

## 4. ЛАБОРАТОРНАЯ РАБОТА 4. РАБОТА СО СТРОКАМИ

### 4.1. СПРАВОЧНЫЙ МАТЕРИАЛ

Строка представляет последовательность символов в кодировке Unicode, заключенных в кавычки. Причем для определения строк Python позволяет использовать как одинарные, так и двойные кавычки.

Для примера присвоим переменным S1, S2 и S3 текстовые значения. Если строка длинная, ее можно разбить на части и разместить их на разных строках кода. В этом случае вся строка заключается в круглые скобки, а ее отдельные части - в кавычки (S3).

```
S1 = "Унылая пора! Очей очарованье!"  
S2 = "Приятна мне твоя прощальная краса —"  
S3 = ("Люблю я пышное "  
      "природы увяданье!")
```

Если же мы хотим определить многострочный текст, то такой текст заключается в тройные двойные или одинарные кавычки:

```
S4 = """Унылая пора! очей очарованье!  
Приятна мне твоя прощальная краса --  
Люблю я пышное природы увяданье,  
В багрец и в золото одетые леса,  
В их сенях ветра шум и свежее дыханье,  
И мглой волнистою покрыты небеса,  
И редкий солнца луч, и первые морозы,  
И отдаленные седой зимы угрозы."""
```

При использовании тройных одинарных кавычек не стоит путать их с комментариями: если текст в тройных одинарных кавычках присваивается переменной, то это строка, а не комментарий. Кроме того, *строки могут заключаться в одинарные кавычки*. Обычно это используется, когда внутри строки нужно обраться к другой строке знаками кавычек.

#### Управляющие последовательности в строке

Строка может содержать ряд специальных символов - управляющих последовательностей или escape-последовательности. Некоторые из них:

- \ позволяет добавить внутрь строки слеш
- \' позволяет добавить внутрь строки одинарную кавычку
- \\" позволяет добавить внутрь строки двойную кавычку
- \\n осуществляет переход на новую строку
- \\t добавляет табуляцию (4 отступа)

#### Шаблоны в строках

В Python можно подставить в шаблон строки элементы из кортежа или словаря. Знак процента «%» между строкой и кортежем, заменяет в строке символы «%s» на элемент кортежа. Словари позволяют вставлять в строку элемент под заданным индексом. Для этого надо использовать в строке

конструкцию «%(индекс)s». В этом случае вместо «%(индекс)s» будет подставлено значение словаря под заданным индексом. Например,

```
name="проф.Иванов"
mark=4
print ('Имя: %s\nОценка: %s\nПримечание: %s' % (name, mark, 5 * "!"))

Имя: проф.Иванов
Оценка: 4
Примечание: !!!!!

print ('This %(verb)s a %(noun)s.' % {'noun': 'test', 'verb': 'is'})
#This is a test.
```

Таким образом, можно вставлять в строку значения через ключ словаря:

```
print('%(1)s, %(2).3f' % {'1': 'AB', '2': 2.33333})
#AB, 2.333
```

### Форматирование строк с использованием функции format()

Метод **format** возвращает отформатированную версию строки, заменяя идентификаторы в фигурных скобках. Идентификаторы могут быть позиционными, числовыми индексами, ключами словарей, именами переменных.

Аргументов в format() может быть больше, чем идентификаторов в строке. В таком случае оставшиеся игнорируются.

Идентификаторы могут быть либо индексами аргументов, либо ключами:

```
print ("{}, {} and {}".format('one', 1, 'I'))
#one, 1 and I
print (" {1}, {2} and {0}".format('one', 1, 'I'))
#1, I and one
nums=[3, 4, 5, 6, 2, 0]
print (nums)
print ("{}{}{}".format(*nums))
#345
print ("{0}{2}{4}".format(*nums))
#352
u = {'name': 'bob', 'age': 35}
print (u)
print ('{name}-{age}'.format(**u))
#bob-35
print ('{name}'.format(**u))
#bob
print ('{name}-{age}'.format(name="pi", age=3.14))
#pi-3.14
```

```

print ('{0}-{age}'.format("sin", **u))
#sin-35
#Вывод атрибутов объекта:
class house:
    size = "big"
    street = "main"

h = house()
print (h)
#<__main__.house object at 0x7fde00dcfb90>
print ('{0.size}, {0.street}'.format(h))
#big, main'

```

Можно задавать ширину поля и выравнивание:

```

print ('{name:10}-{age:3}'.format(**u))
#bob          - 35'
print ('{name:>10}-{age:>3}'.format(**u))
#'          bob- 35'
print ('{name:^10}-{age:^3}'.format(**u))
#'    bob    -35 '

```

Вывод вещественных чисел:

```

print ('{0}'.format(4/3))
#1.3333333333333333'
print ('{0:f}'.format(4/3))
#1.333333'
print ('{0:.2f}'.format(4/3))
#1.33'
print ('{0:10.2f}'.format(4/3))
#'      1.33'
print ('{0:10e}'.format(4/3))
#1.333333e+00'

```

### **f-строки**

В Python версии 3.6 и выше появился еще один метод форматирования строк — "f-строки", с его помощью можно использовать встроенные выражения внутри строк:

```

name = 'Alex'
print(f'Hello, {name}!')
#Hello, Alex!

```

Такой способ форматирования очень мощный, так как дает возможность встраивать выражения:

```

a = 7
b = 8
print(f'Семь плюс восемь будет {a + b}, а не {2 * (a + b)}.')
#Семь плюс восемь будет 15, а не 30.'

```

Приведем еще пример использования **f-строк**:

```
login = "Николай"
password = "123456789"
user = f"Текущий пользователь {login}, его пароль: {password}"
print(user)
```

### Обращение к символам строки

И мы можем обратиться к отдельным символам строки по индексу в квадратных скобках:

```
c = login[2]
print(c)
```

Результатом выполнения этой программы будет буква «к», потому что индексация элементов строки начинается с 0, а индекс **len(str)-1** – это последний символ строки.

Чтобы получить доступ к символам, начиная с конца строки, можно использовать отрицательные индексы. Так, индекс -1 будет представлять последний символ, а -2 - предпоследний символ и так далее:

```
cl = login[-3]
print(cl)
```

Результатом выполнения этой программы будет буква «л».

### Срез строки

Если в индексе указано через двоеточие два значения a и b, то это подстрока с индексами от a до b (не включая его). Иногда называют **срез строки**, например,

```
S='Hello'
```

```
S[1:4] # это строка ell
```

```
S[-4:-1] # это строка ell
```

**S[1:-1]** # это тоже строка ell — строка без первого и последнего символа (срез начинается с символа с индексом 1 и заканчивается индексом -1, не включая его)

```
S[1:] # это строка ello – удаляет первый символ в строке
```

```
S[:-1] # это строка Hell – удаляет последний символ в строке
```

Если задать срез с тремя параметрами **S[a:b:d]**, то третий параметр задает шаг, то есть будут взяты символы с индексами a, a + d, a + 2 \* d и т. д. При задании значения третьего параметра, равному 2, в срез попадет каждый второй символ, а если взять значение среза, равное -1, то символы будут идти в обратном порядке. Например, можно перевернуть строку срезом **S[::-1]**.

### Функции и методы работы со строками

Язык Python имеет очень широкий набор функций по обработке строк. Ниже приводится перечень основных функций и методов работы со строками.

Функция или метод	Назначение	Пример
<code>S1+S2</code>	Конкатенация (сложение строк)	<code>print(S1+S2)</code> <sup>1</sup>
<code>S1*3</code>	Повторение строки	<code>print(S1*3)</code> <sup>2</sup>
<code>S1[i]</code>	Обращение по индексу	<code>print(S1[0])</code> – «У»
<code>S1[i:j:step]</code>	Извлечение среза - извлекается последовательность символов начиная с индекса i по индекс j (не включая) через шаг step.	<code>print(S1[1:10:2])</code> Результат выполнения: нляпр
<code>len(S1)</code>	Длина строки	<code>print(len(S1))</code> Результат: 29
<code>S.join(список)</code>	Соединение строк из последовательности через разделитель, заданный строкой. Например, <code>S1.join("123")</code>	Повторит строку S1, 3 раза, разбивая символами 1, 2 и 3.
<code>S1.count(S[,i,j])</code>	Количество вхождений подстроки S в строку S1. Результатом является число. Можно указать позицию начала поиска i и окончания поиска j.	
<code>S.find(str, [start], [end])</code>	Поиск подстроки в строке. Возвращает номер первого вхождения или вызывает ValueError.	
<code>S.rindex(str, [start], [end])</code>	Поиск подстроки в строке. Возвращает номер последнего вхождения или вызывает ValueError.	
<code>S.replace(шаблон, замена)</code>	Замена шаблона.	
<code>S.split(символ)</code>	Разбиение строки по разделителю.	
<code>S.upper()</code>	Преобразование строки к верхнему регистру.	
<code>S.lower()</code>	Преобразование строки к нижнему регистру.	

Приведем еще некоторые функции и методы работы со строками

<sup>1</sup> | Унылая пора! Очей очарованье! Приятна мне твоя прощальная краса –

<sup>2</sup>

| Унылая пора! Очей очарованье! Унылая пора! Очей очарованье! Унылая пора! Очей очарованье!

Функция или метод	Описание
<code>S = 'str'; S = "str"; S = """str"""; S = """"str"""""</code>	Литералы строк
<code>S = "s\np\ta\nbbb"</code>	Экранированные последовательности
<code>S = r"C:\temp\new"</code>	Неформатированные строки (подавляют экранирование)
<code>S = b"byte"</code>	Строка байтов
<code>S.rfind(str, [start],[end])</code>	Поиск подстроки в строке. Возвращает номер последнего вхождения или -1
<code>S.index(str, [start],[end])</code>	Поиск подстроки в строке. Возвращает номер первого вхождения или вызывает ValueError
<code>S.isdigit()</code>	Состоит ли строка из цифр (функции с приставкой is возвращают True – если условие выполняется, иначе - False)
<code>S.isalpha()</code>	Состоит ли строка из букв
<code>S.isalnum()</code>	Состоит ли строка из цифр или букв
<code>S.islower()</code>	Состоит ли строка из символов в нижнем регистре
<code>S.isupper()</code>	Состоит ли строка из символов в верхнем регистре
<code>S.isspace()</code>	Состоит ли строка из неотображаемых символов (пробел, символ перевода страницы ("f"), "новая строка" ("n"), "перевод каретки" ("r"), "горизонтальная табуляция" ("t") и "вертикальная табуляция" ("v"))
<code>S.istitle()</code>	Начинаются ли слова в строке с заглавной буквы
<code>S.startswith(str)</code>	Начинается ли строка S с шаблона str
<code>S.endswith(str)</code>	Заканчивается ли строка S шаблоном str
<code>ord(символ)</code>	Выдает код символа
<code>chr(число)</code>	Выдает символ по его коду ASCII
<code>S.capitalize()</code>	Переводит первый символ строки в верхний регистр, а все остальные в нижний
<code>S.center(width, [fill])</code>	Возвращает отцентрированную строку, по краям которой стоит символ fill (пробел по умолчанию)
<code>S.count(str, [start],[end])</code>	Возвращает количество непересекающихся вхождений подстроки в диапазоне [начало, конец] (0 и длина строки по умолчанию)

Функция или метод	Описание
<b>S.expandtabs</b> ([tabsize])	Возвращает копию строки, в которой все символы табуляции заменяются одним или несколькими пробелами, в зависимости от текущего столбца. Если TabSize не указан, размер табуляции полагается равным 8 пробелам
<b>S.lstrip</b> ([chars])	Удаление пробельных символов в начале строки
<b>S.rstrip</b> ([chars])	Удаление пробельных символов в конце строки
<b>S.strip</b> ([chars])	Удаление пробельных символов в начале и в конце строки
<b>S.partition</b> (шаблон)	Возвращает кортеж, содержащий часть перед первым шаблоном, сам шаблон, и часть после шаблона. Если шаблон не найден, возвращается кортеж, содержащий саму строку, а затем две пустых строки
<b>S.rpartition</b> (sep)	Возвращает кортеж, содержащий часть перед последним шаблоном, сам шаблон, и часть после шаблона. Если шаблон не найден, возвращается кортеж, содержащий две пустых строки, а затем саму строку
<b>S.swapcase</b> ()	Переводит символы нижнего регистра в верхний, а верхнего – в нижний
<b>S.title</b> ()	Первую букву каждого слова переводит в верхний регистр, а все остальные в нижний
<b>S.zfill</b> (width)	Делает длину строки не меньшей width, по необходимости заполняя первые символы нулями
<b>S.ljust</b> (width, fillchar=" ")	Делает длину строки не меньшей width, по необходимости заполняя последние символы символом fillchar
<b>S.rjust</b> (width, fillchar=" ")	Делает длину строки не меньшей width, по необходимости заполняя первые символы символом fillchar

При работе с символами следует учитывать, что строка - это неизменяемый (immutable) тип, поэтому если мы попробуем изменить какой-то отдельный символ строки, то мы получим ошибку.

## 4.2. ПРИМЕРЫ РЕШЕНИЯ ЗАДАЧ

**Пример 1.** Ввести строку и любой символ. Подсчитать сколько раз встретился в этой строке введенный символ.

**Код решения:**

```
sum = 0
s1=input("Введите строку: ")
simv=input("Введите символ: ")
for i in range(len(s1)):
    if (s1[i] == simv):
        sum = sum + 1
print("Количество символов '"+simv+"' в строке '"+s1+"' равно ",sum)
```

**Скриншот выполнения:**

```
Введите строку: Унылая пора! Очей очарованье !
Введите символ: !
Количество символов '!' в строке 'Унылая пора! Очей очарованье !' равно 2
```

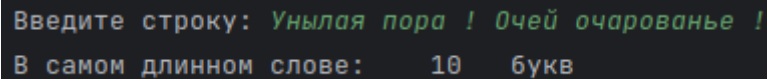
**Пример 2.** Ввести строку, содержащую несколько слов, между которыми один или несколько пробелов. Подсчитать: сколько букв в самом длинном слове.

**Код решения:**

```
row=input("Введите строку: ")
k = len(row) #длина введенной строки
maxletters = 1 #максимальное количество букв в самом длинном слове
flag = 0 #флаг начала нового слова
kol = 0 #количество букв в текущем слове
for i in range(k): #цикл по всем буквам в строке
    if (flag == 0): #новое слово?
        if (row[i] != ' '):#текущий символ не пробел?
            flag = 1 #включаем флаг для подсчета букв в новом слове
            kol = 0
    else: #продолжаем считать буквы в новом слове
        kol = kol + 1 #увеличиваем количество букв в слове
        if (row[i] == ' '): #если встречен пробел - завершаем и...
            flag = 0 #устанавливаем флаг нового слова
        if (kol > maxletters): #если количество букв в текущем слове больше max
            maxletters = kol #запоминаем новое максимальное количество букв в слове
print("В самом длинном слове: ", maxletters, " букв ")
```



### Скриншот выполнения:



```
Введите строку: Унылая пора ! Очей очарованье !
В самом длинном слове: 10 букв
```

**Пример 3.** Проверить, является ли слово или строка палиндромом (то есть одинаково читается справа налево и слева направо). Примеры палиндромов: «шалаш», «А роза упала на лапу Азора».

### Решение:

1. Пользователь вводит строку или слово: `S=input("Введите слово или строку")`.
2. Для введенной строки необходимо удалить пробелы. Используем конструкцию `FOR`, которая выполнится столько раз, какую длину имеет строка. Затем определим логическую переменную `flag` и присвоим ей значение 1: `flag=1`.
3. В теле цикла будем проверять следующее условие: `s[i]!=' '`. Данное логическое выражение будет истинно в том случае, если *i*-ый элемент строки не будет равен пробелу, тогда выполнится команда следующая после двоеточия: `string+=s[i]`.
4. К строке `string`, которая была объявлена в начале программы, будет добавляться посимвольно строка `s`, но уже без пробелов. Затем определим логическую переменную `flag` и присвоим ей значение 1: `flag=1`.
5. Для проверки строки на "палиндром" воспользуемся циклической конструкцией `for`. Длина половины строки находится делением нацело на 2. Если количество символов нечетно, то стоящий в середине не учитывается, т.к. его сравниваемая пара - он сам. Количество повторов цикла равно длине половины строки. Длину строки определим функцией `len(s)`, где аргумент введенная нами строка `s`. Зная длину строки, можно вычислить количество повторов цикла. Для этого целочисленно разделим длину строки на 2: `len(s)//2`.
6. Для задания диапазона для цикла используем функцию `range()`, в которой аргументом будет являться половина длины строки: `range(len(s)//2 )`. `for i in range(len(s)//2 )`.
7. Если символ с индексом *i* не равен "симметричному" символу с конца строки (который находится путем индексации с конца) `if s[i] != s[-1-i]`, то переменной `flag` присваивается значение 0 и происходит выход из цикла командой `break`.
8. Далее, при помощи условной конструкции `if-else` в зависимости от значения `flag` либо - 0, либо -1 выводится сообщение, что строка палиндром, либо нет.

### Код программы:

```
S=input("Введите слово или строку: \n")
flag = 1
string = " "
S=S.lower()
for i in range(len(S)):
    if S[i]!=" ":
        string = string + S[i]
print(string)
for i in range(len(S)//2):
    if string[i]!=string[-i]:
        flag = 0
        break
if flag: print("Палиндром")
else: print("Не палиндром")
```

### Результат:

```
-----
Введите слово или строку:
А роза упала на лапу Азора
  арозаупаланалапуазора
Палиндром
```

## 4.3. ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

### Общие задания для всех вариантов

1. Нарисовать блок-схему алгоритма, написать и отладить программу для примера 1.
2. Нарисовать блок-схему алгоритма, написать и отладить программу для примера 2.
3. Нарисовать блок-схему алгоритма, написать и отладить программу для примера 3.

### Задачи для различных вариантов

При выполнении заданий на обработку русских букв можно считать, что буква «ё» в исходных строковых данных отсутствует (или приравнивается в букве е).

### Символы и их коды. Формирование строк

1. Дан символ С. Вывести его код (то есть номер в кодовой таблице).
2. Дано целое число N ( $32 \leq N \leq 126$ ). Вывести символ с кодом, равным N.
3. Дан символ С. Вывести два символа, первый из которых предшествует символу С в кодовой таблице, а второй следует за символом С.
4. Дано целое число N ( $1 \leq N \leq 26$ ). Вывести N первых прописных (то есть заглавных) букв латинского алфавита.

5. Дано целое число  $N$  ( $1 \leq N \leq 26$ ). Вывести  $N$  последних строчных (то есть маленьких) букв латинского алфавита в обратном порядке (начиная с буквы «z»).
6. Дан символ  $C$ , изображающий цифру или букву (латинскую или русскую). Если  $C$  изображает цифру, то вывести строку «digit», если латинскую букву — вывести строку «lat», если русскую — вывести строку «rus».
7. Дана непустая строка. Вывести коды ее первого и последнего символа.
8. Дано целое число  $N$  ( $> 0$ ) и символ  $C$ . Вывести строку длины  $N$ , которая состоит из символов  $C$ .
9. Дано четное число  $N$  ( $> 0$ ) и символы  $C1$  и  $C2$ . Вывести строку длины  $N$ , которая состоит из чередующихся символов  $C1$  и  $C2$ , начиная с  $C1$ .
10. Дана строка. Вывести строку, содержащую те же символы, но расположенные в обратном порядке.
11. Дана непустая строка  $S$ . Вывести строку, содержащую символы строки  $S$ , между которыми вставлено по одному пробелу.
12. Дана непустая строка  $S$  и целое число  $N$  ( $> 0$ ). Вывести строку, содержащую символы строки  $S$ , между которыми вставлено по  $N$  символов «\*» (звездочка).

### **Посимвольный анализ и преобразование строк. Строки и числа**

13. Дана строка. Подсчитать количество содержащихся в ней цифр.
14. Дана строка. Подсчитать количество содержащихся в ней прописных латинских букв.
15. Дана строка. Подсчитать общее количество содержащихся в ней строчных латинских и русских букв.
16. Дана строка. Преобразовать в ней все прописные латинские буквы в строчные.
17. Дана строка. Преобразовать в ней все строчные буквы (как латинские, так и русские) в прописные.
18. Дана строка. Преобразовать в ней все строчные буквы (как латинские, так и русские) в прописные, а прописные — в строчные.
19. Дана строка. Если она представляет собой запись целого числа, то вывести 1, если вещественного (с дробной частью) — вывести 2; если строку нельзя преобразовать в число, то вывести 0. Считать, что дробная часть вещественного числа отделяется от его целой части десятичной точкой «.».
20. Дано целое положительное число. Вывести символы, изображающие цифры этого числа (в порядке слева направо).
21. Дано целое положительное число. Вывести символы, изображающие цифры этого числа (в порядке справа налево).
22. Дана строка, изображающая целое положительное число. Вывести сумму цифр этого числа.
23. Дана строка, изображающая арифметическое выражение вида: «<цифра>±<цифра>±. . . ±<цифра>», где на месте знака операции «±» находится символ «+» или «-» (например, «4+7-2-8»). Вывести значение данного выражения (целое число).
24. Дана строка, изображающая двоичную запись целого положительного числа. Вывести строку, изображающую десятичную запись этого же числа.
25. Дана строка, изображающая десятичную запись целого положительного числа. Вывести строку, изображающую двоичную запись этого же числа.

### **Обработка строк с помощью стандартных функций. Поиск и замена**

В заданиях, связанных с поиском и заменой подстрок, можно считать, что исходная строка не содержит перекрывающихся вхождений требуемых подстрок. В заданиях 32, 35 и 38, кроме этого, можно также считать, что удаление (в 32 и 35) или замена (в 38) любого вхождения подстроки не приведет к появлению в строке новых вхождений данной подстроки.

26. Дано целое число  $N (> 0)$  и строка  $S$ . Преобразовать строку  $S$  в строку длины  $N$  следующим образом: если длина строки  $S$  больше  $N$ , то отбросить первые символы, если длина строки  $S$  меньше  $N$ , то в ее начало добавить символы «.» (точка).
27. Даны целые положительные числа  $N1$  и  $N2$  и строки  $S1$  и  $S2$ . Получить из этих строк новую строку, содержащую первые  $N1$  символов строки  $S1$  и последние  $N2$  символов строки  $S2$  (в указанном порядке).
28. Дан символ  $C$  и строка  $S$ . Удвоить каждое вхождение символа  $C$  в строку  $S$ .
29. Дан символ  $C$  и строки  $S$ ,  $S0$ . Перед каждым вхождением символа  $C$  в строку  $S$  вставить строку  $S0$ .
30. Дан символ  $C$  и строки  $S$ ,  $S0$ . После каждого вхождения символа  $C$  в строку  $S$  вставить строку  $S0$ .
31. Даны строки  $S$  и  $S0$ . Проверить, содержится ли строка  $S0$  в строке  $S$ . Если содержится, то вывести TRUE, если не содержится, то вывести FALSE.
32. Даны строки  $S$  и  $S0$ . Найти количество вхождений строки  $S0$  в строку  $S$ .
33. Даны строки  $S$  и  $S0$ . Удалить из строки  $S$  первую подстроку, совпадающую с  $S0$ . Если совпадающих подстрок нет, то вывести строку  $S$  без изменений.
34. Даны строки  $S$  и  $S0$ . Удалить из строки  $S$  последнюю подстроку, совпадающую с  $S0$ . Если совпадающих подстрок нет, то вывести строку  $S$  без изменений.
35. Даны строки  $S$  и  $S0$ . Удалить из строки  $S$  все подстроки, совпадающие с  $S0$ . Если совпадающих подстрок нет, то вывести строку  $S$  без изменений.
36. Даны строки  $S$ ,  $S1$  и  $S2$ . Заменить в строке  $S$  первое вхождение строки  $S1$  на строку  $S2$ .
37. Даны строки  $S$ ,  $S1$  и  $S2$ . Заменить в строке  $S$  последнее вхождение строки  $S1$  на строку  $S2$ .
38. Даны строки  $S$ ,  $S1$  и  $S2$ . Заменить в строке  $S$  все вхождения строки  $S1$  на строку  $S2$ .
39. Дана строка, содержащая по крайней мере один символ пробела. Вывести подстроку, расположенную между первым и вторым пробелом исходной строки. Если строка содержит только один пробел, то вывести пустую строку.
40. Дана строка, содержащая по крайней мере один символ пробела. Вывести подстроку, расположенную между первым и последним пробелом исходной строки. Если строка содержит только один пробел, то вывести пустую строку.

### **Анализ и преобразование слов в строке**

Во всех заданиях данного пункта предполагается, что исходные строки являются непустыми и не содержат начальных и конечных пробелов.

41. Дана строка, состоящая из русских слов, разделенных пробелами (одним или несколькими). Найти количество слов в строке.
42. Дана строка, состоящая из русских слов, набранных заглавными буквами и разделенных пробелами (одним или несколькими). Найти количество слов, которые начинаются и заканчиваются одной и той же буквой.
43. Дана строка, состоящая из русских слов, набранных заглавными буквами и разделенных пробелами (одним или несколькими). Найти количество слов, которые содержат хотя бы одну букву «А».
44. Дана строка, состоящая из русских слов, набранных заглавными буквами и разделенных пробелами (одним или несколькими). Найти количество слов, которые содержат ровно три буквы «А».
45. Дана строка, состоящая из русских слов, разделенных пробелами (одним или несколькими). Найти длину самого короткого слова.
46. Дана строка, состоящая из русских слов, разделенных пробелами (одним или несколькими). Найти длину самого длинного слова.
47. Дана строка, состоящая из русских слов, разделенных пробелами (одним или несколькими). Вывести строку, содержащую эти же слова, разделенные одним символом «.» (точка). В конце строки точку не ставить.
48. Дана строка, состоящая из русских слов, набранных заглавными буквами и разделенных пробелами (одним или несколькими). Преобразовать каждое слово в строке, заменив в нем все

последующие вхождения его первой буквы на символ «.» (точка). Например, слово «МИНИМУМ» надо преобразовать в «МИНИ.У.». Количество пробелов между словами не изменять.

49. Дана строка, состоящая из русских слов, набранных заглавными буквами и разделенных пробелами (одним или несколькими). Преобразовать каждое слово в строке, заменив в нем все предыдущие вхождения его последней буквы на символ «.» (точка). Например, слово «МИНИМУМ» надо преобразовать в «.ИНИ.УМ». Количество пробелов между словами не изменять.

50. Дана строка, состоящая из русских слов, разделенных пробелами (одним или несколькими). Вывести строку, содержащую эти же слова, разделенные одним пробелом и расположенные в обратном порядке.

51. Дана строка, состоящая из русских слов, набранных заглавными буквами и разделенных пробелами (одним или несколькими). Вывести строку, содержащую эти же слова, разделенные одним пробелом и расположенные в алфавитном порядке.

52. Дана строка-предложение на русском языке. Преобразовать строку так, чтобы каждое слово начиналось с заглавной буквы. Словом считать набор символов, не содержащий пробелов и ограниченный пробелами или началом/концом строки. Слова, не начинающиеся с буквы, не изменять.

53. Дана строка-предложение на русском языке. Подсчитать количество содержащихся в строке знаков препинания.

54. Дана строка-предложение на русском языке. Подсчитать количество содержащихся в строке гласных букв.

55. Дана строка-предложение на русском языке. Вывести самое длинное слово в предложении. Если таких слов несколько, то вывести первое из них. Словом считать набор символов, не содержащий пробелов, знаков препинания и ограниченный пробелами, знаками препинания или началом/концом строки.

56. Дана строка-предложение на русском языке. Вывести самое короткое слово в предложении. Если таких слов несколько, то вывести последнее из них. Словом считать набор символов, не содержащий пробелов, знаков препинания и ограниченный пробелами, знаками препинания или началом/концом строки.

57. Дана строка-предложение с избыточными пробелами между словами. Преобразовать ее так, чтобы между словами был ровно один пробел.

### **Дополнительные задания на обработку строк**

58. Дана строка, содержащая полное имя файла, то есть имя диска, список каталогов (путь), собственно имя и расширение. Выделить из этой строки имя файла (без расширения).

59. Дана строка, содержащая полное имя файла, то есть имя диска, список каталогов (путь), собственно имя и расширение. Выделить из этой строки расширение файла (без предшествующей точки).

60. Дана строка, содержащая полное имя файла. Выделить из этой строки название первого каталога (без символов «\»). Если файл содержится в корневом каталоге, то вывести символ «\».

61. Дана строка, содержащая полное имя файла. Выделить из этой строки название последнего каталога (без символов «\»). Если файл содержится в корневом каталоге, то вывести символ «\».

62. Дана строка-предложение на русском языке. Зашифровать ее, выполнив циклическую замену каждой буквы на следующую за ней в алфавите и сохранив при этом регистр букв («А» перейдет в «Б», «а» — в «б», «Б» — в «В», «я» — в «а» и т. д.). Букву «ё» в алфавите не учитывать («е» должна переходить в «ж»). Знаки препинания и пробелы не изменять.

63. Дана строка-предложение на русском языке и число  $K$  ( $0 < K < 10$ ). Зашифровать строку, выполнив циклическую замену каждой буквы на букву того же регистра, расположенную в алфавите на  $K$ -й позиции после шифруемой буквы (например, для  $K = 2$  «А» перейдет в «В», «а» — в «в», «Б» — в «Г», «я» — в «б» и т. д.). Букву «ё» в алфавите не учитывать, знаки препинания и пробелы не изменять.

64. Дано зашифрованное предложение на русском языке (способ шифрования описан в задании 63) и кодовое смещение  $K$  ( $0 < K < 10$ ). Расшифровать предложение.

65. Дано зашифрованное предложение на русском языке (способ шифрования описан в задании 63) и его расшифрованный первый символ С. Найти кодовое смещение К и расшифровать предложение.

66. Дана строка-предложение. Зашифровать ее, поместив вначале все символы, расположенные на четных позициях строки, а затем, в обратном порядке, все символы, расположенные на нечетных позициях (например, строка «Программа» превратится в «ргамамроП»).

67. Дано предложение, зашифрованное по правилу, описанному в задании 66. Расшифровать это предложение.

68. Дана строка, содержащая цифры и строчные латинские буквы. Если буквы в строке упорядочены по алфавиту, то вывести 0; в противном случае вывести номер первого символа строки, нарушающего алфавитный порядок.

69. Дана строка, содержащая латинские буквы и круглые скобки. Если скобки расставлены правильно (то есть каждой открывающей соответствует одна закрывающая), то вывести число 0. В противном случае вывести или номер позиции, в которой расположена первая ошибочная закрывающая скобка, или, если закрывающих скобок не хватает, число -1.

70. Дана строка символов, среди которых есть одна открывающаяся и одна закрывающаяся скобки. Вывести на экран все символы, расположенные внутри этих скобок.

**Бонусное задание.** Строка содержит произвольный русский текст. Проверить: каких букв в нем больше: гласных или согласных и напечатать их количество.

#### 4.4. ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ ДОЛЖЕН ВКЛЮЧАТЬ

1. Титульный лист по форме с номером варианта.
2. Для каждой задачи из общего списка задач (3 общих задач) и списка задач по вариантам (14 задач по вариантам), всего 17 задач:
  - условие задачи;
  - блок-схема алгоритма;
  - программный код решения этой задачи (листинг);
  - скриншоты выполнения программы.

---

**Внимание!** Отчет должен быть набран шрифтом **Times New Roman** и отформатирован: поля: левое – 3,5; правое – 1,5; нижнее и верхнее – 2 см; красная строка (отступ) - 1 см; межстрочный интервал – одинарный; правый край выровнен по ширине; рисунки сопровождаются надписями.

## Варианты заданий

[illegible]