

# **ЗАДАНИЕ №24**

20 минут

# **ТИП 1**

**ПОИСК ИДУЩИХ ПОДРЯД ОДИНАКОВЫХ / РАЗЛИЧНЫХ  
СИМВОЛОВ**

# ЗАДАНИЕ 1

В текстовом файле **1.txt** находится цепочка из символов, в которую могут входить заглавные буквы латинского алфавита A...Z и десятичные цифры. Найдите длину самой длинной подцепочки, состоящей из одинаковых символов. Если в файле несколько подходящих цепочек одинаковой длины, нужно взять **первую** из них. Выведите сначала символ, из которого строится эта подцепочка, а затем через пробел – длину этой подцепочки.

# ЗАДАНИЕ 1

План решения задачи:

- 1) открыть файл, сохранить данные из файла в строковую переменную
- 2) идти по символам строковой переменной, сравнивая текущий символ с предыдущим и увеличивая переменную, отвечающую за длину цепочки
- 3) если текущий символ  $\neq$  предыдущему – конец очередной цепочки  $\Rightarrow$  нужно посмотреть, не максимальная ли длина у нашей цепочки
- 4) вывод максимальной длины цепочки

# ЗАДАНИЕ 1

Открытие файла, сохранение данных из файла в строковую переменную

```
with open( "1.txt", "r" ) as F:
```

```
    s = F.readline()
```

Открыли файл 1.txt на чтение (флажок r – reading), считали **весь** файл в строчку s

Идём по символам строки s:

```
for c in s:    #перебор всех символов
```

```
    ....
```

# ЗАДАНИЕ 1

Дальше нужно сравнивать текущий символ (с) с предыдущим (р) и если что – увеличивать длину цепочки, поэтому:

```
p = "" #предыдущий символ
```

```
L = 0 #длина текущей цепочки
```

```
for c in s: #перебор всех символов
```

```
    if c == p:
```

```
        L += 1
```

```
    else:
```

```
        .....
```

```
    p = c #на следующем шаге предыдущий символ – это с
```

# ЗАДАНИЕ 1

Если цепочка закончилась ( $c \neq p$ ), длина текущей цепочки становится равна 1:

$p = ''$  #предыдущий символ

$L = 0$  #длина текущей цепочки

$\max L = 0$  #максимальная длина цепочки

$\max C = ''$  #символ, из которого состоит первая цепочка максимальной длины

for  $c$  in  $s$ : #перебор всех символов

    if  $c == p$ :

$L += 1$

    else:

$L = 1$  #теперь цепочка состоит всего **из одного** символа!

$p = c$

# ЗАДАНИЕ 1

На **каждом** шаге проверяем, не нашли ли мы цепочку максимальной длины:

`p = ""` #предыдущий символ

`L = 0` #длина текущей цепочки

`maxL = 0` #максимальная длина цепочки

`maxC = ""` #символ, из которого состоит первая цепочка максимальной длины

`for c in s:` #перебор всех символов

`if c == p:`

`L += 1`

`else:`

`L = 1`

**`if L > maxL:`**

**`maxL = L`**

**`maxC = p`**

`p = c`



```
with open( "1.txt", "r" ) as F:
    s = F.readline()
p = ""
L = 0
maxL = 0
maxC = ""
for c in s:
    if c == p:
        L += 1
    else:
        L = 1
    if L > maxL:
        maxL = L
        maxC = p
    p = c
print(maxC, maxL)
```

**Ответ: S  
153**

# САМОСТОЯТЕЛЬНО

В текстовом файле `1.txt` находится цепочка из символов, в которую могут входить заглавные буквы латинского алфавита A...Z и десятичные цифры. Найдите длину самой длинной подцепочки, состоящей из одинаковых символов. Если в файле несколько подходящих цепочек одинаковой длины, нужно взять **последнюю** из них. Выведите сначала символ, из которого строится эта подцепочка, а затем через пробел – длину этой подцепочки.

# САМОСТОЯТЕЛЬНО

4 153

Т.к. ищется последняя подходящая цепочка, нужно обновлять максимальное значение  $\text{maxL}$ , если мы встретили новое  $L$ , равное  $\text{maxL}$ :

if  $L \geq \text{maxL}$ :

# ЗАДАНИЕ 2

В текстовом файле **2.txt** находится цепочка из символов, в которую могут входить заглавные буквы латинского алфавита A...Z и десятичные цифры. Найдите длину самой длинной подцепочки, состоящей из одинаковых символов. Если в файле несколько подходящих цепочек одинаковой длины, нужно взять первую из них. **Для каждой цепочки максимальной длины** выведите в отдельной строке сначала символ, из которого строится эта цепочка, а затем через пробел – длину этой цепочки.

# ЗАДАНИЕ 2

Решение практически не отличается от предыдущего, но если длины цепочек равны, нужно сохранять не один символ, а множество символов. Удобнее всего сохранять их в строчку.

Задание 1:

```
if L > maxL:  
    maxL = L  
    maxC = p
```

Задание 2:

```
if L > maxL:  
    maxL = L  
    maxC = p  
elif L == maxL:  
    maxC = maxC + c
```

# ЗАДАНИЕ 2

Вывод теперь тоже не один символ, а все символы из maxC:

```
for m in maxC:  
    print(m, maxL)
```

```
with open( "2.txt", "r" ) as F:
```

```
    s = F.readline()
```

```
    p = ""
```

```
    L = 0
```

```
    maxL = 0
```

```
    maxC = ""
```

```
    for c in s:
```

```
        if c == p:
```

```
            L += 1
```

```
        else:
```

```
            L = 1
```

```
        if L > maxL:
```

```
            maxL = L
```

```
            maxC = p
```

```
        elif L == maxL:
```

```
            maxC = maxC + c
```

```
        p = c
```

```
    for m in maxC:
```

```
        print(m, maxL)
```

**Отве**

**T:**

**2 3**

**7 3**

**7 3**

**5 3**

# САМОСТОЯТЕЛЬНО

Текстовый файл `s2.txt` состоит не более чем из  $10^6$  символов. Определите максимальное количество идущих подряд символов, среди которых **каждые два соседних различны**.



# САМОСТОЯТЕЛЬНО

82

Нам больше не нужно сохранять `maxC` (можно удалить эту переменную из кода) и условие цепочки: каждые два соседних символа **различны**, т.е. цепочка увеличивается, если  **$c \neq p$**

```
with open( "s2.txt", "r" ) as F:
    s = F.readline()
p = ""
L = 0
maxL = 0
for c in s:
    if c != p:
        L += 1
    else:
        L = 1
    if L > maxL:
        maxL = L
    p = c
print(maxL)
```

**Ответ:**  
**82**

# ТИП 2

цепочка, каждый символ которой удовлетворяет  
каким-то ограничениям

# ЗАДАНИЕ 3

В текстовом файле **3.txt** находится цепочка из символов латинского алфавита A, B, C. Найдите длину самой длинной подцепочки, состоящей из символов C.

# ЗАДАНИЕ 3

Задача очень похожа на задачу 1-го типа, но теперь символ должен быть равен не предыдущему, а символу 'C':

```
if c == 'C':
```

Предыдущий символ нам вообще больше **не нужен**.

Кроме того, если символ не равен 'C', длина L устанавливается не в 1, а в 0.

```
with open( "3.txt", "r" ) as F:
```

```
    s = F.readline()
```

```
L = 0
```

```
maxL = 0 #длину делаем равной 0, т.к. в файле C может не быть
```

```
for c in s:
```

```
    if c == 'C':
```

```
        L += 1
```

```
    else:
```

```
        L = 0
```

```
    if L > maxL:
```

```
        maxL = L
```

```
print(maxL)
```

**Ответ:**

**0**

# САМОСТОЯТЕЛЬНО

В текстовом файле `s3_1.txt` находится цепочка из символов латинского алфавита A, B, C, D, E. Найдите длину самой длинной подцепочки, состоящей из символов A, B или C (в произвольном порядке).

# САМОСТОЯТЕЛЬНО

16

Меняем единственную строчку:

for c in s:

**if (c == 'A') or (c == 'B') or (c == 'C') :**



# САМОСТОЯТЕЛЬНО

В текстовом файле `s3_2.txt` находится цепочка из символов латинского алфавита A, B, C, D, E, F. Найдите длину самой длинной подцепочки, не содержащей символа D.

# САМОСТОЯТЕЛЬНО

44

Меняем единственную строчку:

for c in s:

**if (c != 'D') :**

# САМОСТОЯТЕЛЬНО

В текстовом файле `s3_3.txt` находится цепочка из символов латинского алфавита A, B, C, D, E, F. Найдите длину самой длинной подцепочки, не содержащей гласных букв.

# САМОСТОЯТЕЛЬНО

20

Меняем единственную строчку:

for c in s:

**if (c != 'A') and (c != 'E') :**

# ЗАДАНИЕ 4

В текстовом файле **4.txt** находится цепочка из символов латинского алфавита A, B, C, D, E. Найдите максимальную длину цепочки вида EABEABEABE... (состоящей из фрагментов EAB, последний фрагмент может быть неполным).

# ЗАДАНИЕ 4

В условии сказано, что последний фрагмент может быть неполным, т.е. возможные цепочки:

E

EA

EAB

EABE

и т.д.

При этом начинаться цепочка всегда должна с E, за E всегда должна быть A, за A – всегда B, за B – снова E.

```
with open( "4.txt", "r" ) as F:
```

```
    s = F.readline()
```

```
L = 0
```

```
maxL = 0
```

```
p = 'B' #сделаем в начале p = 'B', чтобы не рассматривать отдельно начало  
цепочки
```

```
for c in s:
```

```
    if (p == 'B' and c == 'E') or (p == 'E' and c == 'A') or (p == 'A' and c == 'B'):
```

```
        L += 1
```

```
        p = c
```

```
    else:
```

```
        L = 0
```

```
        p = 'B' #снова делаем p = 'B', чтобы не рассматривать начало цепочки
```

```
    if L > maxL:
```

```
        maxL = L
```

```
print(maxL)
```

**Ответ:**

**7**

# САМОСТОЯТЕЛЬНО

В текстовом файле **s4.txt** находится цепочка из символов латинского алфавита A, B, C, D, E, F. Найдите максимальную длину цепочки вида DBACDBACDBAC....



# САМОСТОЯТЕЛЬНО

95

# ЗАДАНИЕ 5

В текстовом файле **5.txt** находится цепочка из символов латинского алфавита A, B, C, D, E. Найдите количество цепочек длины 3, удовлетворяющих следующим условиям:

1-й символ – один из символов B, C или D;

2-й символ – один из символов B, D, E, который не совпадает с первым;

3-й символ – один из символов B, C, E, который не совпадает со вторым.

# ЗАДАНИЕ 5

Эту задачу можно решать несколькими способами. Можно сохранять текущий, последний и предпоследние символы и проверять, что для них выполняются условия. Или можно по-другому бежать по символам в строке. Вместо:

```
for c in s:
```

```
....
```

будем использовать цикл:

```
for i in range(0, len(s)):
```

```
....
```

А обращаться к символу будем так: `s[i]`

# ЗАДАНИЕ 5

Берём  $i$ -й символ,  $s[i]$ . Пусть это будет первый символ цепочки. Тогда он должен быть равен либо B, либо C, либо D:

```
for i in range(0, len(s)):
    if (s[i] == 'B') or (s[i] == 'C') or (s[i] == 'D'):
        ....
```

Следующий символ имеет номер  $i + 1$  и для него по условию задачи должно выполняться условие:  $s[i + 1]$  – один из символов B, D, E, который не совпадает с первым.

```
for i in range(0, len(s)):
    if (s[i] == 'B') or (s[i] == 'C') or (s[i] == 'D'):
        if ( (s[i+1] == 'B') or (s[i+1] == 'D') or (s[i+1] == 'E') ) and (s[i] != s[i + 1]):
            .....
```

# ЗАДАНИЕ 5

Третий символ имеет номер  $i + 2$  и для него по условию задачи должно выполняться условие:  $s[i + 2]$  – один из символов B, C, E, который не совпадает со вторым.

```
for i in range(0, len(s)):
```

```
    if (s[i] == 'B') or (s[i] == 'C') or (s[i] == 'D'):
```

```
        if ( (s[i+1] == 'B') or (s[i+1] == 'D') or (s[i+1] == 'E') ) and (s[i] != s[i + 1]):
```

```
            if ((s[i+2] == 'B') or (s[i+2] == 'C') or (s[i+2] == 'E') ) and (s[i+1] != s[i+2]):
```

```
                .....
```

Если все условия выполняются, то нужно увеличить количество найденных цепочек. В начале оно равно 0.

Сейчас программа будет вылетать с ошибкой, т.к. проходятся все символы с первого до последнего, и для каждого символа мы пытаемся найти следующие два (но **для последнего и предпоследнего символов нельзя взять следующие два**).

Исправляем ошибку: `for i in range(0, len(s) - 2):`

```
with open( "5.txt", "r" ) as F:
```

```
    s = F.readline()
```

```
    count = 0
```

```
    for i in range(0, len(s) - 2):
```

```
        if (s[i] == 'B') or (s[i] == 'C') or (s[i] == 'D'):
```

```
            if ( (s[i+1] == 'B') or (s[i+1] == 'D') or (s[i+1] == 'E') ) and (s[i] != s[i + 1]):
```

```
                if ((s[i+2] == 'B') or (s[i+2] == 'C') or (s[i+2] == 'E') ) and (s[i+1] != s[i+2]) :
```

```
                    count += 1
```

```
    print(count)
```

**Ответ:**

**1280**

# САМОСТОЯТЕЛЬНО

В текстовом файле `s5_1.txt` находится цепочка из символов латинского алфавита A, B, C, D, E, F. Найдите количество цепочек длины 3, удовлетворяющих следующим условиям:

2-й символ – один из B, D, E;

3-й символ – один из A, C, D, который не совпадает со вторым;

1-й символ – совпадает с третьим.

# САМОСТОЯТЕЛЬНО

Ответ: 362

```
for i in range(0, len(s) - 2):
```

```
    if s[i] == s[i+2]:
```

```
        if (s[i+1] == 'B') or (s[i+1] == 'D') or (s[i+1] == 'E') :
```

```
            if ((s[i+2] == 'A') or (s[i+2] == 'C') or (s[i+2] == 'D') ) and (s[i+1] != s[i+2]) :
```

```
                count += 1
```



# САМОСТОЯТЕЛЬНО

В текстовом файле `s5_2.txt` находится цепочка из символов латинского алфавита A, B, C, D, E, F. Найдите количество цепочек длины 5, в которых соседние символы не совпадают.

# САМОСТОЯТЕЛЬНО

Ответ: 4904

```
for i in range(0, len(s) - 4):
```

```
    if (s[i] != s[i+1]) and (s[i+1] != s[i+2]) and (s[i+2] != s[i+3]) and (s[i+3] != s[i+4]) :
```

```
        count += 1
```

# **ТИП 3**

работа с кодами символов

# КОД СИМВОЛА

В компьютере каждому символу соответствует какой-то код. Этот код **не равен** номеру символа в алфавите! Например, английской букве 'А' соответствует код 65, букве 'В' – код 66, букве 'С' – 76 ... букве 'Z' – 90. Более того, цифрам, которые хранятся как символы, тоже соответствует какой-то код, **не совпадающий** с самой цифрой. Цифра '0' имеет код 48, цифра '1' – 49, цифра '9' – 57. Коды цифр идут друг за другом, также как коды заглавных букв и коды строчных букв:

буквам А – Z соответствуют коды 65-90

буквам a-z соответствуют коды 97-122

цифрам 0-9 соответствуют коды 48-57

Обратите внимание: строчные буквы не идут сразу следом за заглавными!

# КОД СИМВОЛА

Чтобы не запоминать код конкретного символа, можно воспользоваться функцией `ord`:

```
print( ord('A') )    # эта строчка на консоль выведет 65
```

Если символ `c1` идёт в алфавите до символа `c2`, то:

```
ord(c2) – ord(c1) == -1
```

Если символ `c1` идёт в алфавите после символа `c2`, то:

```
ord(c1) – ord(c2) == -1
```

# ОПРЕДЕЛЕНИЕ СИМВОЛА ПО КОДУ

Предположим, у Вас есть код символа. Чтобы определить, какому символу соответствует этот код, используется функция `chr`:

```
print( chr(65) ) # эта строчка выведет на консоль букву 'A'
```

```
N = ord('A') + 25
```

```
print( chr(N) ) # эта строчка выведет на консоль букву 'Z'
```

# ЗАДАНИЕ 6

В текстовом файле **6.txt** находится цепочка из прописных (заглавных) символов латинского алфавита A, B, C, D, E, F. Найдите количество подцепочек из трех символов, в которых символы идут в обратном алфавитном порядке и номер начала последней найденной подцепочки (первый символ исходной цепочки имеет номер 0).

Например, у цепочки FDCBA таких подцепочек три: FDC, DCB и CBA, а номер начала последней найденной подцепочки (CBA) два и, следовательно, ответ: 3 2.

# ЗАДАНИЕ 6

Берём  $i$ -й символ,  $s[i]$ . Пусть это будет первый символ цепочки. Его код равен  $\text{ord}(s[i])$ . Если следующий за ним символ  $s[i+1]$  стоит в алфавите до  $s[i]$  (необязательно ПРЯМО перед  $s[i]$ : например, буква А стоит до буквы Р, но не прямо перед ней):

$$\text{ord}(s[i]) > \text{ord}(s[i+1])$$

То же самое верно для символа с  $s[i+2]$ : если в алфавите он стоит до символа  $s[i+1]$ , то:

$$\text{ord}(s[i+1]) > \text{ord}(s[i+2])$$

Примечание: если бы в условии задачи было сказано, что  $s[i+1]$  в алфавите стоит ПРЯМО перед  $s[i]$ , то условие выглядело бы так:  $\text{ord}(s[i]) - \text{ord}(s[i+1]) == 1$

В цикле, в котором проходятся все символы строки, не забываем вычесть из длины строки 2 последних символа: длина цепочки равна 3, до последних символов мы не доходим!

Получаем следующий цикл:

```
for i in range(0, len(s) - 2):
```

```
    if ( ord( s[i] ) > ord( s[i+1] ) ) and ( ord( s[i+1] ) > ord( s[i+2] ) ):
```

....



# ЗАДАНИЕ 6

В задании просили найти количество цепочек и номер начала последней цепочки. Количество цепочек найти легко:

```
count = 0
```

```
for i in range(0, len(s) - 2):
```

```
    if ( ord( s[i] ) > ord( s[i+1] ) ) and ( ord( s[i+1] ) > ord( s[i+2] ) ):
```

```
        count += 1
```

Номер начала последней цепочки найти сложнее. Заметим, что  $i$  – номер начала текущей цепочки. Тогда если цепочка нам подходит, то  $lastBeg = i$  (переприсваиваем начало последней цепочки):

```
lastBeg = 0
```

```
count = 0
```

```
for i in range(0, len(s) - 2):
```

```
    if ( ord( s[i] ) > ord( s[i+1] ) ) and ( ord( s[i+1] ) > ord( s[i+2] ) ):
```

```
        count += 1
```

```
        lastBeg = i
```

```
with open( "6.txt", "r" ) as F:
```

```
    s = F.readline()
```

```
lastBeg = 0
```

```
count = 0
```

```
for i in range(0, len(s) - 2):
```

```
    if ( ord( s[i] ) > ord( s[i+1] ) ) and ( ord( s[i+1] ) > ord( s[i+2] ) ):
```

```
        count += 1
```

```
            lastBeg = i
```

```
print(count, lastBeg)
```

**Ответ: 18 145**

# САМОСТОЯТЕЛЬНО

В текстовом файле `s6_1.txt` находится цепочка из прописных (заглавных) символов латинского алфавита A, B, C, D, E, F. Найдите количество подцепочек из трех символов, в которых символы идут не в убывающем алфавитном порядке и номер начала последней найденной подцепочки (первый символ исходной цепочки имеет номер 0).

Например, у цепочки ABCFF таких подцепочек три: ABC, BCF и CFF, а номер начала последней найденной подцепочки (CFF) два и, следовательно, ответ: 3 2.

# САМОСТОЯТЕЛЬНО

Ответ: 72 148

Условие для не убывающего порядка:

if ( ord( s[i] ) <= ord( s[i+1] ) ) and ( ord( s[i+1] ) <= ord( s[i+2] ) ) ):

# САМОСТОЯТЕЛЬНО

В текстовом файле `s6_2.txt` находится цепочка из прописных (заглавных) символов латинского алфавита A, B, C, D, E, F. Найдите количество подцепочек из трех символов, в которых средний символ ближе к концу алфавита, чем символ слева и справа от него, а также найдите номер начала последней найденной подцепочки (первый символ исходной цепочки имеет номер 0). Например, у цепочки ACBFAED таких подцепочек три: ACB, BFA и AED, а номер начала последней найденной подцепочки (AED) четыре и, следовательно, ответ: 3 4.

# САМОСТОЯТЕЛЬНО

Ответ: 19 156

Если средний символ ближе к концу алфавита, чем символ слева и справа от него, то условие в цикле выглядит так:

```
if ( ord( s[i] ) <= ord( s[i+1] ) ) and ( ord( s[i+1] ) >= ord( s[i+2] ) ):
```

# ЗАДАНИЕ 7

Текстовый файл **7.txt** содержит только заглавные буквы латинского алфавита (A...Z). Определите **первую** подцепочку максимальной длины, в которой символы стоят в возрастающем алфавитном порядке.

Например, у цепочки QABCD A максимальная возрастающая подцепочка - ABCD.

# ЗАДАНИЕ 7

Общая схема решения задачи выглядит следующим образом:

1) если очередной символ больше предыдущего (стоит ближе к концу алфавита, чем предыдущий), то подцепочка увеличивается на символ  $c$ :

if ( $p < c$ ) or ( $p == ''$ ):

$$L = L + c$$

Здесь  $p$  – предыдущий символ,  $c$  – текущий символ,  $L$  – текущая подцепочка.



# ЗАДАНИЕ 7

2) если очередной символ меньше или равен предыдущему, то подцепочка обрывается и становится равна  $c$  (текущему символу).

И в случае, когда подцепочка продолжается, и в случае, когда подцепочка обрывается, текущий символ становится предыдущим:

**for  $c$  in  $s$ :** #для каждого символа в  $s$

if ( $p < c$ ) or ( $p == ''$ ):

$L = L + c$

**else:**

$L = c$

**$p = c$**

# ЗАДАНИЕ 7

Не хватает только сохранения максимальной подцепочки:

```
for c in s:
```

```
    if (p < c) or (p == ""):
```

```
        L = L + c
```

```
    else:
```

```
        L = c
```

```
    if len(maxL) < len(L):
```

```
        maxL = L
```

```
    p = c
```

Полная программа приведена на следующем слайде.

```
with open( "7.txt", "r" ) as F:
```

```
    s = F.readline()
```

```
    p = ""
```

```
    maxL = ""
```

```
    L = ""
```

```
    for c in s:
```

```
        if (p < c) or (p == ""):
```

```
            L = L + c
```

```
        else:
```

```
            L = c
```

```
        if len(maxL) < len(L):
```

```
            maxL = L
```

```
        p = c
```

```
    print(maxL)
```

**Ответ: BEFGTUV**

# САМОСТОЯТЕЛЬНО

Текстовый файл **7.txt** содержит только заглавные буквы латинского алфавита (A...Z). Определите **длину** подцепочки максимальной длины, в которой символы стоят в **убывающем** алфавитном порядке.

Например, у цепочки QDCBAA длина максимальной убывающей подцепочки - 4.

```
with open( "7.txt", "r" ) as F:
```

```
    s = F.readline()
```

```
    p = ""
```

```
    maxL = 0
```

```
    L = 0
```

```
    for c in s:
```

```
        if (p > c) or (p == ""):
```

```
            L = L + 1
```

```
        else:
```

```
            L = 1
```

```
        if maxL < L:
```

```
            maxL = L
```

```
        p = c
```

```
    print(maxL)
```

**Ответ: 6**

# **ТИП 4**

подсчёт количества букв через массив счётчиков

# ЗАДАНИЕ 8

Текстовый файл 7.txt содержит только заглавные буквы латинского алфавита (A...Z). Определите символ, который чаще всего встречается в файле сразу после буквы E, и количество таких символов. Если таких символов несколько – выведите первый. Например, в тексте EВСЕЕВЕДDD после буквы E два раза стоит B, по одному разу – E и D. Для этого текста ответ будет B 2.

# ЗАДАНИЕ 8

В этой задаче потребуется использовать **массив счётчиков**. В латинском алфавите 26 букв, в тексте используются только заглавные буквы => массив должен хранить в себе 26 различных чисел.

Создаём массив на 26 чисел и заполняем его нулями:

```
count = [0] * 26
```

Теперь пробежимся по всем символам строки, и если предыдущий символ `p` равен 'E', увеличим счётчик для текущего символа `c`. Увеличение счётчика будет выглядеть так:

```
if p == 'E':
```

```
    ind = ord(c) - ord('A') #определили номер счётчика
```

```
    count [ ind ] += 1     #увеличили нужный счётчик
```

Функция `ord` возвращает код символа. Код символа 'A' равен 65, код символа 'B' – 66, 'C' – 67 ... 'Z' – 90. Букве 'A' соответствует нулевой счётчик, букве 'B' – первый, букве 'C' – второй ... букве 'Z' – 25 счётчик.



# ЗАДАНИЕ 8

Если мы увеличили счётчик, нужно тут же проверить, не найдена ли новая самая часто встречающаяся буква:

```
count [ ind ] += 1
```

```
if maxCount < count[ ind ]:
```

```
    ...
```

Если действительно буква с встречается чаще всего, изменим maxCount и самую часто встречающуюся букву maxC:

```
if maxCount < count[ ind ]:
```

```
    maxC = c
```

```
    maxCount = count[ ind ]
```

```
with open( "7.txt", "r" ) as F:
```

```
    s = F.readline()
```

```
    p = ""
```

```
    maxC = ""
```

```
    maxCount = 0
```

```
    count = [0] * 26
```

```
    for c in s:
```

```
        if p == 'E':
```

```
            ind = ord(c) - ord('A')
```

```
            count[ ind ] += 1
```

```
            if maxCount < count[ind]:
```

```
                maxC = c
```

```
                maxCount = count[ind]
```

```
        p = c
```

```
print(maxC, maxCount)
```

**Ответ: Y 25**

# САМОСТОЯТЕЛЬНО

Текстовый файл **7.txt** содержит только заглавные буквы латинского алфавита (A...Z). Определите символ, который чаще всего встречается между буквами D и F, и количество таких символов. Если таких символов несколько – выведите первый. Например, в тексте DAFADUFFFFI символ A дважды стоит между символами D и F, символ U – один раз. Для этого текста ответ будет A 2.

ОТВЕТ: D 4

```
with open( "7.txt", "r" ) as F:
```

```
    s = F.readline()
```

```
    p = ""
```

```
    maxC = ""
```

```
    maxCount = 0
```

```
    count = [0] * 26
```

```
    for i in range(1, len(s) - 1):
```

```
        if (s[i-1] == 'D') and (s[i+1] == 'F') or (s[i-1] == 'F') and (s[i+1] == 'D'):
```

```
            ind = ord(s[i]) - ord('A')
```

```
            count[ ind ] += 1
```

```
            if maxCount < count[ind]:
```

```
                maxC = s[i]
```

```
                maxCount = count[ind]
```

```
    p = s[i]
```

```
print(maxC, maxCount)
```