DOI: 10.32517/2221-1993-2022-21-6-5-11

Л. Л. Босова
Московский педагогический государственный университет, г. Москва, Россия

Н. А. Аквилянов
Фирма «1С», г. Москва, Россия

ОБРАБОТКА СИМВОЛЬНЫХ ДАННЫХ: ОТ ПРОСТОГО К СЛОЖНОМУ

Аннотация

В статье предложены разноуровневые материалы для освоения учащимися VII—VIII классов темы «Обработка символьных данных» на языке программирования Python.

Первый уровень сложности предполагает рассмотрение минимальных теоретических сведений и базовых алгоритмов обработки символьных данных, в том числе алгоритмов редактирования текстов (замена символа, удаление и вставка символа, поиск вхождения заданного образца); материал ориентирован на учеников, осваивающих информатику на базовом уровне.

Второй (повышенный) уровень сложности предполагает не только фактическое знание изучаемого материала, включая методы обработки строк, но и способность применить его в новых ситуациях; материал может быть предложен учащимся основной школы, осваивающим информатику на углубленном уровне.

Третий (высокий) уровень сложности включает знакомство обучающихся с олимпиадными задачами; материал ориентирован на наиболее подготовленных и мотивированных школьников.

Работа со строковыми данными, реализованная на разных уровнях сложности, может стать как альтернативой, так и дополнением для задач с математическим контекстом, традиционно включаемых в содержание курса информатики основной школы.

Ключевые слова: информатика, обучение программированию, язык программирования Python, символьные данные.

1. Введение

В последние годы в условиях цифровой трансформации многих сфер нашей жизни во всем мире наблюдается большой интерес к обучению школьников программированию [3].

Отмечаются развивающие и социальные аспекты программирования [1, 2]:

- занятия программированием развивают мышление человека;
- изучение программирования способствует формированию у людей новых ценностей цифрового

Контактная информация

Босова Людмила Леонидовна, доктор пед. наук, доцент, член-корреспондент РАО, зав. кафедрой теории и методики обучения математике и информатике, Институт математики и информатики, Московский педагогический государственный университет, г. Москва, Россия; *адрес:* 119991, Россия, г. Москва, ул. Малая Пироговская, д. 1, стр. 1; *e-mail:* akulll@mail.ru

Аквилянов Никита Александрович, руководитель проекта, фирма «1С», г. Москва, Россия; *адрес:* 127473, Россия, г. Москва, ул. Селезневская, д. 34, стр. 4; *e-mail:* akvilianov@gmail.com

L. L. Bosova
Moscow Pedagogical State University, Moscow, Russia
N. A. Akvilianov
1C Company, Moscow, Russia

CHARACTER DATA PROCESSING: FROM SIMPLE TO COMPLEX

Abstract

The article proposes multi-level materials for students of grades 7–8 to master the theme "Character data processing" in the Python programming language.

The first level of complexity involves consideration of the minimum theoretical information and basic algorithms for processing character data, including algorithms for texts editing (replacing a character, deleting and inserting a character, searching for an occurrence of a given pattern); the material is aimed at students who master informatics at a basic level.

The second (advanced) level of complexity implies not only actual knowledge of the material being studied, including string processing methods, but also the ability to apply it in new situations; the material can be offered to the basic school students who master informatics at an advanced level.

The third (high) level of complexity involves the acquaintance of students with the Olympiad tasks; the material is aimed at the most prepared and motivated students.

Working with string data, implemented at different levels of complexity, can become both an alternative and a supplement for tasks with a mathematical context, traditionally included in the content of the informatics course of the basic school.

Keywords: informatics, programming training, Python programming language, character data.

общества (сообщества, совместная работа, обмен знаниями);
- благодаря навыкам программирования человек лучше понимает правила поведения в цифровой среде, чувствует себя в ней более уверенно и комфортно.

Тематический раздел «Алгоритмы и программирование» является традиционным для школьного курса информатики, традиционными являются и трудности, возникающие у школьников при его освоении. Одна из основных причин трудностей, возникающих у обучающихся при знакомстве с языком программирования, — преобладание задач с математическим контекстом (проверка делимости одного целого числа на другое; решение квадратного уравнения, имеющего вещественные корни; нахождение наибольшего общего делителя двух натуральных чисел; разложение натурального числа на простые сомножители; разбиение записи натурального числа в позиционной системе с основанием, меньшим или равным 10, на отдельные цифры и др.) [9, 10]. Устранить такого рода причину можно, если наряду с программированием алгоритмов обработки чисел *познакомить школьников с алгоритмами обработки символьных данных*. Заметим, что соответствующая тематика достаточно полно представлена в примерных рабочих программах по учебному предмету «Информатика» для основной школы базового и углубленного уровней: «Обработка символьных данных. Символьные (строковые) переменные. Посимвольная обработка строк. Подсчет частоты появления символа в строке. Встроенные функции для обработки строк» [9, 10].

Покажем, как может быть организовано освоение темы «Обработка символьных данных» на языке программирования Python учениками VII—VIII классов.

2. Краткие теоретические сведения

Приведем краткие теоретические сведения о символьных данных (данных строкового типа), с которыми необходимо познакомить обучающихся [4].

Значением строковой величины (тип *str*) в языке Python является произвольная последовательность символов, заключенная в одинарные или двойные кавычки. Символьная строка рассматривается как единый объект.

В тексте программы переменную строкового типа можно задать, заключив цепочку символов в одинарные или двойные кавычки.

Новое значение может быть записано в строку с помощью оператора ввода с клавиатуры: $s = input()$. Если значение символьной переменной считывается с клавиатуры, то его следует набирать без апострофов.

Можно проверить равенство (совпадение) строк или выяснить, какая из двух строк меньше (при этом используется поочередное сравнение кодов пар символов, образующих слова; меньшим будет то слово, у которого код символа окажется меньше).

В Python строки можно сцеплять: $a + b$ (к концу строки a прикрепляется, или «приписывается», строка b).

В результате операции $a * k$, где k — целое число, строка a повторяется k раз.

Строка состоит из последовательности символов. Узнать длину строки (количество символов в строке) можно при помощи функции *len*().

Из строки можно выделить срез — любое количество последовательно идущих символов:
- $s[i]$ — извлечение из строки одного символа, имеющего номер i; при этом считается, что нумерация начинается с 0; чтобы извлекать элементы строки справа налево, указывают отрицательные значения i, считая, что последний символ строки имеет номер –1, предпоследний — –2 и т. д.;
- $s[m:n]$ — извлечение из строки последовательности символов, начиная с символа, имеющего номер m, до символа с номером n; при этом символ с номером n в срез не входит. Если не указывать параметр m, то срез будет взят от начала строки; если не указывать параметр n, то срез будет взят до конца строки.

С помощью оператора *in* можно проверить, входит ли значение некоторой строковой переменной в ту или иную строку.

Функция *str*() переводит число или любой другой объект к строке.

Функция *int*() переводит строку в число в десятичной системе счисления. Если вызвать функцию *int*() без аргументов, она вернет 0. Чаще всего используется функция *int*() с одним аргументом — символьным представлением целого числа. В случае, когда указывается второй аргумент для функции *int*(), он обозначает систему счисления, в которой находится число, указанное в строке первого аргумента; функция *int*() возвращает его значение в десятичной системе счисления.

Другие теоретические сведения на начальном этапе работы со строковыми величинами не требуются.

3. Программирование базовых алгоритмов обработки строк

Рассмотрим программирование базовых алгоритмов обработки строк на примере последовательности задач, представленных в пособии [5].

Задание 1. Две строки.

Составьте программу, которая предлагает ввести с клавиатуры две строки a и b, а затем:

а) соединяет их в новую строку c и выводит результат;
б) определяет количество символов в новой строке c;
в) выводит на экран более длинную из строк a и b;
г) выводит на экран бóльшую из строк a и b.

Пример ввода	Пример вывода
тепло ход	а) теплоход
	б) 8
	в) тепло
	г) ход

Решение.

```
a=input()
b=input()
c=a+b
print(c)
k=len(a+b)
```

```
print(k)
print(a) if len(a)>=len(b) else print(b)
print(a) if a>=b else print(b)
```

Задание 2. Составление слов.

Составьте программу, которая из слова ИНФОРМАТИКА получает слова:

а) ФОРМА;
б) ФИРМА;
в) МАК.

Решение.

```
a="ИНФОРМАТИКА"
print(a[2:7])
print(a[2]+a[0]+a[4:7])
print(a[5:7]+a[-2])
```

Задание 3. Работа со словами строки.

Вводится строка из двух слов, разделенных пробелом. Составьте программу, которая:

а) находит позицию пробела во введенной строке и выводит номер этой позиции на экран;
б) выводит на экран первое слово введенной строки;
в) выводит на экран второе слово введенной строки;
г) меняет местами слова в исходной строке и выводит новую строку на экран.

Пример ввода	Пример вывода
обработка строк	а) 10
	б) обработка
	в) строк
	г) строк обработка

Решение.

```
a=input()
for i in range(len(a)):
    if a[i]==" ":
        n=i
print(n+1)
print(a[0:n])
print(a[n+1:len(a)])
b=a[n+1:len(a)]+" "+ a[0:n]
print(b)
```

Задание 4. Подсчет количества разных букв в строке.

Составьте программу, которая предлагает ввести строку, состоящую из строчных английских букв, и находит:

а) количество букв «s» в этой строке;
б) общее количество букв «s» и «t» в этой строке;
в) каких букв — «s» или «t» — больше в этой строке.

Пример ввода	Пример вывода
rstsstrssr	а) 5
	б) 7
	в) Больше s

```
a=input()
ks=kt=0
for i in range(len(a)):
  if a[i]=="s":
    ks=ks+1
  if a[i]=="t":
    kt=kt+1
print(ks)
print(ks+kt)
print("Больше s") if ks>kt else print("Больше t")
```

Задание 5. Подсчет слов в предложении.

Составьте программу, которая осуществляет подсчет слов в предложении, если слова в предложении разделены одним пробелом.

Решение.

```
a=input()
k=1
for i in range(len(a)):
  if a[i]==" ":
    k=k+1
print(k)
```

Задание 6. Замена букв.

Составьте программу, которая заменяет в строке из строчных английских букв все буквы «a» на букву «o».

Пример ввода	Пример вывода
abcaaf	obcoof

Решение.

```
a=input()
n=len(a)
c=""
for i in range(n):
  if a[i]=="a":
    c=c+"o"
  else:
    c=c+a[i]
print(c)
```

Задание 7. Удаление символов.

Составьте программу, которая удаляет в строке из строчных английских букв все вхождения буквы, введенной с клавиатуры.

Пример ввода	Пример вывода
rsassarssra r	sassassa

Решение.

```
a=input()
b=input()
c=""
for i in range(len(a)):
  if a[i]!=b:
    c=c+a[i]
print(c)
```

Задание 8. Вставка символов.

Составьте программу, позволяющую:

а) вставить после каждого символа исходной строки пробел и вывести полученную строку на экран;

б) удвоить каждый символ исходной строки и вывести полученную строку на экран.

Пример ввода	Пример вывода
ПРИВЕТ	а) П Р И В Е Т
	б) ППРРИИВВЕЕТТ

Решение.

```
a=input()
c=""
d=""
for i in range(len(a)):
  c=c+a[i]+" "
  d=d+2*a[i]
print(c)
print(d)
```

Обратите внимание! Программный код для части *б* рассматриваемого задания можно записать короче:

```
a=input()
for i in a:
print(2*i, end="")
```

Задание 9. Строка-палиндром.

Составьте программу, позволяющую определить, является ли введенная строка:

а) палиндромом, т. е. верно ли, что она читается одинаково слева направо и справа налево;

б) палиндромом после удаления из нее всех пробелов.

	Пример ввода	Пример вывода
а)	КАЗАК	Палиндром
	БАНАН	Нет
б)	КОТУ ТАЩАТ УТОК	Палиндром
	КОТУ ТАЩАТ УТКУ	Нет

Решение.

Программный код для части *а*:

```
a=input()
c=""
for i in range(len(a)):
  c=a[i]+c
print("Палиндром") if a==c else print ("Нет")
```

Обратите внимание! Этот программный код можно записать короче:

```
a=input()
print("Палиндром") if a==a[::-1] else
print("Нет")
```

Программный код для части *б*:

```
a=input()
d=""
c=""
for i in range(len(a)):
    if a[i]==" ":
      d=d+""
for i in range(len(d)):
    c=d[i]+d
print("Палиндром") if d==c else print ("Нет")
```

Выполнив представленную выше последовательность заданий, учащиеся:

получат представление:

- о том, как устроены базовые алгоритмы обработки символьных данных;

научатся:

- определять длину строки;
- применять операцию конкатенации к строкам;
- определять номер заданного символа в строке;
- подсчитывать количество слов в строке;
- менять местами слова в строке;
- осуществлять поиск и подсчет количества заданных элементов строки;
- осуществлять поиск и замену заданных элементов строки;
- удалять заданные элементы из строки;
- вставлять заданные элементы в строку;
- разрабатывать несложные программы обработки строк с использованием ветвлений и циклов.

4. Задачи повышенного уровня сложности

Задание 10. Строка-число.

Составьте программу, которая проверяет, может ли введенная с клавиатуры строка быть записью двоичного числа.

Решение.

```
a=input()
binary="01"
flagB=1
for i in range (len(a)):
  if a[i] not in binary:
    flagB=0
if flagB:
  print("Перед вами двоичный код")
else:
  print("Нет")
```

Задание 11. Проверка правописания.

Составьте программу, которая проверяет во введенной строке соблюдение правила правописания, выводит на экран количество допущенных ошибок и исправленную строку:

а) правописание «ча» и «ща»;

б) правописание «жи» и «ши».

	Пример ввода	Пример вывода
а)	рощя, чащя, чяща, чяша	4, роща, чаща, чаща, чаша
б)	шыло, лужы, ошыбка, дружыть	4, шило, лужи, ошибка, дружить

Решение.

```
a=input()
k=0
for i in range (len(a)-1):
  if (a[i]+a[i+1]=="чя") or (a[i]+a[i+1]=="щя"):
    a=(a[:(i+1)]+"а"+a[(i+2):])
    k+=1
  if (a[i]+a[i+1]=="жы") or (a[i]+a[i+1]=="шы"):
    a=(a[:(i+1)]+"и"+a[(i+2):])
    k+=1
print(k,", ",a,sep="")
```

5. Методы обработки строк

Во всех программах, рассмотренных выше, данные и функции использовались отдельно. Но это не единственный способ организации кода: объектно-ориентированный код строится на объединении данных и функций в одну сущность — ***объект***. Данные в таком случае называются ***атрибутами***, а функции — ***методами***. Вызов метода происходит через точку, которая идет сразу за именем переменной. В остальном методы работают как обычные функции.

После того как школьники освоят представленные выше базовые алгоритмы обработки строк (задания 1—9) и попробуют применять их для решения задач (задания 10—11), можно познакомить их с некоторыми методами обработки строк (см. табл.) [8].

Таблица

Некоторые методы обработки строк

Метод	Описание
upper()	Возвращает строку, символы которой приведены к верхнему регистру. Исходная строка не изменяется
lower()	Возвращает строку, символы которой приведены к нижнему регистру. Исходная строка не изменяется
capitalize()	Возвращает строку, в которой первая буква прописная, остальные — строчные
replace(*old*, *new*)	Возвращает строку, в которой вхождения строки *old* заменены строкой *new*
find(*str*)	Поиск подстроки в строке. Возвращает номер первого вхождения или –1.
rfind(*str*)	Поиск подстроки в строке. Возвращает номер последнего вхождения или –1
split(*символ*)	Разбиение строки по разделителю
count(*str*)	Возвращает количество непересекающихся вхождений подстроки
join(*список*) *join*(*str*)	Сборка строки из списка с разделителем. Если передать в качестве аргумента функции строку, то она будет разбита по символам с определенным разделителем

Покажем, как с использованием приведенных в таблице методов обработки строк можно оптимизировать программный код для рассмотренных выше заданий.

Программный код к рассмотренному выше ***заданию 3*** с использованием метода *find*(*str*) будет выглядеть так:

```
a=input()
print(a.find(" ")+1)
print(a[:a.find(" ")])
print(a[a.find(" ")+1:])
print(a[a.find(" ")+1:]+" "+a[:a.find(" ")])
```

Программный код к ***заданию 4*** с использованием метода *count*(*str*) будет выглядеть так:

```
a=input()
print(a.count("s"))
print(a.count("s")+a.count("t"))
print("Больше s") if (a.count("s")>a.count("t"))
  else print("Больше t")
```

Программный код к ***заданию 5*** с использованием метода *count*(*str*) будет выглядеть так:

```
print(input().count(" ")+1)
```

Программный код к ***заданию 6*** с использованием метода *replace*(*old*, *new*) будет выглядеть так:

```
a=input()
print(a.replace("a","o"))
```

Программный код к ***заданию 7*** с использованием метода *replace*(*old*, *new*) будет выглядеть так:

```
a=input()
b=input()
print(a.replace(b,""))
```

Этот код можно «свернуть» в одну строку:

```
print(input().replace(input(),""))
```

Программный код к ***заданию 8а*** с использованием метода *join*(*str*) будет выглядеть так:

```
print(" ".join(input()))
```

Программный код к ***заданию 8б*** с использованием метода *join*(*str*) и генератора списка будет выглядеть так:

```
print("".join([x*2 for x in (input())]))
```

Программный код к ***заданию 9б*** с использованием метода *replace*(*old*, *new*) будет выглядеть так:

```
a=input()
a=a.replace(" ","",a.count(" "))
print("Палиндром") if a==a[::-1] else
print("Нет")
```

Программный код к ***заданию 10*** с использованием метода *count*(*str*) будет выглядеть так:

```
x=input()
print("Перед вами двоичный код")
  if (x.count("1")+x.count("0") == len(x))
  else print("Нет")
```

Программный код к ***заданию 11*** с использованием методов *replace*(*old*, *new*) и *count*(*str*) будет выглядеть так:

```
a=input()
k=a.count("чя")+a.count("щя")+a.count("жы")+
  a.count("шы")
a=a.replace("щя","ща")
a=a.replace("чя","ча")
a=a.replace("жы","жи")
a=a.replace("шы","ши")
print(k,", ",a,sep="")
```

Мы рассмотрели лишь некоторые методы обработки строк. С другими из них мотивированные ученики могут познакомиться самостоятельно.

6. Задания высокого уровня сложности

Задание 12. Системы счисления (задание 14 демонстрационного варианта ЕГЭ-2023) [6].

Операнды арифметического выражения записаны в системе счисления с основанием 15:

$123x5_{15} + 1x233_{15}$.

В записи чисел переменной x обозначена неизвестная цифра из алфавита 15-ричной системы счисления. Определите наименьшее значение x, при котором значение данного арифметического выражения кратно 14. Для найденного значения x вычислите частное от деления значения арифметического выражения на 14 и укажите его в ответе в десятичной системе счисления. Основание системы счисления в ответе указывать не надо.

Решение.

```
a="0123456789ABCDE"
for i in a:
  x1="123"+i+"5"
  x2="1"+i+"233"
  x=int(x1,15)+int(x2,15)
  if x%14==0:
    print(x//14)
    break
```

Задание 13. Маска числа (задание 25 демонстрационного варианта ЕГЭ-2023) [6].

Назовем маской числа последовательность цифр, в которой также могут встречаться следующие символы:

- символ «?» означает ровно одну произвольную цифру;
- символ «*» означает любую последовательность цифр произвольной длины; в том числе «*» может задавать и пустую последовательность.

Например, маске 123*4?5 соответствуют числа 123405 и 12300405.

Среди натуральных чисел, не превышающих 10^{10}, найдите все числа, соответствующие маске 1?2139*4, делящиеся на 2023 без остатка.

Решение.

Сначала исследуем числа, имеющие вид: 1?21394. Другими словами, в записи чисел по заданной маске знаку «*» соответствует пустая последовательность. Вместо знака «?» может стоять любая из цифр от 0 до 9. Построим такие числа и проверим их делимость на число 2023:

```
for i in range(0,10):
  x=int("1"+str(i)+"21394")
  if x%2023==0:
    print(x)
```

Добавим к рассмотрению числа, в которых вместо знака «*» может быть любая последовательность из одной, двух или трех десятичных цифр. Такие числа можно получить, перебрав в цикле и подставив на место знака «*» все однозначные, двузначные или трехзначные числа (при попытке подставить вместо знака «*» большее количество цифр мы выйдем за границы числа 10^{10}):

```
for j in range(0,1000):
  y=x//10
  y=int(str(y)+str(j)+"4")
  if y%2023==0:
    print(y)
```

Важно не упустить из рассмотрения и такие числа, в которых вместо знака «*» стоят строки, начинающиеся с одного или двух нулей. Дополним программный код:

```
if j<100:
  y=int(str(y)+"0"+str(j)+"4")
  if y%2023==0:
    print(y)
if j<10:
  y=int(str(y)+"00"+str(j)+"4")
  if y%2023==0:
    print(y)
```

В заключение рассмотрим олимпиадную задачу, сложность решения которой связана с разработкой алгоритма, а не с его записью на языке программирования.

Задание 14. Счастливые числа (задача муниципального этапа Всероссийской олимпиады школьников по информатике, IX—XI классы, 2021 год, Московская область) [7].

Федя совсем недавно поступил в лучший вуз страны. Особенно ему стала интересна кафедра изучения счастливых чисел, т. е. тех чисел, которые состоят только из цифр 2 и 5. Научные сотрудники этой кафедры исследуют их распределение. Они поняли, что существует последовательность всех счастливых чисел в порядке возрастания (2 — первое число, 5 — второе, 22 — третье и т. д.). Они хотят найти порядковый номер счастливого числа N в данной последовательности. Федю очень заинтересовала эта задача. Он думал над ней целый день, но так ни к чему и не пришел. Можете ли вы помочь Феде и кафедре счастливых чисел найти ответ?

Решение.

Выпишем начало последовательности счастливых чисел:

2, 5, 22, 25, 52, 55, 222, 225, 252, 255, 522, 525, 552, 555, ...

Все числа записаны в алфавите из двух цифр — 2 и 5. Перейдем к более привычному двоичному алфавиту — 0 и 1. Получим:

0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, 101, 110, 111, ...

Полученные двоичные цепочки различны; цепочки одной длины расположены в алфавитном порядке.

Если бы не нули, стоящие в начале некоторых цепочек, то эти цепочки можно было бы трактовать как разные двоичные числа, а их десятичный эквивалент можно было бы связать с номером, занимаемым числом в последовательности.

Есть прием, позволяющий превратить полученные двоичные цепочки в двоичные числа: надо дописать в начале каждой цепочки единицу:

10, 11, 100, 101, 110, 111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111, ...

Мы получили двоичное представление последовательности натуральных чисел, начиная с 2:

2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, ...

Номер, занимаемый числом в последовательности, на 1 меньше значения этого числа.

Итак, чтобы выяснить номер произвольного счастливого числа, следует:

1) в записи исходного числа произвести замену цифры 2 на 0, цифры 5 — на 1;
2) в начало полученной двоичной строки приписать цифру 1;
3) считать полученную двоичную цепочку двоичным числом, перевести это двоичное число в десятичную систему счисления;
4) вычесть из полученного десятичного числа 1.

Запишем описанный выше алгоритм на языке программирования:

```
s=input()
s=s.replace("2","0").replace("5","1")
b="1"+s
d=int(b,2)
print(d-1)
```

7. Заключение

Содержание данной статьи условно можно разбить на три части.

В первой части мы рассмотрели минимальные теоретические сведения (понятие символьной величины, операции над строками, некоторые встроенные функции языка программирования для обработки строк) и базовые алгоритмы обработки символьных данных, в том числе алгоритмы редактирования текстов (замена символа, удаление и вставка символа, поиск вхождения заданного образца). К этой части относятся задания 1—9. Их выполнение по силам ученикам, осваивающим информатику на базовом уровне.

Во второй части (задания 10—11) мы рассмотрели задания повышенного уровня сложности, требующие не только фактического знания изучаемого материала, но и умения его применять в новых ситуациях. Здесь же мы рассмотрели ряд методов обработки строк, перейдя, по сути, от записи программного кода базовых алгоритмов к использованию соответствующих функций. Такой переход целесообразно делать именно после того, как обучающиеся познакомились с «внутренним» устройством функции, поняли, как она работает. Материал второй части может быть предложен учащимся основной школы, осваивающим информатику на углубленном уровне, а также мотивированным школьникам, изучающим информатику на базовом уровне.

Третья часть представлена заданиями 12—14, ориентированными на наиболее подготовленных и мотивированных учащихся, осваивающих информатику на углубленном уровне.

Таким образом, работа со строковыми данными, реализованная на разных уровнях сложности, может стать как альтернативой, так и дополнением для задач с математическим контекстом, традиционно включаемых в содержание курса информатики основной школы.

Список источников

1. *Босова Л. Л.* Как учат программированию в XXI веке: отечественный и зарубежный опыт обучения программированию в школе // Информатика в школе. 2018. № 6. С. 3—11. EDN: XZOOJV.

2. *Босова Л. Л.* Программирование как инструмент формирования вычислительного мышления обучающихся // Информатика в школе. 2020. № 10. С. 4—10. DOI 10.32517/2221-1993-2020-19-10-4-10. EDN: GURIPH.

3. *Босова Л. Л.* Школьная информатика в России и в мире // Информатизация образования и науки. 2018. № 3. С. 134—145. EDN: XSMSVV.

4. *Босова Л. Л., Аквилянов Н. А., Кочергин И. О., Штепа Ю. Л., Бурцева Т. А.* Информатика. 8—9 классы. Начала программирования на языке Python: Дополнительные главы к учебникам. М.: БИНОМ. Лаборатория знаний, 2020. 96 с. EDN: LPMXCB.

5. *Босова Л. Л., Босова А. Ю., Аквилянов Н. А.* Информатика: 7—9 классы: компьютерный практикум. М.: Просвещение, 2022. 192 с.

6. Демонстрационный вариант контрольных измерительных материалов единого государственного экзамена 2023 года по информатике. Подготовлен федеральным государственным бюджетным научным учреждением «Федеральный институт педагогических измерений». https://fipi.ru/ege/demoversii-specifikacii-kodifikatory#!/tab/151883967-5

7. Задания муниципального этапа всероссийской олимпиады школьников по информатике 2021—2022 учебный год. 9—11 классы. Московская область // Образовательный центр «Взлёт». https://olympmo.ru/news_img/training/iikt/olymp/mun_2021-2022/tasks-iikt-9-11-mun-2020-2021.pdf

8. *Кольцов Д. В.* Python: создаем программы и игры. 2-е изд. СПб.: Наука и Техника, 2019. 400 с.

9. Примерная рабочая программа основного общего образования. Информатика. Базовый уровень (для 7—9 классов образовательных организаций). Одобрена решением федерального учебно-методического объединения по общему образованию, протокол 3/21 от 27.09.2021 г. https://fgosreestr.ru/oop/237

10. Примерная рабочая программа основного общего образования. Информатика. Углубленный уровень (для 7—9 классов образовательных организаций). Одобрена решением федерального учебно-методического объединения по общему образованию, протокол 2/22 от 29.04.2022 г. https://fgosreestr.ru/oop/313