ISSN 2221-1993 • **Informatics in School** • 2026 • Volume 25 No 1

DOI: 10.32517/2221-1993-2026-25-1-68-72

И. А. Быкова
Ярославский государственный педагогический университет им. К. Д. Ушинского, г. Ярославль, Россия;
Образовательный комплекс № 10, г. Ярославль, Россия

М. А. Заводчиков, Н. Д. Елисеева
Ярославский государственный университет им. П. Г. Демидова, г. Ярославль, Россия

СПОСОБЫ РЕШЕНИЯ ЗАДАЧ НА ПЕРЕЧИСЛЕНИЕ СЛОВ, СОСТАВЛЕННЫХ ИЗ СИМВОЛОВ ЗАДАННОГО АЛФАВИТА

Аннотация

В последнее время задачи по теме «Комбинаторика» перестали быть прерогативой олимпиадной подготовки школьников. С понятиями и методами комбинаторики школьники встречаются в курсах информатики, математики, вероятности и статистики. Задание 8 ЕГЭ по информатике предполагает наличие у учащихся умений перечислять и подсчитывать конфигурации из объектов конечного множества, построенных в соответствии с некоторыми правилами. В статье описаны различные методы решения задач данного типа: вложенные циклы, использование библиотеки itertools, рекурсивных функций и рекурсивных функций-генераторов. Последние два метода позволяют школьникам получить представление о итерируемых объектах, лучше понять устройство объектов-генераторов. Решение одной задачи несколькими способами развивает критичность мышления, положительно сказывается на создании внутрипредметных связей в изучаемом материале, способствует более глубокому усвоению материала.

Ключевые слова: комбинаторика, генерация комбинаторных объектов, функции-генераторы, итерируемые объекты, ЕГЭ по информатике.

Информация об авторах

Быкова Ирина Альбертовна, старший преподаватель кафедры теории и методики обучения информатике, физико-математический факультет, Ярославский государственный педагогический университет им. К. Д. Ушинского, г. Ярославль, Россия; учитель информатики, центр образования — школа № 26, образовательный комплекс № 10, г. Ярославль, Россия; *e-mail:* i.bukova@yandex.ru

Заводчиков Михаил Александрович, канд. физ.-мат. наук, доцент кафедры алгебры и математической логики, математический факультет, Ярославский государственный университет им. П. Г. Демидова, г. Ярославль, Россия; *e-mail:* zav-mikhail@yandex.ru

Елисеева Надежда Дмитриевна, канд. физ.-мат. наук, доцент кафедры математического моделирования, математический факультет, Ярославский государственный университет им. П. Г. Демидова, г. Ярославль, Россия; *e-mail:* n.bykova90@gmail.com

I. A. Bikova
Yaroslavl State Pedagogical University named after K. D. Ushinsky, Yaroslavl, Russia;
Educational Complex 10, Yaroslavl, Russia
M. A. Zavodchikov, N. D. Eliseeva
P. G. Demidov Yaroslavl State University, Yaroslavl, Russia

METHODS OF SOLVING PROBLEMS FOR ENUMERATING WORDS MADE UP OF CHARACTERS OF A GIVEN ALPHABET

Abstract

Recently, tasks on the theme of "Combinatorics" have ceased to be the prerogative of Olympiad training for schoolchildren. Students learn the concepts and methods of combinatorics in informatics, mathematics, probability, and statistics courses. The task 8 of the Unified State Exam in informatics assumes that students have the ability to enumerate and calculate configurations from objects of a finite set constructed in accordance with certain rules. The article describes various methods for solving these problems: nested loops, using the itertools library, using recursive functions and recursive generator functions. The last two methods allow students to get an idea of iterated objects and better understand the structure of generator objects. Solving one problem in several ways develops criticality of thinking, has a positive effect on the creation of intrasubject connections in the studied material, and promotes deeper learning of the material.

Keywords: combinatorics, generation of combinatorial objects, generator functions, iterable objects, Unified State Exam in informatics.

Information about the authors

Irina A. Bikova, Senior Lecturer at the Department of Theory and Methods of Teaching Informatics, Faculty of Physics and Mathematics, Yaroslavl State Pedagogical University named after K. D. Ushinsky, Yaroslavl, Russia; an informatics teacher, Education Center — School 26, Educational Complex 10, Yaroslavl, Russia; *e-mail:* i.bukova@yandex.ru

Mikhail A. Zavodchikov, Candidate of Sciences (Physics and Mathematics), Associate Professor at the Department of Algebra, Number Theory and Mathematical Logic, Faculty of Mathematics, P. G. Demidov Yaroslavl State University, Yaroslavl, Russia; *e-mail:* zav-mikhail@yandex.ru

Nadezhda D. Eliseeva, Candidate of Sciences (Physics and Mathematics), Associate Professor at the Department of Mathematical Modeling, Faculty of Mathematics, P. G. Demidov Yaroslavl State University, Yaroslavl, Russia; *e-mail:* n.bykova90@gmail.com

ISSN 2221-1993 • **Информатика в школе** • 2026 • Том 25 № 1

1. Введение

В настоящее время наиболее часто при обучении программированию используют язык Python. Данный язык имеет большое количество библиотек и встроенных функций, которые школьники используют при решении задач, порой не задумываясь, с какими объектами они работают и как эти объекты функционируют. Например, школьники часто используют функцию range, применяют для решения задач функции *product*, *permutations*, *combinations* библиотеки *itertools*, которые возвращают объект-генератор. Стоит отметить, что осознанное использование цикла *for* учащимися требует от них понимания того факта, что этот цикл перебирает элементы итерируемого объекта. Согласно федеральной рабочей программе по информатике [11], на углубленном уровне не предполагается изучение понятий «итерируемый объект» и «объект-генератор», однако сформировать указанные представления необходимо. Это можно сделать при изучении предусмотренных программой тем школьного курса информатики, например, при рассмотрении способов генерации всех слов в некотором алфавите, удовлетворяющих заданным ограничениям.

Задачи по теме «Комбинаторика» являются традиционными для олимпиад по информатике и математике [2, 7, 13, 14]. В последнее время школьники сталкиваются с понятиями и алгоритмами комбинаторики в школьных курсах математики [12] и информатики [11]. В задании 8 ЕГЭ по информатике учащиеся должны продемонстрировать умение перечислять и подсчитывать конфигурации из объектов конечного множества, построенные в соответствии с некоторыми правилами [4].

Обычно, прочитав условие, школьники сразу применяют известные им алгоритмы, а решение задачи заканчивают, получив ответ. Тем не менее бывает полезно продолжить работу над анализом условия и способов решения задачи уже после получения ответа. Как отмечал А. А. Окунев, такая работа позволяет оценить сделанное, критически посмотреть на найденное решение, закрепить удачные приемы анализа условия задачи и ее решения [8].

Существуют различные способы решения задания 8 ЕГЭ по информатике. *Аналитический* способ решения предполагает знание школьниками формул комбинаторики и умение разбивать множество слов на непересекающиеся подмножества. *Программные* способы решения основаны на перечислении всех комбинаторных объектов, для чего используют вложенные циклы или функции для перечисления размещений, перестановок и сочетаний библиотеки *itertools* [5, 10].

На рассмотрении указанных способов решения можно остановиться, а можно продолжить поиск лучших решений. В некоторых задачах при использовании вложенных циклов (или функции *product*) перебираются все комбинаторные конфигурации и из них выбираются подходящие под условие. Сокращение перебора для построения только подходящих слов при решении задачи с помощью вложенных циклов, как правило, выглядит громоздко. Удобнее использовать рекурсивный подход: строить слово длины n на базе слов длины $n - 1$, в этом случае все слова длины $n - 1$ необходимо будет хранить. Избежать такого хранения позволяет использование «ленивых» вычислений с сохранением результата в глобальной переменной, однако в Python для этого можно использовать функции-генераторы.

Таким образом, **можно рассмотреть со школьниками следующие средства решения задач на перечисление слов, составленных из символов заданного алфавита, удовлетворяющих некоторым условиям:**

1) вложенные циклы;
2) функции библиотеки *itertools*;
3) рекурсивные функции;
4) рекурсивные функции-генераторы.

Рассмотрение указанных способов решения может происходить при изучении различных тем раздела «Алгоритмы и программирование» курса информатики. Возвращение к задаче о перечислении слов, составленных из символов некоторого алфавита, может осуществляться несколько раз за время изучения углубленного курса информатики в X—XI классах, что будет способствовать усилению внутрипредметных связей и, как следствие, более глубокому пониманию изучаемого материала [6].

В следующем разделе статьи мы рассмотрим особенности указанных способов решения для задачи перечисления всех размещений с повторениями заданной длины для данного алфавита.

2. Задача 1. Перечисление всех размещений с повторениями

Рассмотрим способы решения следующей **задачи:**

Миша составляет трехбуквенные слова из букв М, И, Ш, А. Каждая буква может использоваться несколько раз или не использоваться вообще. Выведите на экран все слова, которые может получить Миша.

Решение.

Способ 1. Использование вложенных циклов.

Впервые школьники могут встретиться с подобной задачей при изучении темы «Вложенные циклы». Переменная каждого из вложенных циклов отвечает за значение одного символа слова и последовательно принимает значения символов строки «МИША». На каждой итерации формируется новое слово и выводится на экран.

```
alp = "МИША"
counter = 0
for ch1 in alp:
  for ch2 in alp:
    for ch3 in alp:
      word = ch1 + ch2 + ch3
      print(word)
```

Способ 2. Использование библиотеки *itertools*.

Федеральная рабочая программа по информатике для среднего общего образования на углубленном уровне предполагает знакомство школьников с возможностями использования при разработке программ библиотек подпрограмм [11]. При решении указанной задачи можно рассмотреть функции библиотеки *itertools*. Необходимо обсудить со школьниками, что функции

ISSN 2221-1993 • **Informatics in School** • 2026 • Volume 25 No 1

product, *combination* и *permutation* возвращают объект-генератор. Объект-генератор не хранит в памяти все элементы последовательности, а строит их по одному, когда вызывается специальным образом, например, в цикле *for* [10].

```
from itertools import product
alp = "МИША"
for w in product(alp,repeat=3):
  word = "".join(w)
  print(word)
```

Способ 3. Рекурсивная генерация слов.

При изучении темы «Рекурсия» можно рассмотреть рекурсивный способ построения всех размещений с повторениями из четырех по три. Трехбуквенные слова можно получить из двухбуквенных, дописывая к каждому из них символы данного алфавита. Например, из слова «МИ» получаются слова «МИМ», «МИИ», «МИШ» и «МИА». В свою очередь, двухбуквенные слова получаются из однобуквенных, а однобуквенные — из пустой строки также дописыванием символов алфавита. Пусть функция *words*(*n*) возвращает список слов длины *n*. Слова длины *n* строятся из слов длины *n* – 1 дописыванием символов данного алфавита. Базой рекурсии является *n* = 0, в этом случае функция возвращает список, состоящий из пустой строки.

```
alp = "МИША"

def words(n):
  if n == 0:
    return [""]
  rez = []
  for word in words(n-1):
    for ch in alp:
      rez.append(word+ch)
  return rez

for word in words(3):
  print(word)
```

Способ 4. Использование рекурсивной функции-генератора.

При использовании рекурсивной функции, описанной выше, для получения всех подходящих слов длины *n* необходимо предварительно получить список слов длины *n* – 1. Заметим, что для получения слов «МИМ», «МИИ», «МИШ», «МИА» нет необходимости хранить все двухбуквенные слова — достаточно иметь слово «МИ». Избежать хранения всех двухбуквенных слов позволяет использование «ленивых» вычислений с сохранением результата в глобальной переменной, однако в Python для этого можно использовать функции-генераторы.

Для того чтобы сделать из функции генератор, вместо служебного слова *return* используют *yield*. В результате получится функция, которая возвращает итерируемый объект, т. е. при встрече слова *yield* функция не завершает свою работу, а приостанавливает, возвращая значение выражения, стоящего после слова *yield*. При следующем обращении к функции она продолжает свою работу с того места, на котором остановилась [1, 9].

Функция-генератор *words* формирует очередное подходящее слово в тот момент, когда она в очередной раз вызывается. При этом «запоминается», к какому слову *word* происходит добавление очередного символа *ch*. Таким образом, создается итерируемый объект *words*(*n*).

```
alp = "МИША"
def words(n):
  if n==0:
    yield("")
  else:
    for word in words(n-1):
      for ch in alp:
        yield(word+ch)

for word in words(3):
  print(word)
```

3. Задача 2. Перечисление размещений с повторениями, удовлетворяющих некоторым условиям

Проанализируем способы решения **задачи**, в которой множество подходящих слов не совпадает с размещениями с повторениями из символов заданного алфавита:

Миша составляет трехбуквенные слова из букв М, И, Ш, А, причем никакие две гласные или две согласные не должны стоять рядом. Каждая буква может использоваться несколько раз или не использоваться вообще. Выведите на экран все слова, которые может получить Миша.

Решение.

При реализации первого и второго способов решения можно перебирать все размещения с повторениями ($4^3 = 64$ слова) и из них выбирать слова, подходящие под условие. Очевидно, что при увеличении мощности алфавита и длины слов количество лишних комбинаций быстро растет.

Способ 1. Использование вложенных циклов.

```
alp = "МИША"
for ch1 in alp:
  for ch2 in alp:
    for ch3 in alp:
      word = ch1 + ch2 + ch3
      word1 = word.replace('И','A').
replace('М','Ш')
      if 'AA' not in word1 and 'ШШ' not in
word1:
        print(word)
```

Способ 2. Использование библиотеки *itertools*.

```
from itertools import product
alp = "МИША"
for word in (product(alp,repeat=3)):
  word = "".join(word)
  word1 = word.replace('И','A').
replace('М','Ш')
  if 'AA' not in word1 and 'ШШ' not in word1:
    print(word)
```

ISSN 2221-1993 • **Информатика в школе** • 2026 • Том 25 № 1

Способ 3. Рекурсивная генерация слов.

При реализации рекурсивного решения трехбуквенные слова также можно получить из двухбуквенных слов, дописывая к каждому из них символы, допустимые по условию задачи. Например, из слова «МИ» можно получить только два слова: «МИМ» и «МИШ». В свою очередь, двухбуквенные слова можно получить из однобуквенных. Пусть функция *words*(*n*) возвращает список слов длины *n*. Слова длины *n* получаются из слов длины *n* – 1 дописыванием *допустимых* символов данного алфавита: если последний символ текущей строки длины *n* – 1 гласный, то выбор добавляемого символа необходимо производить из согласных, и наоборот. Базой индукции удобнее взять *n* = 1, в этом случае функция возвращает список, состоящий из символов алфавита.

```
alp = "МИША"
vowel = "ИА"
consonant = "МШ"
def words(n):
  if n == 1:
    return list(alp)
  else:
    rez = []
    for word in words(n-1):
      if word[-1] in vowel:
        alph = consonant
      else:
        alph = vowel
      for ch in alph:
        rez.append(word+ch)
    return rez

for word in words(3):
  print(word)
```

Способ 4. Использование рекурсивной функции-генератора.

Такой же порядок формирования подходящих слов можно осуществить с помощью функции-генератора, при этом не сохраняя все слова длины *n* – 1 при построении слов длины *n*. При увеличении алфавита и длины генерируемых слов четвертый способ будет более эффективен как по времени, так и по используемой памяти.

```
alp = "МИША"
vowel = "ИА"
consonant = "МШ"
def words(n):
  if n == 1:
    for word in alp:
      yield word
  else:
    for word in words(n-1):
      if word[-1] in vowel:
        alph = consonant
      else:
        alph = vowel
      for ch in alph:
        yield word+ch

for word in words(3):
  print(word)
```

4. Задача 3. Поиск размещения с повторениями в упорядоченном списке по заданному номеру

Рассмотрим особенности различных способов решения задачи, в которой увеличены размер алфавита и длина генерируемой последовательности, не требующей построения всех подходящих слов:

Миша составляет 15-буквенные слова из букв С, М, Е, Ш, И, Н, К, А, причем никакие две гласные или две согласные не должны стоять рядом. Каждая буква может использоваться несколько раз или не использоваться вообще. Миша упорядочивает все полученные слова в алфавитном порядке. Вот начало списка:

1 АКАКАКАКАКАКА
2 АКАКАКАКАКАКЕ
3 АКАКАКАКАКАКИ
4 АКАКАКАКАКАМА
5 АКАКАКАКАКАМЕ
6 АКАКАКАКАКАМИ
7 АКАКАКАКАКАНА
8 АКАКАКАКАКАНЕ
9 АКАКАКАКАКАНИ
10 ...

Найдите слово под номером 45.

Решение.

Способы решения 1 и 2 будут основаны на переборе 8^{15} вариантов слов, что невозможно осуществить за разумное время.

Способ 3. Рекурсивная генерация слов.

Реализация перебора подходящих комбинаций с помощью рекурсивной функции позволяет сократить время перебора более чем в 25 000 раз ($3^7 \cdot 5^8 + 3^8 \cdot 5^7 = = 1\ 366\ 875\ 000$ вариантов слов вместо $8^{15} = 35\ 184\ 372\ 088\ 832$), однако хранение всех слов длины *n* – 1 для нахождения слов длины *n* требует большого объема памяти.

```
alp = sorted("СМЕШИНКА")
vowel = sorted("ЕИА")
consonant = sorted("СМШНК")
def words(n):
  if n == 1:
    return list(alp)
  else:
    rez = []
    for word in words(n-1):
      if word[-1] in vowel:
        alph = consonant
      else:
        alph = vowel
      for ch in alph:
        rez.append(word+ch)
    return rez

print(words(15)[44])
```

Способ 4. Использование рекурсивной функции-генератора.

Программа, приведенная выше, строит все 1 366 875 000 слов описанного в задаче списка, а по ус-

ловию необходимы только первые 45. Использование третьего способа решения без существенного его преобразования не дает возможности остановить перебор при нахождении 45-го слова списка. Использование функции-генератора позволяет это сделать естественным образом.

```
alp = sorted("СМЕШИНКА")
vowel = sorted("ЕИА")
consonant = sorted("СМШНК")
def words(n):
  if n==1:
    for ch in alp:
      yield(ch)
  else:
    for word in words(n-1):
      if word[-1] in vowel:
        alph = consonant
      else:
        alph = vowel
      for ch in alph:
        yield(word+ch)

for i, word in enumerate(words(15),1):
  if i == 45:
    print(word)
    break
```

5. Заключение

Как отмечалось в работе [3], прочтя задачу и еще не производя никаких действий, ученик должен стремиться к тому, чтобы научиться сразу видеть, что один способ непригоден для ее решения, а другой способ может быть использован. Рассмотрение описанных выше подходов позволит сформировать у школьников основу, которая необходима для осознанного выбора метода решения заданий на перечисление слов, составленных из символов заданного алфавита, удовлетворяющих некоторым условиям.

Использование рекурсивных алгоритмов и функций-генераторов даст школьникам возможность познакомиться со стандартными алгоритмами построения комбинаторных конфигураций, лучше понять работу рекурсивных функций и уже знакомых генераторов, таких как *range*, *product* и т. п., положительно скажется на создании внутрипредметных связей, что будет способствовать более глубокому усвоению материала [6].

Список источников

1. Генераторы в Python // Сайт школы 179 г. Москвы. https://ejudge.179.ru/tasks/python/2022b/pgm25__Generators.html

2. *Горбачев Н. В.* Сборник олимпиадных задач по математике. М.: МЦНМО, 2010. 559 с. EDN: QYBWMT.

3. *Готман Э. Г., Скопец З. А.* Задача одна — решения разные. М.: Просвещение, 2000. 223 с.

4. Демоверсии, спецификации, кодификаторы ЕГЭ. https://fipi.ru/ege/demoversii-specifikacii-kodifikatory#!/tab/151883967-5

5. ЕГЭ по информатике (2026) // Преподавание, наука и жизнь: сайт Константина Полякова. https://kpolyakov.spb.ru/school/ege.htm

6. *Заводчикова Н. И., Быкова И. А.* Психологические аспекты установления внутрипредметных связей между темой «Системы счисления» и разделом «Алгоритмы и программирование» // Информатика в школе. 2024. Т. 23. № 5 (184). С. 23–29. DOI: 10.32517/2221-1993-2024-23-5-23-29. EDN: WGDHWW.

7. *Окулов С. М.* Программирование в алгоритмах. М.: БИНОМ. Лаборатория знаний, 2001, 341 с. EDN: XWWXSB.

8. *Окунев А. А.* Спасибо за урок, дети! О развитии творческих способностей учащихся. М.: Просвещение, 1988. 128 с.

9. Перебор комбинаторных объектов // Сайт школы 179 г. Москвы. https://ejudge.179.ru/tasks/python/2022b/pgm26__Combinatorics.html

10. *Поляков К. Ю.* 100 баллов по информатике. Решаем задачи ЕГЭ на языке Python. М.: Лаборатория знаний, 2025. 367 с.

11. Федеральная рабочая программа среднего общего образования. Информатика (углубленный уровень) (для 10–11 классов образовательных организаций). https://edsoo.ru/wp-content/uploads/2025/07/2025_soo_frp_informatika_10_11_ugl.pdf

12. Федеральная рабочая программа среднего общего образования. Математика (углубленный уровень) (для 10–11 классов образовательных организаций). https://edsoo.ru/wp-content/uploads/2025/07/2025_soo_frp_matematika_10_11_ugl.pdf

13. *Яковлев И. В.* Комбинаторика для олимпиадников. М.: МЦНМО, 2014. 72 с.

14. Ярославские олимпиады по информатике. Сборник задач с решениями / С. Г. Волченков, П. А. Корнилов, Ю. А. Белов и др. М.: БИНОМ. Лаборатория знаний, 2010. 405 с.