



Все шаблонные программы для решения заданий ЕГЭ

ВАЖНО! Шаблоны могут помочь в решении многих задач, но ограничиваться только ими НЕЛЬЗЯ! Важно понимать все темы глубинно, потому что на экзамене будут встречаться и измененные формулировки задач.

Задание 2

Прототип 1 (нулевые строки):

Логическая функция F задаётся выражением

$$\neg(x \rightarrow w) \vee (y \rightarrow z) \vee \neg y.$$

На рисунке приведён частично заполненный фрагмент таблицы истинности функции F , содержащий неповторяющиеся строки. Определите, какому столбцу таблицы истинности функции F соответствует каждая из переменных x, y, z, w .

?	?	?	?	F
	1		0	0
	0	1		0
		0		0

Шаблон 1 (через вложенные форы):

```
print("x y z w")
for x in range(2):
    for y in range(2):
        for z in range(2):
            for w in range(2):
                # тут пишем функцию из задания
                f = (not (x <= w)) or (y <= z) or (not y)
                if not f:
                    print(x, y, z, w)
```



Шаблон 2 (через dict() и zip()):

```
def f(x, y, z, w):  
    # тут пишем функцию из задания  
    return (not (x <= w)) or (y <= z) or (not y)  
  
from itertools import *  
  
for a1, a2, a3, a4, a5, a6, a7 in product([0, 1], repeat = 7):  
    # составляем таблицу из задания  
    t = [(a1, 1, a2, 0), (a3, 0, 1, a4), (a5, a6, 0, a7)]  
    if len(t) == len(set(t)):  
        for p in permutations("xyzw"):  
            if [f(**dict(zip(p, s))) for s in t] == [0, 0, 0]:  
                print(*p)
```

Прототип 2 (единичные строки):

Логическая функция F задаётся выражением $(y \rightarrow x) \wedge \neg z \wedge w$. На рисунке приведён частично заполненный фрагмент таблицы истинности функции F , содержащий неповторяющиеся строки. Определите, какому столбцу таблицы истинности функции F соответствует каждая из переменных x, y, z, w .

?	?	?	?	F
1	0			1
1	1			1
	1	1		1



Шаблон 1 (через модуль itertools):

```
print("x y z w")
from itertools import *
for x, y, z, w in product([0, 1], repeat = 4):
    # тут пишем функцию из задания
    f = (y <= x) and (not z) and w
    if f:
        print(x, y, z, w)
```

Шаблон 2 (через dict() и zip()):

```
def f(x, y, z, w):
    # тут пишем функцию из задания
    return (y <= x) and (not z) and w

from itertools import *

for a1, a2, a3, a4, a5, a6 in product([0, 1], repeat = 6):
    # составляем таблицу из задания
    t = [(1, 0, a1, a2), (1, 1, a3, a4), (a5, 1, 0, a6)]
    if len(t) == len(set(t)):
        for p in permutations("xyzw"):
            if [f(**dict(zip(p, s))) for s in t] == [1, 1, 1]:
                print(*p)
```

Прототип 3 (2 функции, в значениях есть как 1, так и 0 – редкий прототип, но может встретиться):

Логические функции F_1 и F_2 задаются выражениями

$$F_1 = (w \rightarrow z) \equiv (y \rightarrow x), \quad F_2 = (w \rightarrow z) \wedge (\neg x \equiv y).$$

На рисунке приведён частично заполненный фрагмент таблицы истинности этих функций, содержащий неповторяющиеся строки. Определите, какому столбцу таблицы истинности соответствует каждая из переменных x, y, z, w .

Шаблон:

```
print("x y z w f1 f2")
from itertools import *
for x, y, z, w in product([0, 1], repeat = 4):
    # тут пишем функции из задания
    f1 = (w <= z) == (y <= x)
    f2 = (w <= z) and ((not x) == y)
    print(x, y, z, w, int(f1), int(f2))
```

Задание 8

Прототип 1 (слова под номерами):

Все пятибуквенные слова, составленные из букв Ф, О, К, У, С записаны в алфавитном порядке и пронумерованы. Вот начало списка:



1. KKKKK
2. KKKKO
3. KKKKS
4. KKKKY
5. KKKKF

”

Под каким номером в списке идёт последнее слово, которое не содержит букв Ф и содержит ровно две буквы У?

Шаблон (через вложенные форы):

```
num = 0
# сколько букв в словах, столько и форов
# буквы в строке надо написать в алфавитном порядке
for l1 in "КОСУФ":
    for l2 in "КОСУФ":
        for l3 in "КОСУФ":
            for l4 in "КОСУФ":
                for l5 in "КОСУФ":
                    w = l1 + l2 + l3 + l4 + l5
                    num += 1
                    # тут пишем условие из вопроса
                    if w.count('Ф') == 0 and w.count('У') == 2:
                        print(n)
                        # берем последний выведенный номер
```

Прототип 2 (слова):

Сергей составляет 6-буквенные коды из букв К, А, Л, И, Й. Буква Й может использоваться в коде не более одного раза, при этом она не может стоять на первом месте, на последнем месте и рядом с буквой И. Все остальные буквы могут встречаться произвольное количество раз или не встречаться совсем. Сколько различных кодов может составить Сергей?

Шаблон (через модуль itertools):

```
from itertools import *
k = 0
for w in product("КАЛИЙ", repeat = 6):
    s = "".join(w)
    # тут расписываем все условия
    if s.count("Й") <= 1 and s[0] != "Й" and s[-1] != "Й" \
        and "ЙИ" not in s and "ИЙ" not in s:
        k += 1
print(k)
```



Прототип 3 (числа):

Сколько существует чисел, восьмеричная запись которых состоит из пяти цифр, не начинается с нечётных цифр, не оканчивается цифрами 2 и 6 и содержит не более двух цифр 7?

Шаблон:

```
from itertools import *
k = 0
# задаем алфавит нужной СС
for w in product('01234567', repeat = 5):
    if w[0] not in '01357' and w[-1] not in '26' and \
        w.count('7') <= 2:
        k += 1
print(k)
```

Прототип 4 (перестановки):

Оля составляет слова путём перестановки букв слова «ТИМАШЕВСК». Она выбирает слова, которые начинаются и заканчиваются согласной буквой и только две гласные стоят рядом. Сколько таких слов может написать Оля?

Шаблон:

```
from itertools import *
gl = "АИЕ"
sogl = "ТМШВСК"
k = 0
for w in permutations("ТИМАШЕВСК"):
    s = ''.join(w)
    for i in range(1, len(s)-3):
        if s[0] in sogl and s[-1] in sogl and s[i] in gl \
            and s[i+1] in gl and s[i+2] in sogl:
            k+=1

print(k)
```



Задание 12

Прототип 1 (просто прогон алгоритма на конкретной строке):

Дана программа для Редактора:

ПОКА нашлось (33333) ИЛИ нашлось (999)

ЕСЛИ нашлось (33333)

ТО заменить (33333, 99)

ИНАЧЕ заменить (999, 3)

КОНЕЦ ПОКА

Какая строка получится в результате применения приведённой ниже программы к строке, состоящей из 100 идущих подряд цифр 9?

Шаблон:

```
# тут собираем строку из задания
s = "9" * 100
# тут переписываем алгоритм из задания
while "33333" in s or "999" in s:
    if "33333" in s:
        s = s.replace("33333", "99", 1)
    else:
        s = s.replace("999", "3", 1)
print(s)
```

Прототип 2 (исходная строка неизвестна):

Дана программа для Редактора:

ПОКА НЕ нашлось(00)

заменить(01, 21022)

заменить(02, 310)

заменить(03, 230112)

КОНЕЦ ПОКА

Известно, что исходная строка начиналась с нуля и заканчивалась нулём, а между ними были только цифры 1, 2 и 3. После выполнения данной программы получилась строка, содержащая 104 единицы, 39 двоек и 83 тройки. Сколько цифр было в исходной строке?



Шаблон:

```
for k1 in range(0, 50):
    for k2 in range(0, 50):
        for k3 in range(0, 50):
            s = "0" + "1" * k1 + "2" * k2 + "3" * k3 + "0"
            while "00" not in s:
                s = s.replace("01", "21022", 1)
                s = s.replace("02", "310", 1)
                s = s.replace("03", "230112", 1 )
            if s.count("1") == 104 and s.count("2") == 39 \
                and s.count("3") == 83:
                print(k1 + k2 + k3 + 2)
                exit()
```



Задание 13

В терминологии сетей TCP/IP маской сети называют двоичное число, которое показывает, какая часть IP-адреса узла сети относится к адресу сети, а какая – к адресу узла в этой сети. Адрес сети получается в результате применения поразрядной конъюнкции к заданному адресу узла и маске сети. Сеть задана IP-адресом 112.160.0.0 и сетевой маской 255.240.0.0. Сколько в этой сети IP-адресов, для которых количество единиц в двоичной записи IP-адреса не кратно 3?

Шаблон (подойдет только для такого или похожего прототипа):

```
from ipaddress import *
k = 0
for ip in ip_network("112.160.0.0/255.240.0.0", 0):
    if bin(int(ip))[2:].count("1") % 3 != 0:
        k += 1
print(k)
```

Задание 14

Прототип 1 (количество или сумма цифр в выражении в некоторой СС):

Значение арифметического выражения

$$3 \cdot 289^{2024} + 81 \cdot 49^{121} - 9 \cdot 16^{81} - 6011$$

записали в системе счисления с основанием 31. Определите сумму цифр с числовым значением, не превышающим 17, в записи этого числа.

Шаблон:

```
x = 3 * 289**2024 + 81 * 49**121 - 9 * 16**81 - 6011
s = 0
while x:
    d = x % 31
    if d <= 17:
        s += d
    x //= 31
print(s)
```

Прототип 2 (найти x в выражении):

Значение арифметического выражения $3^{100} - x$, где x – целое положительное число, не превышающее 2030, записали в троичной системе счисления. Определите наибольшее значение x , при котором в троичной записи числа, являющегося значением данного арифметического выражения, содержится ровно пять нулей. В ответе запишите число в десятичной системе счисления.



Шаблон:

```
def tr(n):  
    s = ""  
    while n:  
        s = str(n % 3) + s  
        n //= 3  
    return s  
  
for x in range(2030, 0, -1):  
    n = 3**100 - x  
    s = tr(n)  
    if s.count("0") == 5:  
        print(x)  
        break
```

Задание 15

Прототип 1 (наибольший отрезок):

На числовой прямой даны два отрезка: $P = [15, 27]$ и $Q = [30, 45]$. Найдите наибольшую возможную длину отрезка A , при котором формула

$$(\neg(x \in P) \vee (x \in Q)) \rightarrow \neg(x \in A) \text{ тождественно}$$

истинна, то есть принимает значение 1 при любом значении переменной x .

Шаблон:

```
line = [0.5 * i for i in range(-1000, 1000)]  
for x in line:  
    P = 15 <= x <= 27  
    Q = 30 <= x <= 45  
    A = 1  
    f = ((not P) or Q) <= (not A)  
    if f:  
        print(x)
```

Прототип 2 (наименьший отрезок):

На числовой прямой даны два отрезка: $P = [25, 98]$, $Q = [1, 42]$. Найдите наименьшую возможную длину отрезка A , при котором формула

$$(x \in Q) \rightarrow (\neg(x \in P) \wedge (x \in Q) \rightarrow (x \in A))$$

тождественно истинна, то есть принимает значение 1 при любом значении переменной x .



Шаблон:

```
line = [0.5 * i for i in range(-1000, 1000)]
for x in line:
    P = 25 <= x <= 98
    Q = 1 <= x <= 42
    A = 0
    f = Q <= (((not P) and Q) <= A)
    if not f:
        print(x)
```

Прототип 3 (делители):

Обозначим через ДЕЛ(n, m) утверждение «натуральное число n делится без остатка на натуральное число m ». Для какого наибольшего натурального числа A логическое выражение

$$\text{ДЕЛ}(x, 33) \rightarrow (\neg \text{ДЕЛ}(x, A) \rightarrow \neg \text{ДЕЛ}(x, 242))$$

тождественно истинно, то есть принимает значение 1 при любом натуральном значении переменной x ?

Шаблон:

```
def Del(n, m):
    return n % m == 0
def f(x, A):
    return Del(x, 33) <= ((not Del(x, A)) <= (not Del(x, 242)))

for A in range(1, 1000):
    if all(f(x, A) for x in range(1, 1000)):
        print(A)
```

Прототип 4 (поразрядная конъюнкция):

Обозначим через $m \& n$ поразрядную конъюнкцию неотрицательных целых чисел m и n . Например, $14 \& 5 = 1110_2 \& 0101_2 = 0100_2 = 4$. Для какого наименьшего натурального числа A формула

$$(x \& 2735 \neq 0) \rightarrow ((x \& 1234 = 0) \rightarrow (x \& A \neq 0))$$

тождественно истинна, то есть принимает значение 1 при любом натуральном значении переменной x ?



Шаблон:

```
def f(x, A):  
    return (x & 2735 != 0) <= ((x & 1234 == 0) <= (x & A != 0))  
  
for A in range(1, 10000):  
    if all(f(x, A) for x in range(1, 10000)):  
        print(A)  
        break
```

Прототип 5 (неравенства):

Для какого наименьшего целого неотрицательного A выражение $(x \cdot y < A) \vee (x < y) \vee (y < x)$ тождественно истинно, то есть принимает значение 1 при любых целых неотрицательных значениях переменных x и y ?

Шаблон:

```
def f(x, y, A):  
    return (x * y < A) or (x < y) or (y < x)  
  
for A in range(0, 1000):  
    if all(f(x, y, A) for x in range(0, 1000) \  
          for y in range(0, 1000)):  
        print(A)  
        break
```

Задания 19-21

Они тоже почти всегда шаблонные, но шаблоны у всех свои, свой навязывать не буду, даю его на курсе :)

Задание 23

Прототип 1 (на увеличение):

У исполнителя имеются три команды, которые обозначены латинскими буквами:

- A. Прибавить 1
- B. Прибавить 2
- C. Прибавить 3

Первая команда увеличивает число на 1, вторая – увеличивает на 2, третья – увеличивает на 3. Программа для исполнителя – это последовательность команд. Сколько существует программ, для которых при исходном числе 5 результатом является число 11, при этом траектория вычислений содержит число 7?



Шаблон:

```
def f(x, y):  
    if x > y:  
        return 0  
    if x == y:  
        return 1  
    return f(x + 1, y) + f(x + 2, y) + f(x + 3, y)  
print(f(5, 7) * f(7, 11))
```

Прототип 2 (на уменьшение):

У исполнителя имеются две команды, которые обозначены латинскими буквами:

- A. Вычесть 2
- B. Найти целую часть от деления на 2

Первая команда уменьшает число на 2, вторая – находит целую часть от деления числа на 2. Программа для исполнителя – это последовательность команд.

Сколько существует программ, для которых при исходном числе 32 результатом является число 1, при этом траектория вычислений содержит число 14?

```
def f(x, y):  
    if x < y:  
        return 0  
    if x == y:  
        return 1  
    return f(x - 2, y) + f(x // 2, y)  
print(f(32, 14) * f(14, 1))
```