Домашняя работа №3

Шамриков Алексей А-05-19

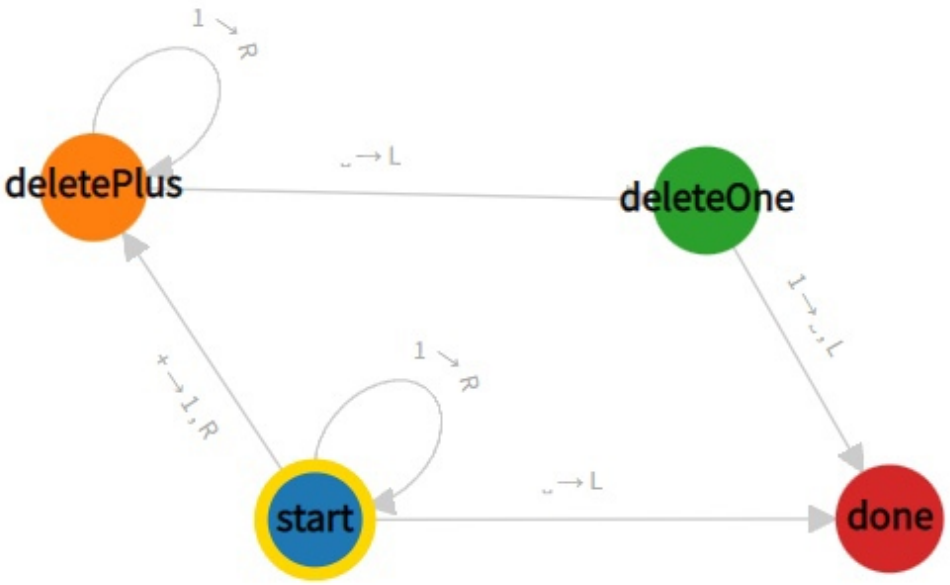
30 мая 2022 г.

1 Машины Тьюринга

1.1 Операции с числами

Реализуйте машины Тьюринга, которые позволяют выполнять следующие операции:

1. Сложение двух унарных чисел

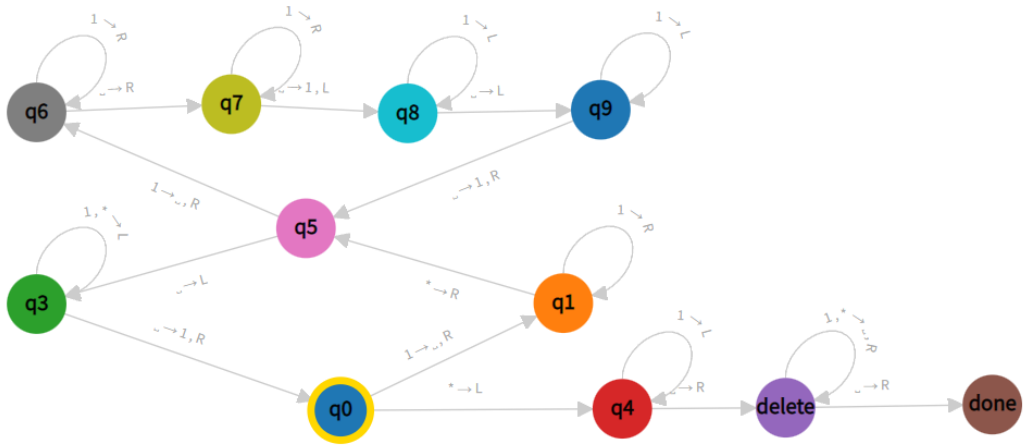


Алгоритм работы такой: проходим по ленте до конца вправо, заменив "+"на единицу. Затем, пройдя до конца, удаляем последнюю единицу.

1+1 -> 111 -> 11

	1	+	’ ’
q0	<q0,x,R>	<q1,’ ’,R>	<done, ’ ’,R>
q1	<q1,1,R>		<q2, ,L>
q2	<done, ,L>		

2. Умножение унарных чисел



111*11 -> 111111

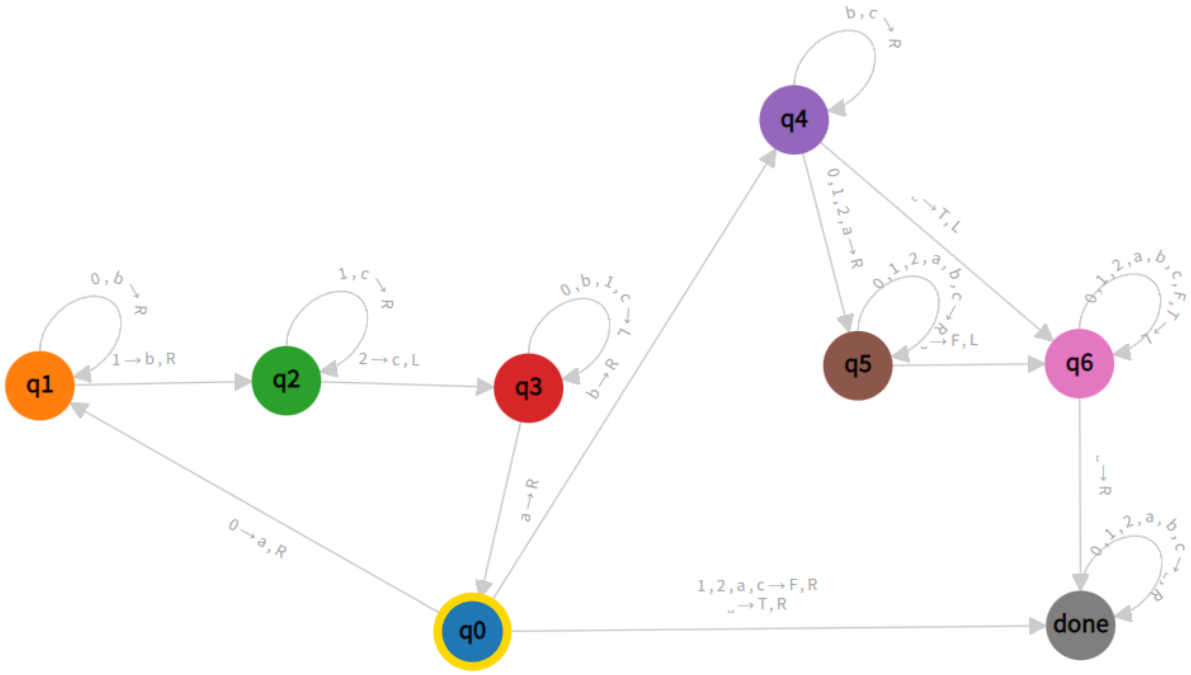
Алгоритм:
Сначала мы получаем первую единицу левого множителя, очищяем ее и идем до оператора произведения.Все, что стоит за оператором мы начинаем переносить вправо по такому принципу: между начальной строкой и скопированными единицами должен быть разделительный пробел. Когда мы переносим единицу, мы оцищаем клетку, однако мы восполяем ее снова, когда начинаем переносить следующую единицу. Когда мы полснотью скопировали правый множитель, мы возвращаемся к первому, восстанавливая первую его единицу и очищая вторую. Так происходит до тех пор, пока мы не пройдем весь левый множитель. Затем мы очищаем все, что находится левее разделительного пробела.

	1	*	' '
q0	<q1, ,R>	<q4,1,L>	
q1	<q1,1,R>	<q5,*,R>	
q3	<q3,1,L>	<q3,*,L>	
q4	<q4,1,L>		<delete,' ',R>
q5	<q6, ,R>		
q6	<q6,1,R>		<q7, ,R>
q7	<q7,1,R>		<q8,1,L>
q8	<q8,1,L>		<q9, ,L>
q9	<q9,1,L>		<q5,1,R>
delete	<delete, , R>	<delete, , R>	<done, ,R>

1.2 Операции с языками и символами

Реализуйте машины Тьюринга, которые позволяют выполнять следующие операции:

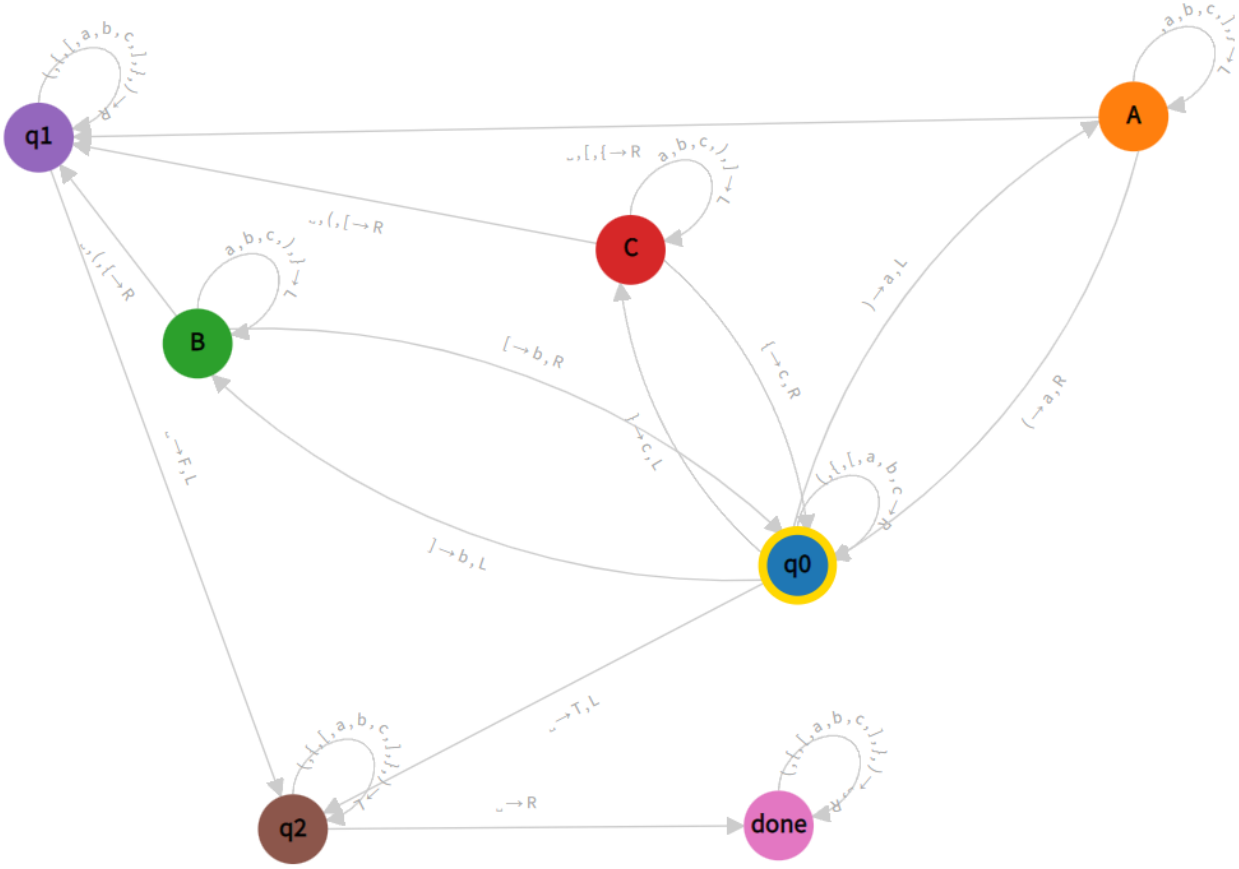
1. Принадлежность к языку $L = \{0^n 1^n 2^n\}, n \geq 0$



Примеры:
001122 -> T
012 -> T
001122 -> T
000111222 -> T
000011112222 -> T
00011222 -> F
000122 -> F

Алгоритм:
Проходим по ленте n раз, заменяя нули на a, единицы на b и двойки на c, причем за один проход мы заменяем только один ноль, одну единицу и одну двойку. Если оказывается, что все единицы и двойки уже были заменены, а нули - нет, тогда мы сразу переходим в терминальное состояние, записывая в ленту F. Если все нули были заменены, то мы прокручиваем все буквы b и c, смещаясь по ленте вправо. Если мы долистали до конца строки, то переходим в терминальное состояние, записывая T, если нет (то есть встречаются цифры) - пролистываем их до конца и записываем F. В терменальном состоянии мы удаляем все, что стоит левее буквы F или T. Получаем ответ.

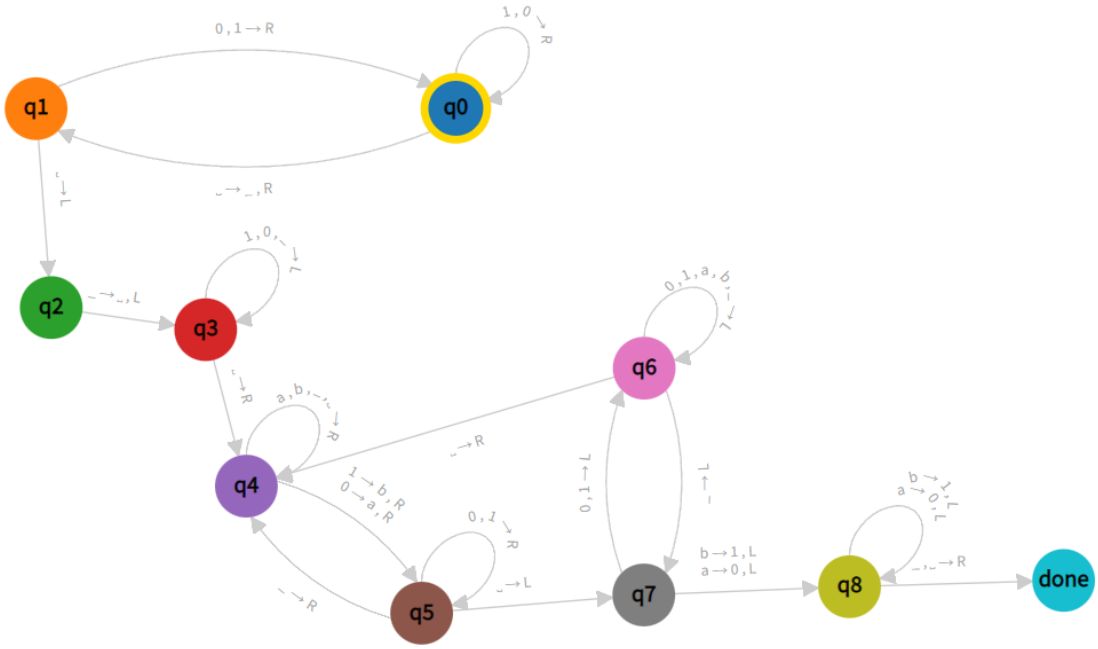
2. Проверка соблюдения правильности скобок в строке (минимум 3 вида скобок) (0.5 балла)



Примеры:
’([([ ])])’ -> T
’( ) [ ] { }’ -> T
’(( )[]{})’ -> T
’([( ])’ -> F
’(({ [ ] } ([ ])’ -> F
’((( ))’ -> F

Алгоритм:
С самого начала мы пропускаем все открывающие скобки. Дойдя до первой закрывающей, мы заменяем ее на соответствующую букву. Затем сдвигаемся влево до тех пор, пока не найдется такая же открывающая скобка. Заменяем ее на ту же букву. Так происходит со всеми скобками. Если мы дошли до начала слова, не найдя открывающей скобки, переходим в терминальное состояние, записав F. Если все хорошо, то доходим до начала слова и начинаем идти вправо, пропуская буквы. Если попадается незатронутая скобка, значит, у нее нет закрывающей пары. Тогда переходим в терминальное состояние, записав F. Если попадаются только буквы, то переходим в терминальное состояние, записав T. В терминальном состоянии мы очищаем с ленты все, что идет левее буквы F или T.

3. Поиск минимального по длине слова в строке (слова состоят из символов 1 и 0 и разделены пробелом)



Алгоритм:
Мы заменяем по одной цифре из каждого слова на буквы. Так мы делаем до тех пор, пока одно из слов не будет состоять только из букв. Если мы остановим замены в этот момент, то это слово и будет самым коротким. Тогда мы проходим по нему, заменив буквы обратно на цифры. Затем мы останавливаем курсор МТ в начале этого слова.
Алгоритм останавливает выполнение в начала искомого слова, т.е. он не выделяет это слово, удалив остальные.

2 Квантовые вычисления

2.1 Генерация суперпозиций 1

Дано N кубитов ($1 \leq N \leq 8$) в нулевом состоянии $|0\dots0\rangle$. Также дана некоторая последовательность битов, которое задаёт ненулевое базисное состояние размера N. Задача получить суперпозицию нулевого состояния и заданного.

$$|S\rangle = \frac{1}{\sqrt{2}}(|0\dots0\rangle + |\psi\rangle)$$

То есть требуется реализовать операцию, которая принимает на вход:

1. Массив кубитов q_s

2. Массив битов *bits* описывающих некоторое состояние $|\psi\rangle$. Это массив имеет тот же самый размер, что и *qs*. Первый элемент этого массива равен 1.

Решение:
Будем испольщовать пакет qiskit для Python.
Функции, необходимые для решения задачи: Для первого кубита всегда применяем оператор Адамара. Затем смотрим на вектор bits, если кубит равен 1, то мы спутываем его с первым кубитом.

```
def inputData():
    N = int(input('Введите N: '))

    q = QuantumRegister(N, name='q')
    bits = []

    print('Input bits (1 - True/ skip - False)')
    for i in range(N):
        bits.append(bool(input()))

    return q, bits
```

Основной код программы:

```
q, bits = inputData()
circuit = QuantumCircuit(q)

task3_1(q, bits)
circuit.draw(output='mpl', filename='task3_1.jpg', initial_state=True)
```

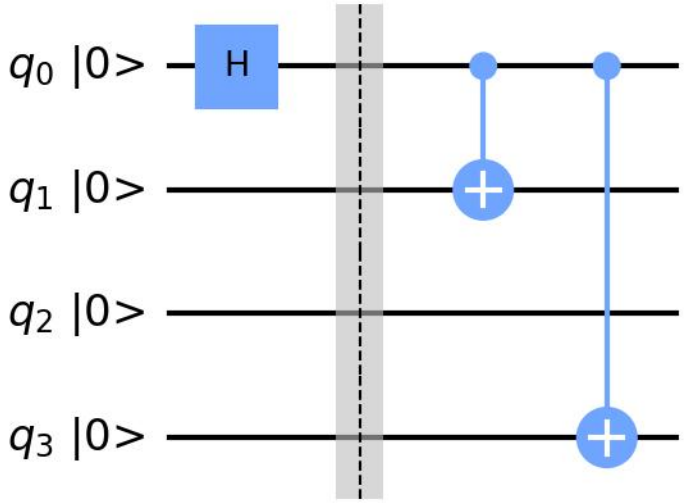
Тест при N = 4:

Результат:

```
Введите N: 4
Input bits (1 - True/ skip - False)
1
1

1

Process finished with exit code 0
```



2.2 Различение состояний 1

Дано N кубитов ($1 \leq N \leq 8$), которые могут быть в одном из двух состояний:

$$|GHZ\rangle = \frac{1}{\sqrt{2}}(|0\ldots0\rangle + |1\ldots1\rangle)$$

$$|W\rangle = \frac{1}{\sqrt{N}}(|10\ldots00\rangle + |01\ldots00\rangle + \cdots + |00\ldots01\rangle)$$

Требуется выполнить необходимые преобразования, чтобы точно различить эти два состояния. Возвращать 0, если первое состояние и 1, если второе.

На вход процедура получает кубитов. Для различения состояний пользуемся следующими соображениями:

Первое состояние при измерении дает все кубиты либо 1, либо 0.

Второе состояние при измерении дает только один купит 1, остальные 0.

Кроме того, нуобходимо учитывать случай N = 1:

$$|GHZ\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

$$|W\rangle = |1\rangle$$

Тогда мы сначала измеряем кубиты, учитываем N = 1. Если количество кубитов равно единице, и заданное состояние равно $|1\rangle$, то сразу определяем его как второе состояние, иначе - как первое. Если кол-во кубитов >1, то считаем единичные кубиты, полученные после измерения. Если в результате имеем только один единичный кубит, то определили второе состояние, иначе - первое.
Решение:
Функция ввода и создания векторов состояний:

```
def inputData3_2():
    N = int(input('Введите N: '))
    # 1
    ghz = [1 / math.sqrt(2)]
    for i in range(1, 2 ** N - 1):
        ghz.append(0)
    ghz.append(1 / math.sqrt(2))
    ghz = Statevector(ghz)

    # 2
    w = [0]
    for i in range(1, 2 ** N):
        if i & i - 1:
            w.append(0)
        else:
            w.append(1 / math.sqrt(N))
    w = Statevector(w)

    return ghz, w
```

Функция, реализующая алгоритм:

```python
def task3_2(q):
    str = q.measure()[0]
    if (len(str) == 1):
        if q == One:
            return 1
        else:
            return 0

    i = 0
    for qubit in str:
        if qubit == '1':
            i += 1
    if (i == 1):
        return 1
    else:
        return 0
```

Основной код:

```python
ghz, w = inputData3_2()
One = Statevector([0, 1])

print(task3_2(ghz))
```

Тест:

```
Введите N: 4
0
```

2.3 Различение состояний 2

Дано 2 кубита, которые могут быть в одном из двух состояний:

$$|S_0\rangle = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$$

$$|S_1\rangle = \frac{1}{2}(|00\rangle - |01\rangle + |10\rangle - |11\rangle)$$

$$|S_2\rangle = \frac{1}{2}(|00\rangle + |01\rangle - |10\rangle - |11\rangle)$$

$$|S_3\rangle = \frac{1}{2}(|00\rangle - |01\rangle - |10\rangle + |11\rangle)$$

Требуется выполнить необходимые преобразования, чтобы точно различить эти четыре состояния. Возвращать требуется индекс состояния (от 0 до 3).

Решение:
Сначала создаем состояния:

```
S_1 = Statevector([1 / 2, 1 / 2, 1 / 2, 1 / 2])
S_2 = Statevector([1 / 2, -1 / 2, 1 / 2, -1 / 2])
S_3 = Statevector([1 / 2, 1 / 2, -1 / 2, -1 / 2])
S_4 = Statevector([1 / 2, -1 / 2, -1 / 2, 1 / 2])
```

Алгоритм будет основываться на том, чтобы применить оператор Адамара для каждого из 2-х кубитов. У нас получиться 4 различных состояния, которые мы впоследствии измерим.
Матрица оператора Адамара для 2-х кубитов:

$$H^2 = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}$$

Переведем операторы состояния в векторный вид и применим оператор Адамара:

$$H^2 |S_1\rangle = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} \frac{1}{2}\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} = \frac{1}{4}\begin{pmatrix} 1+1+1+1 \\ 1-1+1-1 \\ 1+1-1-1 \\ 1-1-1+1 \end{pmatrix} = \frac{1}{4}\begin{pmatrix} 4 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Аналогично для других состояний:

$$|S_1\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}; |S_2\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}; |S_3\rangle = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}; |S_4\rangle = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

Переведем состояния обратно. Получим:

$$|S_1\rangle = |00\rangle; |S_2\rangle = |01\rangle; |S_3\rangle = |10\rangle; |S_4\rangle = |11\rangle$$

Напишем функцию для определения состояния:

```
def task3_3(q):
    global circuit
    circuit.initialize(q)
    circuit.h(range(2))

    state = Statevector(circuit)
    res = state.measure()[0]
    if res[0] == '0':
        if res[1] == '0':
            return 1
        else:
            return 2
    else:
        if res[1] == '0':
            return 3
        else:
            return 4
```

Основной код программы:

```
circuit = QuantumCircuit(2)

print(task3_3(S_2))
```

Результат теста:

```
2

Process finished with exit code 0
```