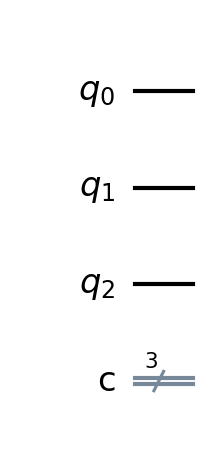
Практическое задание 3

Задача: реализовать схему квантовой телепортации. Проверить программу для различных состояний, которые нужно передать.

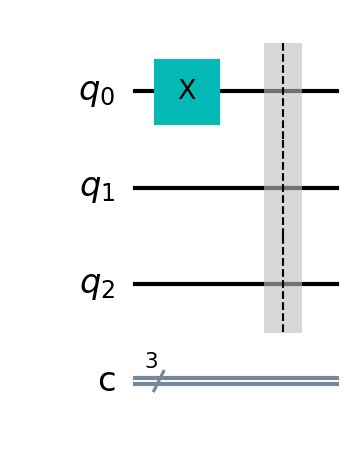
Решение

Алгоритм:

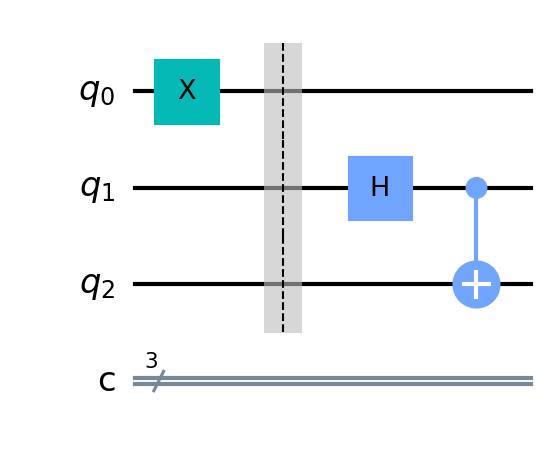
* Создание изначальной схемы



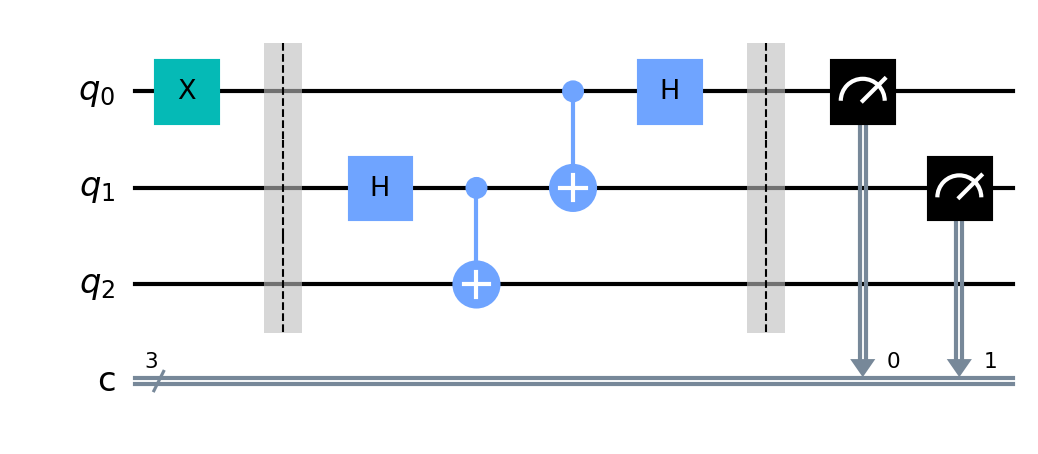
1. Применение к q\_0 (его необходимо телепортировать) гейта X



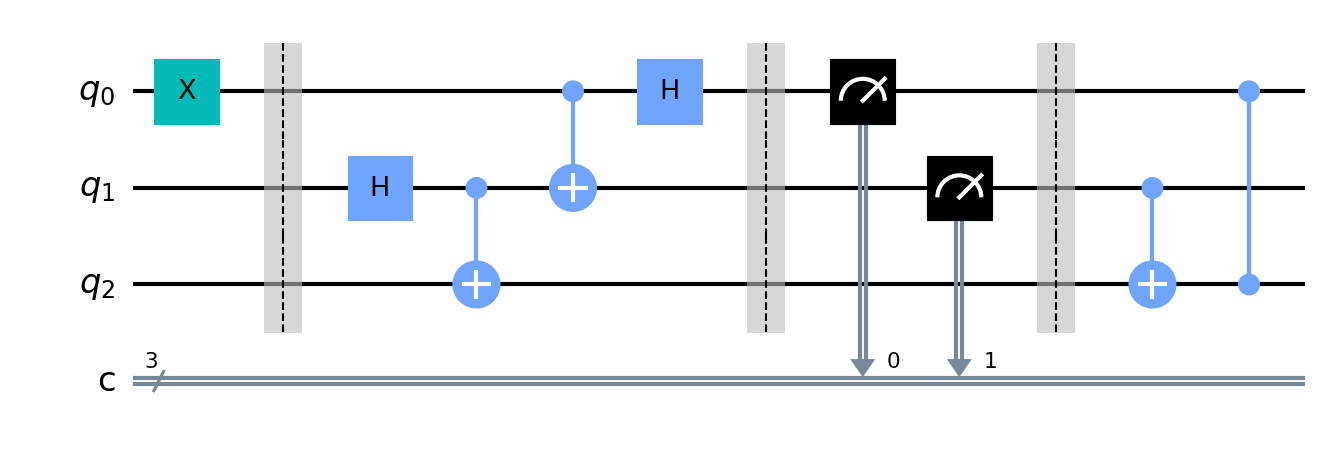
1. Создание запутанности между q\_1 и q\_2, с использованием гейта Адамара к q\_1 и гейта CX к q\_1 и q\_2



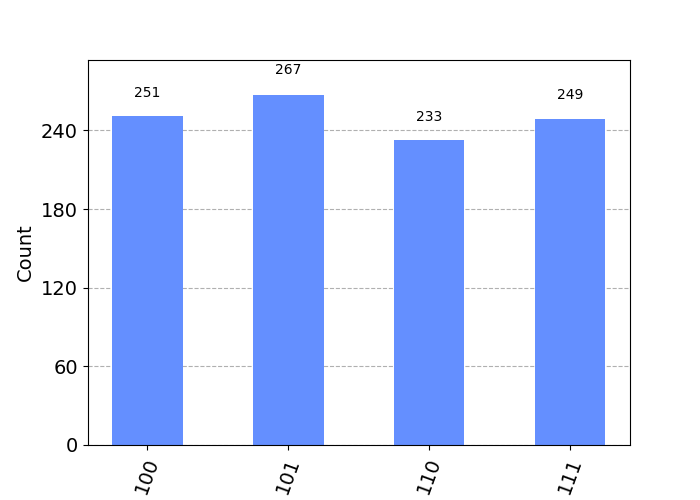
1. Создание запутанности между q\_0 и q\_1



1. Применение гейта CX к q\_1 и q\_2 и гейта CZ к q\_0 и q\_2



1. Вычисление результата и сохранение гистограммы



Код решения:

def task3():  
 # Cоздание квантовой схемы из 3х кубитов и 3х битов и вывод схемы  
 telCirc = QuantumCircuit(3, 3)  
 telCirc.draw(output='mpl', filename='task3\_init\_scheme.png')  
 # Шаг 1: Применение к q\_0 (его необходимо телепортировать) гейта X  
 telCirc.x(0)  
 telCirc.barrier()  
 telCirc.draw(output='mpl', filename='task3\_step1.png')  
 # Шаг 2: Создание запутанности между q\_1 и q\_2, с использованием гейта Адамара к q\_1 и гейта CX к q\_1 и q\_2  
 telCirc.h(1)  
 telCirc.cx(1, 2)  
 telCirc.draw(output='mpl', filename='task3\_step2.png')  
 # Шаг 3: Создание запутанности между q\_0 и q\_1  
 telCirc.cx(0, 1)  
 telCirc.h(0)  
 telCirc.barrier()  
 telCirc.measure([0, 1], [0, 1])  
 telCirc.draw(output='mpl', filename='task3\_step3.png')  
 # Шаг 4: Применение гейта CX к q\_1 и q\_2 и гейта CZ к q\_0 и q\_2  
 telCirc.barrier()  
 telCirc.cx(1, 2)  
 telCirc.cz(0, 2)  
 telCirc.draw(output='mpl', filename='task3\_step4.png')  
 # Вычисление результата и сохранение гистограммы  
 sim = Aer.get\_backend('qasm\_simulator')  
 telCirc.measure(2, 2)  
 result = execute(telCirc, backend=sim, shots=1000).result()  
 counts = result.get\_counts()  
 plot\_histogram(counts)  
 plt.savefig('histogram\_task3.png')