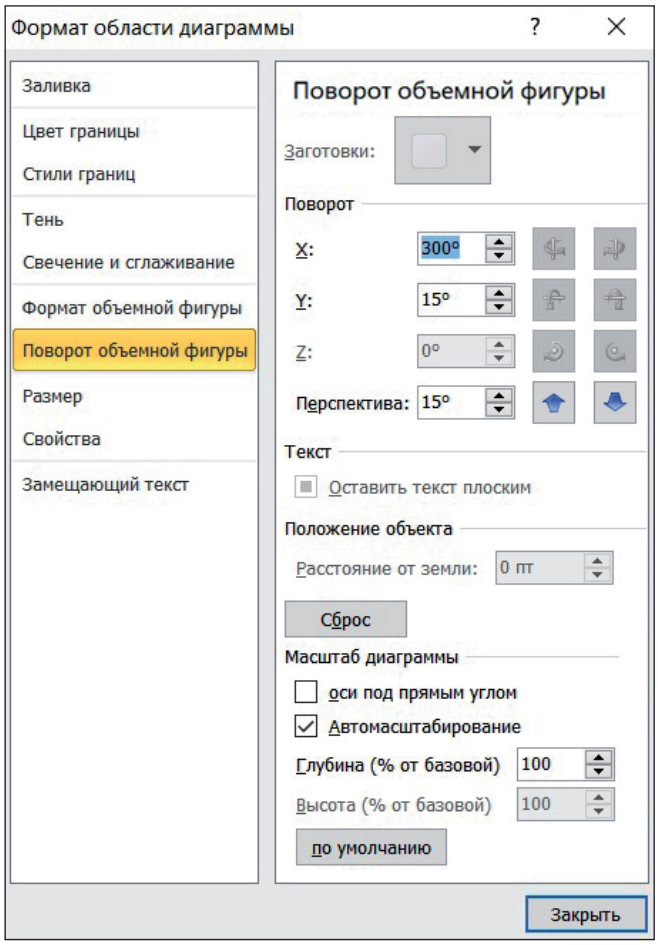


Рис. 7. ***Формат области диаграммы, Поворот объемной фигуры***

В зависимости от редактора электронных таблиц может потребоваться замена разделителя целой и дробной частей — в Microsoft Excel разделитель точку следует заменить на запятую, для чего следует воспользоваться сочетанием клавиш Ctrl+H или командой *Найти и выделить*, *Заменить*. Для более наглядного отображения трехмерного графика после щелчка правой кнопкой мыши в контекстном меню графика следует выбрать пункт *Поворот объемной фигуры*, после чего в открывшемся окне (рис. 7) изменить значения *X*, *Y*.

В качестве еще одного ***нетривиального проблемного вопроса*** *предлагаем рассмотреть причины, по которым функция тетраначчи (функция Фибоначчи порядка 4) с аргументами 13–27 выполняется дольше всех рассмотренных обобщенных функций Фибоначчи как с более низким, так и с более высоким порядком для тех же аргументов.*

3. STEM-компетенции

В процессе выполнения шагов рассмотренного исследования ученики изучают не только методы программирования, но также значимые научные объекты, явления, подходы и методы, относимые в целом к зонтичному термину *STEM* (*англ.* Science, Technology, Engineering and Mathematics — наука, технология, инженерия и математика) [38, 39].

К рассмотренным *научным (Science)* явлениям, методам и даже теориям относятся:

- рекурсия;
- усреднение по ансамблю;
- линейный и полиномиальный рост;
- вычислительная сложность;
- аппроксимация;
- научный эксперимент, обработка и анализ его результатов;
- научная гипотеза.

Можно выделить следующие **аспекты, относящиеся к *технологии программирования и программной инженерии (Technology, Engineering)*:**

- рекурсивные функции;
- циклы и вложенные циклы;
- тайминг кода (вычисление времени выполнения фрагмента программного кода) и его усреднение;
- визуализация данных (на примере построения графиков);
- длинная арифметика;
- реализация аппроксимации;
- одномерные и двумерные списки;
- принцип программирования Don't Repeat Yourself.

Также в рассмотренном программном коде использовались такие возможности языка программирования Python, как:

- генераторы списков;
- индексация элементов с конца списка;
- создание одномерных и двумерных списков;
- вывод вещественных чисел с заданной точностью;
- чтение из текстового файла;
- оператор распаковки «*»;
- изменение глубины рекурсивных вызовов;
- накопление значений в цикле.

К рассмотренным *математическим (Mathematics)* объектам относятся:

- функция вычисления факториала;
- обобщенные функции Фибоначчи различного порядка — от трибоначчи до ноначчи;
- аппроксимирующая функция;
- усреднение;
- 2D- и 3D-графики функций.

По мнению автора, всестороннее формирование STEM-компетенций и декомпозиция компетенций по рассмотренным областям (Science, Technology, Engineering and Mathematics) в углубленном изучении информатики позволяют обучающимся получить предпрофессиональные навыки и глубокие знания, сформировать понимание изучаемых объектов и явлений и положительные ценностные отношения к научному методу, научной картине мира и научному познанию.

4. Дальнейшие исследования

В качестве направлений дальнейшего исследования рекурсии для углубленного уровня изучения информатики можно предложить:

- изучение мемоизации [10] и ее влияния на скорость вычислений;
- нахождение максимального аргумента для линейной сложности вычисления факториала в Python;

- построение трехмерного графика зависимости времени выполнения обобщенных функций Фибоначчи различного порядка при бóльших аргументах;
- построение трехмерных графиков в Python и аппроксимацию полученных трехмерных данных;
- изучение различных видов рекурсии, в том числе с использованием логических и функциональных языков программирования;
- проведение исследований в однозадачных операционных системах.

5. Выводы

Предложенная в статье исследовательская работа способствует:

- углубленному изучению рекурсии;
- формированию базовых представлений о теории сложности и визуализации данных;
- приобретению практического опыта научного эксперимента;
- закреплению ранее пройденного учебного материала.

На примере изучения рекурсивных функций, элементов теории вычислительной сложности и компьютерных экспериментов выделены формируемые компетенции по составляющим STEM: наука, технологии, инженерия, математика.

После изучения предложенного материала и самостоятельного проведения предложенного эмпирического исследования ученики смогут измерять время выполнения фрагментов программного кода, в том числе соотносить эффективность алгоритмов: как было показано в данной статье, способности вычислить время выполнения фрагмента программы и оценить временну́ю сложность алгоритма взаимосвязаны.

Следующим результатом обучения может стать еще один шаг к развитию способности создания более эффективного программного кода — как через уменьшение его вычислительной сложности, так и через уменьшение его избыточности посредством использования принципа Don’t Repeat Yourself.

Предложены гипотезы, поставлены проблемные вопросы, указаны возможные темы для дальнейших исследований в данной и смежных областях научного знания.

Благодарности

Автор выражает благодарность старшему преподавателю кафедры «Информационные системы и телекоммуникации» МГТУ им. Н. Э. Баумана Д. А. Видьманову за плодотворное обсуждение вычислительной сложности упомянутых в статье рекурсивных функций факториала и суммирования, а также методистам ИРПО МГПУ Е. А. Алефиренко и Л. Ю. Черницыной за полезные дискуссии о федеральных рабочих программах по информатике.

Список источников

1. Атлас профессий будущего / Н. Ю. Анисимов, Л. М. Гохберг, Г. О. Греф, Н. В. Дудина, С. В. Черногорцева, Н. А. Шматко и др.; НИУ ВШЭ; ПАО «Сбербанк». Вып. 2. М.: НИУ ВШЭ, 2021. 240 с. https://sberuniversity.ru/upload/edutech/reports/Atlas_future_professions.pdf

2. *Барабаш Г. И., Старикова М. Е.* Рекурсивные задачи в ЕГЭ по информатике // Информатика: проблемы, методы, технологии. Материалы XX Международной научно-методической конференции (Воронеж, 13–14 февраля 2020 года) / под ред. А. А. Зацаринного, Д. Н. Борисова. Воронеж: Научно-исследовательские публикации (ООО «Вэлборн»), 2020. С. 1887–1890. EDN: VETFVK.

3. *Богданова В., Кириак Л.* Изучение рекурсивных алгоритмов с позиции STEAM в среднем образовании // Materialele Conferinței Științifice Internaționale «Abordări inter/transdisciplinare în predarea științelor reale (concept STEAM)», ediția a 2-a (Chișinău: UST, 28–29 Octombrie 2022). С. 366–371. http://dir.upsc.md:8080/xmlui/bitstream/handle/123456789/4962/Conf-Abordari-predare-stiinte-reale-STEAM-2022-p366-371.pdf?sequence=1&isAllowed=y

4. *Богданова В. А., Тодорова Ю. Г., Чумак Л. В.* Из опыта проведения междисциплинарного внеаудиторного мероприятия «Удивительный мир рекурсии» // Инновационные подходы в образовании. Научно-методический семинар (30 марта 2023 года) / под общ. ред. А. Л. Цынцарь, Е. В. Гатанюк. Тирасполь: Изд-во Приднестр. ун-та, 2023. С. 13–15.

5. *Вилков А. Н.* Курс лекций «Методология проведения научного эксперимента». М.: МГТУ им. Н. Э. Баумана, 2012. 33 с.

6. *Городов А. А., Городова Л. В., Кузнецов А. А.* Моделирование сложных процессов при помощи авторегрессии // Вестник Сибирского государственного аэрокосмического университета им. академика М. Ф. Решетнева. 2014. № 5 (57). С. 57–61. EDN: TYWGSF.

7. *Девятков В. В., Лычков И. И., Наунг М. Т.* Прототипирование верификации поведения интеллектуальных агентов в языке логического программирования ПРОЛОГ: учебное пособие. М.: Изд-во МГТУ им. Н. Э. Баумана, 2021. 56 с. EDN: EHIBSP.

8. *Дегтяренко В. А.* Теория алгоритмов в школьном курсе информатики // Актуальные проблемы преподавания информатики и информатизации образовательного процесса в учреждениях основного и дополнительного образования. Сборник научных трудов. Комсомольск-на-Амуре: Амурский гуманитарно-педагогический государственный университет, 2021. С. 33–39. EDN: RUPUKH.

9. *Деза Е. И.* Научная составляющая и методические возможности понятия «факториал» // Проблемы теории и практики инновационного развития и интеграции современной науки и образования. Материалы IV международной междисциплинарной конференции (Москва, 15 февраля 2023 года). М.: Государственный университет просвещения, 2024. С. 23–26. EDN: RNVIGF.

10. *Долинский М. С.* Ускорение рекурсивных решений при помощи мемоизации // Информатика в школе. 2022. № 6. С. 55–67. EDN: PSWCPK. DOI: 10.32517/2221-1993-2022-21-6-55-67.

11. *Дроздюк А. В.* Фибоначчи, его числа и кролики. Торонто: Choven, 2010. 145 с.

12. *Дудина И. П.*, *Коновалова А. Д.* Подготовка учащихся к единому государственному экзамену по информатике (по разделу «Алгоритмизация и программирование») // Инновации. Наука. Образование. 2019. № 10 (11). С. 1–6. EDN: BMGFRK.

13. *Зуева И. Ю.* Рекурсивные алгоритмы — инструмент развития стиля мышления. (ЕГЭ по информатике) // Информатика: проблемы, методология, технологии. Сборник материалов XVII международной научно-методической конференции (Воронеж, 9–10 февраля 2017 года). Т. 5. Воронеж: ООО «Вэлборн», 2017. С. 3–6. EDN: YJOQNX.

14. *Иванов О. А.* Алгебраические и комбинаторные методы исследования обобщений чисел Фибоначчи // Актуальные вопросы современных математических и естественных наук. Сборник научных трудов по итогам международной научно-практической конференции (Екатеринбург, 10 марта 2016 года). Вып. III. Екатеринбург: Инновационный центр развития образования и науки, 2016. С. 13–15. EDN: VPZFVR.

15. *Киттель Ч.* Элементарная статистическая физика / пер. с англ. Л. А. Шубиной, под ред. С. В. Вонсовского. М: Изд-во иностранной литературы, 1960. 288 с.

16. *Куликов В. Л., Олехова Е. Ф., Оселедец В. И.* Замечания об абсолютной непрерывности меры Эрдеша для золотого сечения, числа трибоначчи и марковской цепи // Современная математика и концепции инновационного математического образования. 2022. Т. 9. № 1. С. 66–76. EDN: FITMNB. DOI: 10.54965/24129895_2022_9_1_66.

17. *Куликов В. Л., Олехова Е. Ф., Оселедец В. И.* Об абсолютной непрерывности меры Эрдёша для золотого сечения, числа трибоначчи и марковских цепей второго порядка // Теория вероятностей и ее применения. 2024. Т. 69. Вып. 2. С. 335–353. EDN: YWCGER. DOI: 10.4213/tvp5628.

18. *Куликов В. Л., Олехова Е. Ф., Оселедец В. И.* Сингулярность меры Эрдеша для 2-марковских цепей и числа 4-наччи // Современная математика и концепции инновационного математического образования. 2023. Т. 10. № 1. С. 77–87. EDN: LOWLOV. DOI: 10.54965/24129895_2023_10_1_77.

19. *Ладиков А. В.* Улучшенный алгоритм вычисления факториала // Математические заметки. 2008. Т. 83. Вып. 6. С. 857–863. DOI: 10.4213/mzm4837. https://www.mathnet.ru/links/2ebe8133b2d42988b39aac04eee34917/mzm4837.pdf.

20. *Лапаев А. О.* Реализация алгоритма Карацубы и оценки сложности // Вестник Тамбовского университета. Серия: Естественные и технические науки. 2008. Т. 13. № 1. С. 142–143. EDN: IIWFVP.

21. *Лебедев С. А.* Методология научного познания: учебное пособие для вузов. М.: Юрайт, 2024. 153 с. https://urait.ru/bcode/537439

22. *Локтев Д. А., Видьманов Д. А.* Учебное пособие для поступающих в вузы. Информатика: учебное пособие. 2-е изд., испр. М.: Изд-во МГТУ им. Н. Э. Баумана, 2023. 196 с.

23. *Лорсанова З. М.* Использование рекурсивных методов в решении алгоритмических задач // Актуальные научные исследования в современном мире. 2019. № 12-4(56). С. 129–131. EDN: MJKRKB.

24. *Лычков И. И.* Применение логического программирования для решения олимпиадных задач по информатике // Преподавание информационных технологий в Российской Федерации. Сборник научных трудов Двадцать второй открытой Всероссийской конференции (Тверь, 16–17 мая 2024 года). [В печати.]

25. *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, 14–30 ноября 2022 года). Тула: Тульский государственный педагогический университет им. Л. Н. Толстого, 2022. С. 210–212. EDN: KYOWEN.

26. *Морозов А. А., Вайш А., Полупанов А. Ф., Анциперов В. Е., Лычков И. И., Алфимцев А. Н., Девятков В. В.* Разработка исследовательской программной платформы для параллельного объектно-ориентированного логического программирования интеллектуального видеонаблюдения // Интеллектуализация обработки информации. 2014. Т. 10. № 1. С. 130–131. EDN: YLJMDV.

27. *Никифоров П. В., Вяткин А. А.* Инновационный подход к преподаванию функционального программирования (на примере F#) // Инновации в науке и практике. Сборник статей по материалам XIII международной научно-практической конференции (Барнаул, 26 декабря 2018 года). Ч. 5 (5). Уфа: ООО «Дендра», 2018. С. 126–131. EDN: ZAPBGP.

28. *Огурцов А. П.* Постмодернизм в контексте новых вызовов науки и образования // Вестник Самарской гуманитарной академии. Выпуск «Философия. Филология». 2006. № 1(4). С. 3–27.

29. *Подласый И. П.* Педагогика: Новый курс: учебник для студентов высших учебных заведений. В 2 кн. М.: Гуманит. изд. центр ВЛАДОС, 2003. Кн. 1: Общие основы. Процесс обучения. 576 с.

30. *Пономарев А. Б., Пикулева Э. А.* Методология научных исследований: учебное пособие. Пермь: Изд-во Пермского национального исследовательского политехнического университета, 2014. 186 с.

31. Приоритетные направления современной психологии и педагогики: коллективная монография / Л. А. Абросимова-Романова, А. В. Антоновский, Е. В. Астапенко и др. Тверь: Тверской государственный университет, 2023. 223 с. EDN: JECRVO.

32. *Симанева Т. А., Топченко Р. К.* Проектирование и реализация цифрового образовательного ресурса для изучения темы «Рекурсия» в школьном курсе информатики // Естественные и гуманитарные науки в современном мире. Материалы Международной научно-практической конференции (Орел, 13–15 мая 2021 года). Орел: Орловский государственный университет имени И. С. Тургенева, 2021. С. 465–471. EDN: JFOIMR.

33. *Смирнова А. В.* Особенности изучения темы «Рекурсия» в школьном курсе информатики // Современные проблемы и перспективные направления инновационного развития науки. Сборник статей Международной научно-практической конференции (Томск, 25 апреля 2016 года). В 4 ч. Ч. 2. Томск: ООО «Аэтерна», 2016. С. 188–193. EDN: VWTMNR.

34. Федеральная рабочая программа среднего общего образования. Информатика (базовый уровень) (для 10–11 классов образовательных организаций). М.: Институт стратегии развития образования, 2023. 38 с. https://edsoo.ru/wp-content/uploads/2023/08/21_ФРП-Информатика_10-11-классы_база.pdf

35. Федеральная рабочая программа среднего общего образования. Информатика (углубленный уровень) (для 10–11 классов образовательных организаций). М.: Институт стратегии развития образования, 2023. 52 с. https://edsoo.ru/wp-content/uploads/2023/08/22_ФРП_Информатика-10-11-классы_угл.pdf

36. *Чернов В. М.* Фибоначчи, трибоначчи, ..., гексаначчи и параллельная безошибочная машинная арифметика // Компьютерная оптика. 2019. Т. 43. № 6. С. 1072–1078. EDN: AMUIMF. DOI: 10.18287/2412-6179-2019-43-6-1072-1078.

37. *Шваб А. Г.* Формирование компьютерного (вычислительного) мышления // Наука сегодня: глобальные вызовы и механизмы развития. Материалы международной научно-практической конференции (Вологда, 28 апреля 2021 года). Вологда: ООО «Маркер», 2021. С. 57–58. EDN: DHXEWR.

38. *Breiner J. M., Sheats Harkness S., Johnson C. C., Koehler C. M.* What is STEM? A discussion about conceptions of STEM in education and partnerships // School Science and Mathematics. 2012. Vol. 112 (1). P. 3–11. DOI:10.1111/j.1949-8594.2011.00109.x.

39. *Brown R., Brown J., Reardon K., Merrill C.* Understanding STEM: current perceptions // Technology and Engineering Teacher. March 2011. P. 5–9.

40. *Gardner M.* The second scientific American book of mathematical puzzles and diversions. Chicago: The University of Chicago Press, 1987. 254 p.

41. *Thomas D., Hunt A.* The Pragmatic Programmer: Your journey to mastery. 20th Anniversary Edition. Addison-Wesley Professional, 2019. 352 p.