



## МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего  
образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

---

Институт искусственного интеллекта

Базовая кафедра 252

---

## НАУЧНО-ИССЛЕДОВАТЕЛЬСКАЯ РАБОТА

Тема научно-исследовательской работы:

**«Сравнительный анализ библиотек разработки квантовых схем Cirq, QISKit, Forest,  
QDK на примере генерации истинно случайной последовательности»**

Студенты:

Никишина Анна Александровна

Базарова Елизавета Олеговна

Группа:

ККСО–03–19

Руководитель

научно-исследовательской

работы:

Крючков Андрей Андреевич

Москва

2023

# Содержание

<b>Глава 1. Общее введение в квантовую механику</b>	<b>3</b>
1.1. Развитие квантовой механики . . . . .	3
1.1.1 Историческая справка . . . . .	3
1.1.2 Ключевые фигуры/отцы-основатели . . . . .	4
1.1.3 Главные научные работы . . . . .	4
1.2. Основные свойства квантовой механики . . . . .	4
1.3. Кубиты, операторы и измерения . . . . .	6
1.3.1 Кубиты. . . . .	6
1.3.2 Операторы и измерения . . . . .	6
1.3.3 Схемы . . . . .	7
1.3.4 Преимущества кубитов над битами . . . . .	8
<b>Глава 2. Сравнительный анализ библиотек разработки квантовых схем Cirq, QISKit, Forest, QDK на примере генерации истинно случайной последовательности.</b>	<b>9</b>
2.1. Программная реализация генераций истинно случайных последовательностей	9
2.1.1 Cirq . . . . .	9
2.1.2 QISKit . . . . .	11
2.1.3 Forest . . . . .	13
2.1.4 QDK . . . . .	16
2.2. Статистический анализ полученных результатов . . . . .	21
2.2.1 Теоретико-математическая основа статистических исследований . . .	21
2.2.2 Интерпретация результатов . . . . .	23
<b>3. ЗАКЛЮЧЕНИЕ</b>	<b>26</b>
<b>4. СПИСОК ЛИТЕРАТУРЫ</b>	<b>27</b>

# Глава 1. Общее введение в квантовую механику

## 1.1. Развитие квантовой механики

### 1.1.1 Историческая справка

Квантовая механика — это фундаментальная теория в физике, которая описывает поведение материи и энергии на атомном и субатомном уровнях. Теория квантов, в самой ранней своей формулировке, возникла в связи с неспособностью классической физики объяснить экспериментально наблюдаемое распределение энергии в непрерывном спектре излучения абсолютно черного тела[9].

#### Предыстория (19-й век)

- 1800:** Уильям Гершель обнаруживает инфракрасное излучение;
- 1830:** Джеймс Максвелл формулирует теорию электромагнетизма;
- 1860:** Максвелл предсказывает существование электромагнитных волн;
- 1887:** Генрих Герц экспериментально подтверждает существование электромагнитных волн;
- 1895:** Вильгельм Рентген открывает рентгеновские лучи;
- 1900:** Макс Планк предлагает квантование энергии излучения черного тела.

#### Зарождение квантовой механики (1905-1925)

- 1905:** Альберт Эйнштейн объясняет фотоэлектрический эффект, используя кванты света;
- 1913:** Нильс Бор предлагает модель атома, основанную на квантовании энергии;
- 1923:** Луи де Бройль выдвигает гипотезу волн материи;
- 1924:** Вернер Гейзенберг формулирует матричную механику;
- 1926:** Эрвин Шредингер разрабатывает волновое уравнение;
- 1927:** Макс Борн интерпретирует волновую функцию Шредингера как амплитуду вероятности.

#### Развитие квантовой механики (1926-1950)

- 1928:** Поль Дирак предсказывает существование антиматерии;
- 1930:** Джон фон Нейман формулирует квантовую механику в терминах операторов в гильбертовом пространстве;
- 1932:** Вернер Гейзенберг и Пауль Эренфест формулируют принцип неопределенности;
- 1935:** Эрвин Шредингер публикует «Кот Шредингера»;
- 1948:** Ричард Фейнман, Джулиан Швингер и Фримен Дайсон формулируют квантовую электродинамику.

#### Квантовая механика во второй половине 20-го века

- 1950:** Развитие квантовой теории поля;
- 1960:** Развитие квантовой информатики;

- 1970:** Развитие квантовой оптики;
- 1980:** Развитие квантовой криптографии;
- 1990:** Развитие квантовых вычислений.

### **Квантовая механика в 21-м веке**

- 2000:** Экспериментальная реализация квантовых алгоритмов;
- 2010:** Развитие квантовых симуляторов;
- 2020:** Развитие квантовых коммуникаций.

#### **1.1.2 Ключевые фигуры/отцы-основатели**

- **Макс Планк** (1858-1947): немецкий физик, который в 1900 году предложил квантование энергии излучения черного тела
- **Альберт Эйнштейн** (1879-1955): немецкий физик, который в 1905 году объяснил фотоэлектрический эффект, используя кванты света
- **Нильс Бор** (1885-1962): датский физик, который в 1913 году предложил модель атома, основанную на квантовании энергии
- **Вернер Гейзенберг** (1901-1976): немецкий физик, который в 1927 году сформулировал принцип неопределенности
- **Эрвин Шредингер** (1887-1961): австрийский физик, который в 1926 году разработал волновое уравнение, описывающее квантовое состояние системы
- **Поль Дирак** (1902-1984): английский физик, который в 1928 году предсказал существование антиматерии

#### **1.1.3 Главные научные работы**

- «О квантовой теории излучения» (1900) - Макс Планк;
- «Об эвристической точке зрения на производство и превращение света» (1905) - Альберт Эйнштейн;
- «О строении атомов и молекул» (1913) - Нильс Бор;
- «О квантовой механике» (1927) - Вернер Гейзенберг;
- «Квантование волновых полей» (1926) - Эрвин Шредингер;
- «Теория электронов и позитронов» (1928) - Поль Дирак.

## **1.2. Основные свойства квантовой механики**

Квантовая механика – это раздел физики, который описывает поведение материи и энергии на атомном и субатомном уровне. Она основана на ряде фундаментальных принципов, которые отличают ее от классической физики, используемой для описания макроскопических объектов.

- **Корпускулярно-волновой дуализм**

В квантовой механике частицы, такие как фотоны и электроны, могут демонстрировать как свойства частиц, так и свойства волн[6]. Это означает, что они могут вести себя как точечные объекты, как в случае столкновения с атомами, или как волны, как в случае дифракции на решетке. Дуализм означает, что невозможно одновременно описать все свойства квантового объекта с помощью классических представлений о частицах и волнах.

Пример:

- Фотон может вести себя как частица, когда он выбивает электрон из атома в фотоэлектрическом эффекте;
- Фотон может вести себя как волна, когда он дифрагирует на щели.

- **Вероятность и неопределенность**

В квантовой механике невозможно точно предсказать значения некоторых физических величин, таких как импульс и траектории, по которой движется частица. Описание ее движения с помощью координат и проекций импульса будет тем менее точным, чем меньше масса частицы[7]. Вместо этого можно лишь вычислить вероятность того, что частица будет иметь определенное значение. Эта неопределенность связана с принципом Гейзенберга, который гласит, что произведение погрешностей измерения координаты и импульса частицы не может быть меньше, чем постоянная Планка.

Принцип неопределенности Гейзенберга:

- ▶ Невозможно одновременно точно измерить положение и импульс частицы;
- ▶ Чем точнее измеряется одно, тем менее точно можно измерить другое.

Пример:

- Если мы точно измерим положение электрона, мы не сможем точно измерить его импульс, и наоборот.

- **Квантовая запутанность**

Квантовая запутанность - это явление, при котором две частицы связаны между собой таким образом, что изменение состояния одной частицы мгновенно влияет на состояние другой, независимо от расстояния между ними. Это явление не имеет аналога в классической физике и является одним из самых загадочных аспектов квантовой механики. Запутанность имеет важные приложения в квантовой криптографии и квантовых вычислениях.

Пример:

- Если мы запутаем два фотона, а затем измерим поляризацию одного из них, мы мгновенно узнаем поляризацию другого, даже если они находятся на разных концах галактики.

- **Квантовые суперпозиции**

Квантовые частицы в суперпозиции представляются комбинацией всевозможных состояний, до момента, пока не произошло их измерение и наблюдение[6]. Суперпозиция является одним из ключевых принципов квантовой механики, который позволяет объяснить многие явления, такие как атомные спектры.

Пример:

- Фотон может быть одновременно поляризованным по вертикали и по горизонтали;

– Электрон может одновременно находиться в разных точках пространства.

- **Квантовое туннелирование**

Квантовая частица, с движением которой связан волновой вероятностный процесс, в отличие от классической, может проникать из одной классически разрешенной области в другую даже в случае, когда энергия частицы лежит ниже барьера. Это явление называется туннельным эффектом[7]. Оно лежит в основе работы многих современных технологий, таких как туннельные диоды и транзисторы, а также играет важную роль в ядерных реакциях.

### 1.3. Кубиты, операторы и измерения

#### 1.3.1 Кубиты.

Квантовый бит, или кубит, является базовой единицей информации в квантовых вычислениях. В отличие от классических битов, которые могут находиться только в состояниях 0 или 1, кубиты могут находиться в суперпозиции этих состояний. Т.е. кубит может одновременно представлять собой и 0, и 1 с некоторой вероятностью[2].

Математически состояние кубита описывается квантовым вектором, который называется **вектором состояния**. Его можно представить как вектор в двумерном комплексном гильбертовом пространстве.

- $|0\rangle$  и  $|1\rangle$  – это два базисных вектора, которые соответствуют состояниям 0 и 1 классического бита соответственно.
- Любой вектор в этом пространстве, за исключением базисных векторов, представляет собой суперпозицию состояний 0 и 1.

Состояние кубита в суперпозиции двух базисных векторов (0 и 1) может быть записано как:  $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ , где  $\alpha$  и  $\beta$  – комплексные числа, такие, что  $|\alpha|^2 + |\beta|^2 = 1$ .

Квадрат модуля амплитуды  $|\alpha|^2$  дает вероятность того, что при измерении кубит будет находиться в состоянии 0, а  $|\beta|^2$  – вероятность того, что он будет находиться в состоянии 1.

#### 1.3.2 Операторы и измерения

**Квантовые операторы** – это линейные преобразования, которые действуют на векторы состояния кубитов[1].

Некоторые из наиболее важных операторов:

- *Оператор Паули X*: Оператор NOT, переворачивает состояние кубита.
- *Оператор Паули Y*: Оператор bit-flip, меняет местами  $|0\rangle$  и  $|1\rangle$ .
- *Оператор Паули Z*: Оператор фазы, добавляет фазовый сдвиг к состоянию кубита.
- *Оператор H*: Оператор Адамара, преобразует 
$$\begin{cases} |0\rangle \text{ в } |+\rangle = (|0\rangle + |1\rangle)/\sqrt{2} \\ |1\rangle \text{ в } |-\rangle = (|0\rangle - |1\rangle)/\sqrt{2} \end{cases}$$
- *Оператор унитарного преобразования*: Этот оператор сохраняет норму вектора состояния.
- *Оператор измерения*: Этот оператор преобразует кубит в классический бит, возвращая 0 или 1.

- *Оператор Гамильтона*: Этот оператор описывает эволюцию квантовой системы во времени.

**Измерение** – это процесс, в котором квантовая система преобразуется в классическую.

Измерение кубита приводит к его схлопыванию в одно из двух классических состояний, 0 или 1. Вероятность того, что кубит будет измерен в состоянии 0, равна  $|\alpha|^2$ , а вероятность того, что он будет измерен в состоянии 1, равна  $|\beta|^2$ .

**Пример 1.** Рассмотрим кубит, который находится в суперпозиции  $|0\rangle$  и  $|1\rangle$  с равными вероятностями. Если мы измерим этот кубит, то он с равной вероятностью может быть измерен в состоянии 0 или 1. Если мы приложим к этому кубиту *оператор H*, а затем измерим его, то он будет измерен в состоянии  $|+\rangle$  или  $|-\rangle$  с равной вероятностью.

**Пример 2.** Предположим, что кубит находится в состоянии  $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ . Если мы измерим этот кубит, то он с вероятностью  $|\alpha|^2$  примет значение 0 и с вероятностью  $|\beta|^2$  примет значение 1.

### 1.3.3 Схемы

Схемы квантовых алгоритмов – это графические представления, которые используются для их описания. Схемы состоят из квантовых вентилях, которые представляют собой операции над кубитами.

**Сфера Блоха** – это геометрическое представление квантовых состояний.

**Схема квантового компьютера** – это схема, которая показывает, как кубиты используются для выполнения квантовых вычислений.

Существует множество различных типов квантовых вентилях:

- Унитарные вентили:
  - **H** – преобразование Адамара.
  - **CNOT** – управляемый НЕ.
- Эрмитовы вентили:
  - **Z** – фазовый сдвиг.
  - **X** – NOT.
- Проекционные вентили:
  - **M** – измерение.

**Пример 1.** Схема квантового алгоритма для определения четности числа:

```

q0: - - - -
q1: - - - -
H q0
CNOT q0 q1
M q1

```

В этой схеме два кубита q0 и q1.

Первый вентиль H преобразует кубит q0 в суперпозицию состояний 0 и 1.

Второй вентиль CNOT выполняет операцию НЕ над кубитом q1, если кубит q0 находится в состоянии 1.

Третий вентиль M измеряет состояние кубита q1.

Результат измерения будет 0, если число четное, и 1, если число нечетное.

**Пример 2.** Схема квантового алгоритма для генерации случайного числа.

— —  $|0\rangle$  — —  
       $|H|$   
— —  $|?\rangle$  — —

Кубит сначала инициализируется в состоянии  $|0\rangle$

Затем к нему применяется оператор  $H$ , который переводит его в суперпозицию  $|+\rangle$  и  $|-\rangle$ .

Наконец, кубит измеряется, и результат измерения является случайным числом 0 или 1.

#### 1.3.4 Преимущества кубитов над битами

**Сверхпараллелизм:** Квантовые вычисления могут выполняться параллельно на всех кубитах одновременно.

**Экспоненциальное ускорение:** Некоторые задачи могут быть решены квантовыми вычислениями экспоненциально быстрее, чем классическими.

**Устойчивость к шуму:** Квантовые вычисления могут быть более устойчивыми к шуму, чем классические.

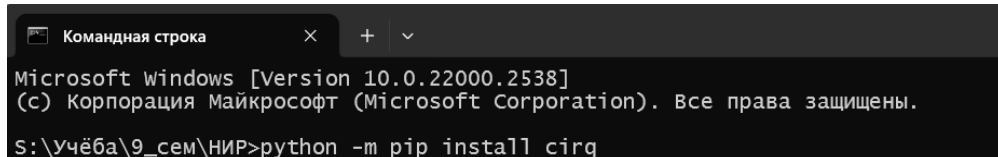


## Глава 2. Сравнительный анализ библиотек разработки квантовых схем Cirq, QISKit, Forest, QDK на примере генерации истинно случайной последовательности.

### 2.1. Программная реализация генераций истинно случайных последовательностей

#### 2.1.1 Cirq

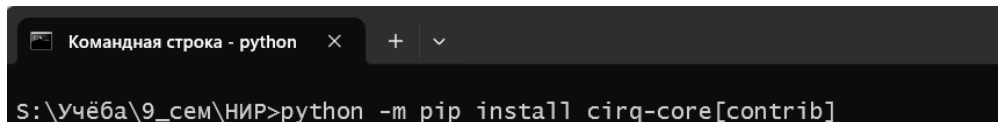
Для начала работы с библиотекой Cirq нам потребуется установить её.



```
Командная строка
Microsoft Windows [Version 10.0.22000.2538]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.
S:\Учёба\9_сем\НИР>python -m pip install cirq
```

Рис. 2.1.1: Установка библиотеки Cirq.

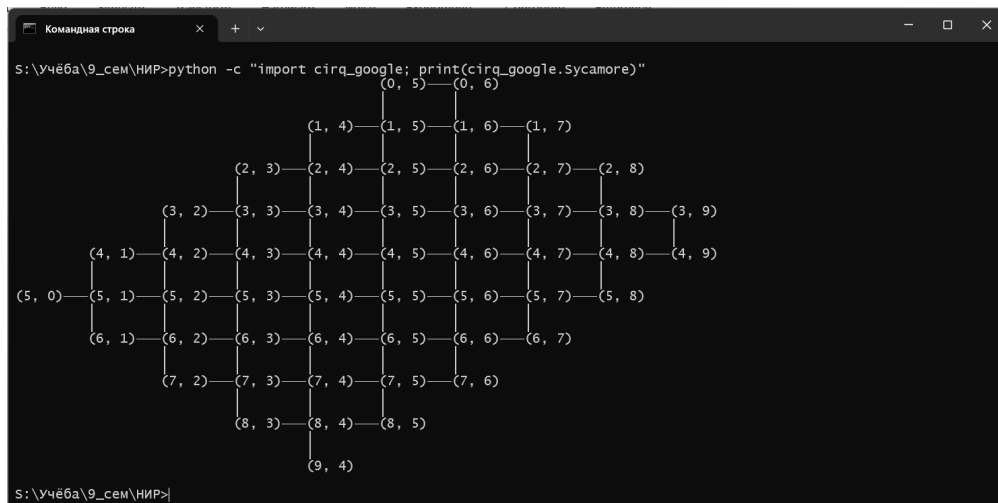
Затем установим зависимости функций в `cirq.contrib`.



```
Командная строка - python
S:\Учёба\9_сем\НИР>python -m pip install cirq-core[contrib]
```

Рис. 2.1.2: Установка зависимости функций.

Далее проверим работоспособность библиотеки.



```
Командная строка
S:\Учёба\9_сем\НИР>python -c "import cirq_google; print(cirq_google.Sycamore)"
(0, 5) — (0, 6)
      (1, 4) — (1, 5) — (1, 6) — (1, 7)
          (2, 3) — (2, 4) — (2, 5) — (2, 6) — (2, 7) — (2, 8)
              (3, 2) — (3, 3) — (3, 4) — (3, 5) — (3, 6) — (3, 7) — (3, 8) — (3, 9)
                  (4, 1) — (4, 2) — (4, 3) — (4, 4) — (4, 5) — (4, 6) — (4, 7) — (4, 8) — (4, 9)
                      (5, 0) — (5, 1) — (5, 2) — (5, 3) — (5, 4) — (5, 5) — (5, 6) — (5, 7) — (5, 8)
                          (6, 1) — (6, 2) — (6, 3) — (6, 4) — (6, 5) — (6, 6) — (6, 7)
                              (7, 2) — (7, 3) — (7, 4) — (7, 5) — (7, 6)
                                  (8, 3) — (8, 4) — (8, 5)
                                      (9, 4)
S:\Учёба\9_сем\НИР>
```

Рис. 2.1.3: Проверка работоспособности.

Ниже приведён код программы для генерации 500 случайных бит на трёх кубитах.

```
1 # Example of generating a random sequence using Cirq
2 import cirq
```

```

3 import numpy as np
4
5 def generate_random_sequence_cirq(qubits, repetitions):
6     circuit = cirq.Circuit()
7     for iter in range(repetitions):
8         # Add a single-qubit quantum circuit with random rotations
9         random_angles = np.random.rand(3) * 2 * np.pi # Generate random
          angles from 0 to 2*pi
10        circuit.append([cirq.rx(random_angles[0])(qubits[0]),
11                        cirq.ry(random_angles[1])(qubits[1]),
12                        cirq.rz(random_angles[2])(qubits[2]),
13                        cirq.measure(qubits[0], key=f'result{iter}')]])
14        #print("\rprogress: {}/{}".format(iter + 1, repetitions), end="")
15
16    simulator = cirq.Simulator()
17    result = simulator.run(circuit, repetitions=repetitions)
18
19    # Wait for completion and get results
20    result = result.data
21
22    random_sequence = [result['result{}'.format(_)][0] for _ in range(
        repetitions)]
23    return random_sequence
24
25 # Usage
26 qubits = cirq.LineQubit.range(3) # Example: 3 qubits
27 repetitions = 500
28
29 print("Generating random sequence...")
30 random_sequence_cirq = generate_random_sequence_cirq(qubits, repetitions)
31 print("\nRandom sequence generated.")
32
33 # Save results to file
34 with open('Cirq_result.txt', 'w') as f:
35     for seq in random_sequence_cirq:
36         f.write(f'{seq} ')

```

Листинг 1: Программа с использованием библиотеки Cirq.

Так выглядит запуск программы:

```

S:\учёба\9_сем\НИР>python Cirq.py
Ожидайте результата...

Результат готов.

```

Рис. 2.1.4: Запуск программы.

А вот результат её работы:

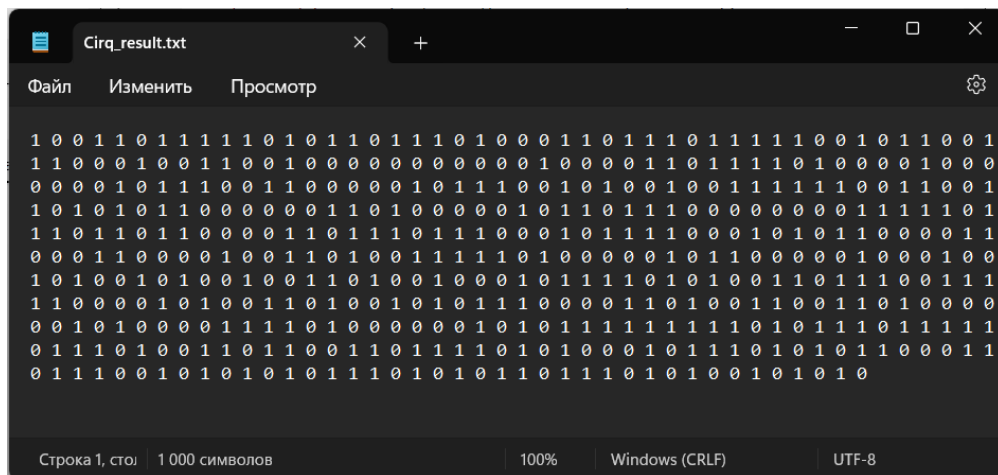


Рис. 2.1.5: Случайные биты, сгенерированные библиотекой Cirq.

## 2.1.2 QISKit

Для начала работы с библиотекой Cirq нам потребуется установить её.

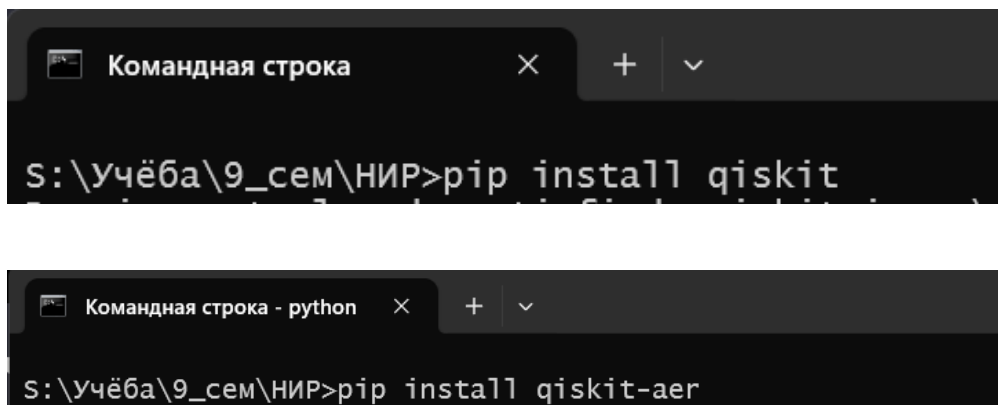


Рис. 2.1.6: Установка библиотеки QISKit.

Ниже приведён код программы для генерации 500 случайных бит на трёх кубитах.

```

1 import numpy as np
2 from qiskit import QuantumCircuit, Aer, transpile, assemble
3
4
5 def generate_random_sequence_qiskit(qubits, repetitions):
6     random_sequence = ''
7
8     # Create a quantum circuit
9     circuit = QuantumCircuit(len(qubits), len(qubits))
10
11     # Add operations and measurements on each iteration
12     for iter in range(repetitions):
13         # Generate random angles from 0 to 2*pi
14         random_angles = np.random.rand(3) * 2 * np.pi
15
16         # Add quantum operations with new angles
17         circuit.rx(random_angles[0], qubits[0])
18         circuit.ry(random_angles[1], qubits[1])

```

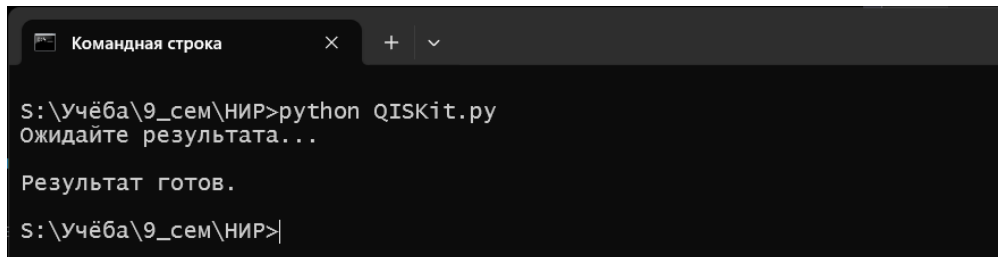
```

19     circuit.rz(random_angles[2], qubits[2])
20
21     circuit.measure(qubits, qubits)
22
23     # Transpile the circuit
24     backend = Aer.get_backend('qasm_simulator')
25     transpiled_circuit = transpile(circuit, backend)
26
27     # Run the simulation
28     result = backend.run(transpiled_circuit, shots=1).result()
29
30     # Get and save the measurement result
31     counts = result.get_counts(circuit)
32     random_sequence += str(int(list(counts.keys())[0][0]) ^ int(list(
counts.keys())[0][1]) ^ int(list(counts.keys())[0][2])) + ' '
33
34     return random_sequence
35
36
37 # Usage
38 qubits = range(3) # Example: 3 qubits
39 repetitions = 500
40
41 print("Generating random sequence...")
42 random_sequence_qiskit = generate_random_sequence_qiskit(qubits, repetitions)
43 print("\nRandom sequence generated.")
44
45 # Save results to file
46 with open('Qiskit_result.txt', 'w') as f:
47     f.write(random_sequence_qiskit)

```

Листинг 2: Программа с использованием библиотеки QISKit.

Так выглядит запуск программы:



```

Командная строка
S:\Учёба\9_сем\НИР>python QISKit.py
Ожидайте результата...

Результат готов.
S:\Учёба\9_сем\НИР>

```

Рис. 2.1.7: Запуск программы.

А вот результат её работы:

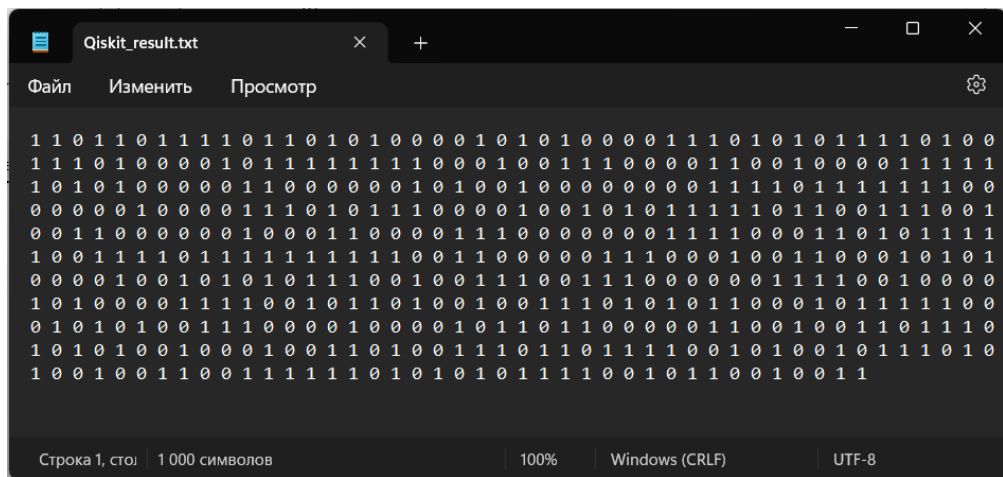


Рис. 2.1.8: Случайные биты, сгенерированные библиотекой QISKit.

### 2.1.3 Forest

Работа с библиотекой Forest будет происходить внутри Microsoft Azure Quantum. Для начала создадим ресурс.

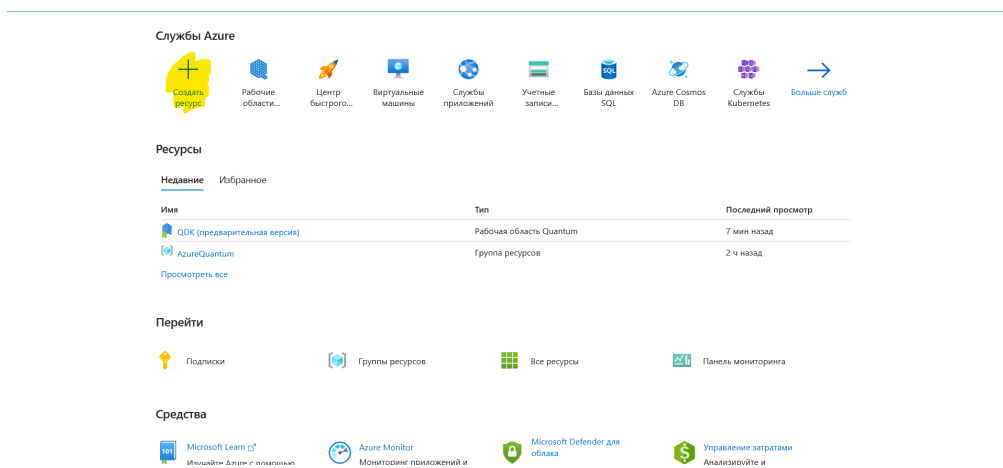


Рис. 2.1.9: Создание нового ресурса в Microsoft Azure Quantum.

Затем создадим новую рабочую область.

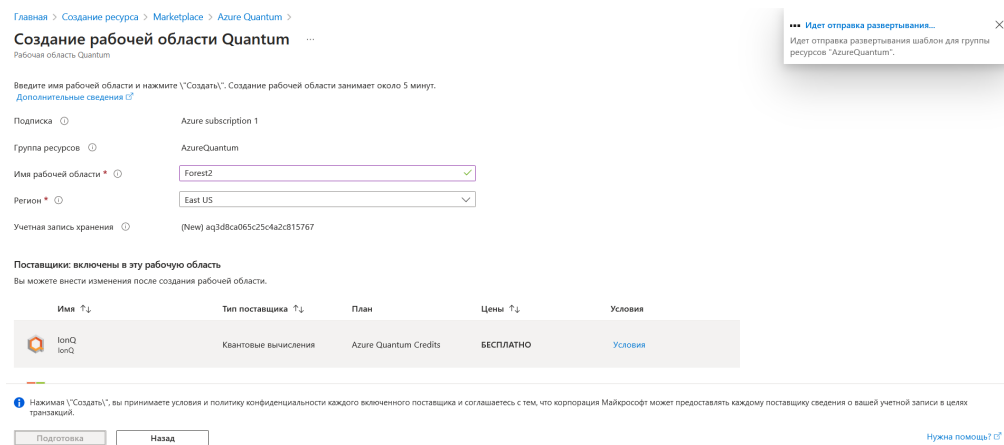


Рис. 2.1.10: Создание новой рабочей области в Microsoft Azure Quantum.

Подождём пока выполнится развёртывание. Это может занять несколько минут.

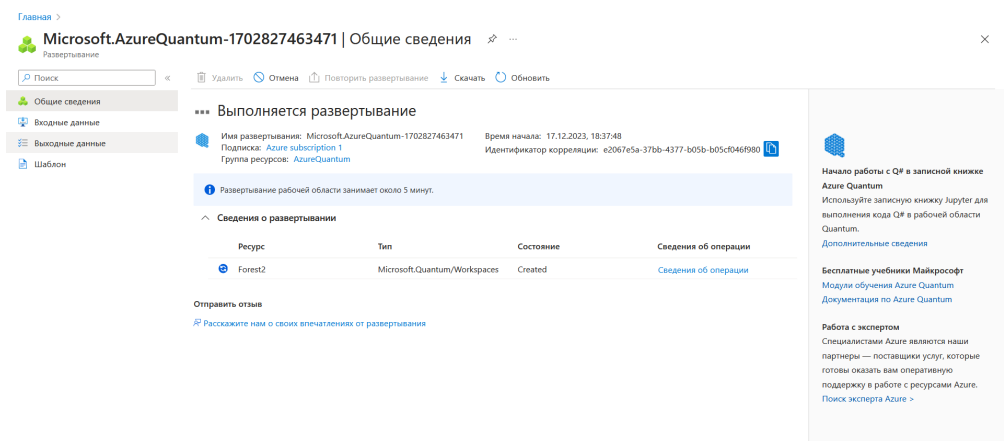


Рис. 2.1.11: Развёртывание новой рабочей области в Microsoft Azure Quantum.

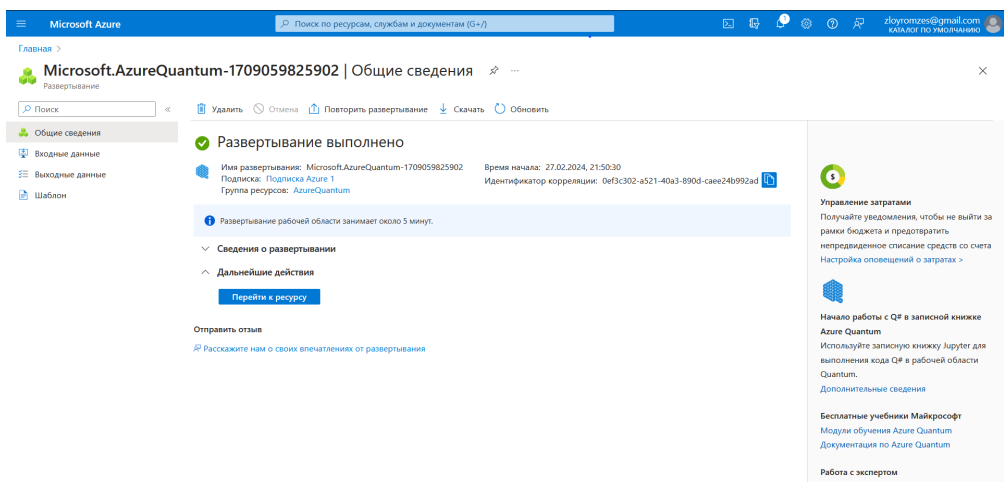


Рис. 2.1.12: Развёртывание новой рабочей области в Microsoft Azure Quantum завершено.

Далее создадим новый Jupiter Notebook и подключим генератор случайности из библиотеки Forest.

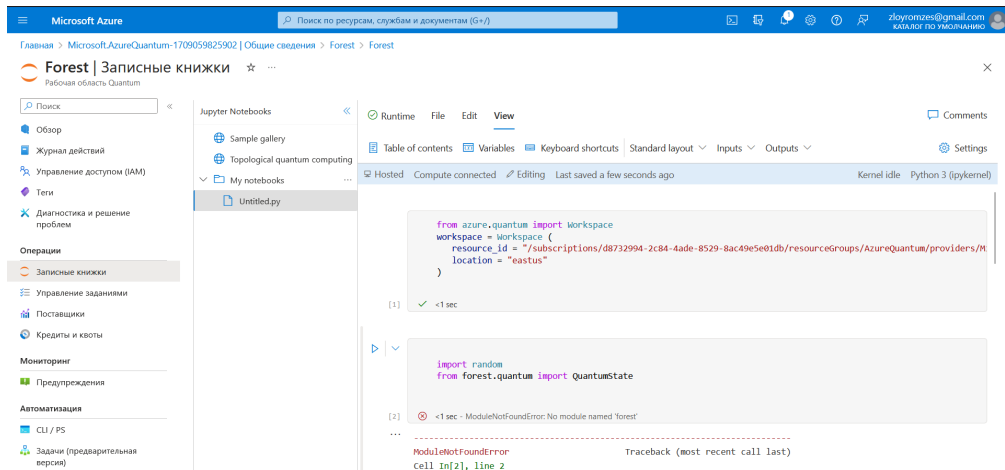


Рис. 2.1.13: Добавление ключа и подключение ресурсов библиотеки.

Также нам не стоит забывать и про непосредственно установку библиотеки Forest внутри нашего окружения.

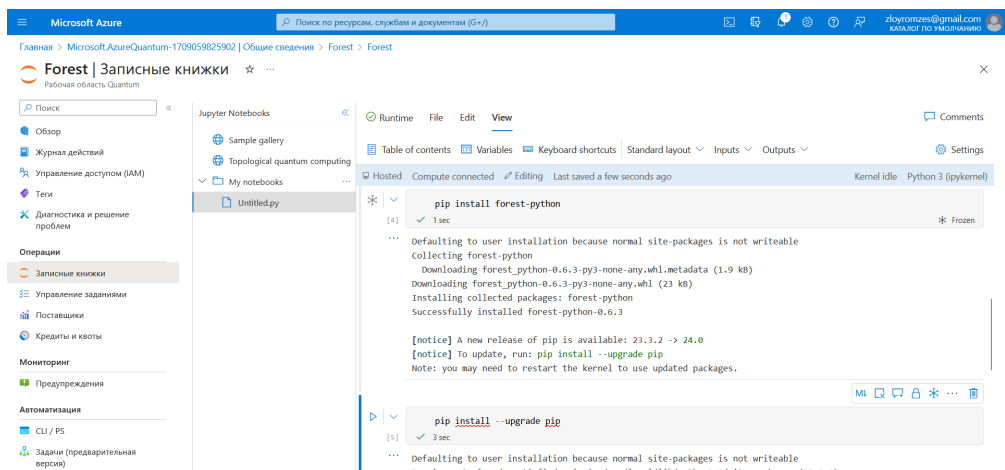


Рис. 2.1.14: Установка библиотеки.

И в конце запускаем программу генерирующую 500 случайных значений на трёх кубитах.

```

from pyquil import Program, get_qc
from pyquil.gates import H
from pyquil.api import local_forest_runtime

# Создаем программу с 3 кубитами
p = Program(H(0), H(1), H(2))

# Получаем квантовую виртуальную машину
qc = get_qc('3q-qvm')

# Запускаем локальный Forest runtime
with local_forest_runtime():
    # Компилируем и выполняем программу
    executable = qc.compile(p)
    result = qc.run(executable)

# Выводим результаты
for i in range(500):
    print(result[i])

```

[3] ✓ <1 sec

```

... ['010', '101', '110', '100', '110', '101', '101', '100', '010', '101', '000', '000', '000', '011', '100', '111',
'010', '100', '101', '101', '010', '101', '010', '001', '001', '101', '001', '001', '101', '100', '010', '100',
'001', '010', '000', '101', '110', '111', '100', '110', '001', '100', '110', '111', '100', '110', '100', '111',
'111', '000', '000', '101', '001', '110', '011', '100', '100', '011', '100', '000', '111', '011', '000', '011',
'110', '001', '110', '010', '010', '111', '110', '111', '011', '010', '001', '100', '010', '100', '000', '101',
'100', '111', '001', '000', '111', '100', '101', '000', '010', '100', '010', '101', '001', '111', '101', '110',
'010', '100', '001', '101', '000', '010', '100', '111', '010', '001', '011', '011', '001', '010', '111', '110',
'010', '010', '111', '110', '010', '100', '000', '110', '100', '001', '110', '110', '101', '101', '011', '110',
'000', '101', '011', '110', '000', '010', '010', '000', '000', '110', '010', '011', '010', '000', '010', '000',
'010', '010', '100', '100', '011', '011', '010', '100', '000', '101', '010', '011', '110', '101', '101',
'000', '011', '100', '110', '010', '011', '001', '101', '010', '001', '011', '001', '100', '001', '111', '110',
'011', '111', '101', '010', '101', '110', '001', '010', '110', '111', '100', '000', '101', '110', '010', '110',
'010', '011', '000', '011', '000', '110', '001', '000', '010', '100', '011', '111', '101', '001', '011', '000',
'000', '110', '011', '000', '000', '010', '111', '010', '001', '010', '000', '100', '100', '101', '000', '101',
'101', '001', '001', '001', '101', '000', '001', '110', '101', '101', '101', '000', '001', '111', '100', '001',
'110', '011', '001', '110', '110', '100', '000', '011', '110', '110', '110', '000', '011', '000', '011', '110',

```

Рис. 2.1.15: Запуск программы.

А вот результат работы программы:

```

[3] ✓ <1 sec
... ['010', '101', '110', '100', '110', '101', '101', '100', '010', '101', '000', '000', '000', '011', '100', '111',
'010', '100', '101', '101', '010', '101', '010', '001', '001', '101', '001', '001', '101', '100', '010', '100',
'001', '010', '000', '101', '110', '111', '100', '110', '001', '100', '110', '111', '100', '110', '100', '111',
'111', '000', '000', '101', '001', '110', '011', '100', '100', '011', '100', '000', '111', '011', '000', '011',
'110', '001', '110', '010', '010', '111', '110', '111', '011', '010', '001', '100', '010', '100', '000', '101',
'100', '111', '001', '000', '111', '100', '101', '000', '010', '100', '010', '101', '001', '111', '101', '110',
'010', '100', '001', '101', '000', '010', '100', '111', '010', '001', '011', '011', '001', '010', '111', '110',
'010', '010', '111', '110', '010', '100', '000', '110', '100', '001', '110', '110', '101', '101', '011', '110',
'000', '101', '011', '110', '000', '010', '010', '000', '000', '110', '010', '011', '010', '000', '010', '000',
'010', '010', '100', '100', '011', '011', '010', '100', '000', '101', '010', '011', '110', '101', '101',
'000', '011', '100', '110', '010', '011', '001', '101', '010', '001', '011', '001', '100', '001', '111', '110',
'011', '111', '101', '010', '101', '110', '001', '010', '110', '111', '100', '000', '101', '110', '010', '110',
'010', '011', '000', '011', '000', '110', '001', '000', '010', '100', '011', '111', '101', '001', '011', '000',
'000', '110', '011', '000', '000', '010', '111', '010', '001', '010', '000', '100', '100', '101', '000', '101',
'101', '001', '001', '001', '101', '000', '001', '110', '101', '101', '101', '000', '001', '111', '100', '001',
'110', '011', '001', '110', '110', '100', '000', '011', '110', '110', '110', '000', '011', '000', '011', '110',

```

Рис. 2.1.16: Получение результата.

## 2.1.4 QDK

Библиотека QDK отличается от прочих, так как она поддерживается языком Q#, а не Python как в предыдущих случаях. Работать мы с ней будем также в Microsoft Azure Quantum. Поэтому первые этапы создание рабочей области будут схожими. Для начала создадим ресурс.



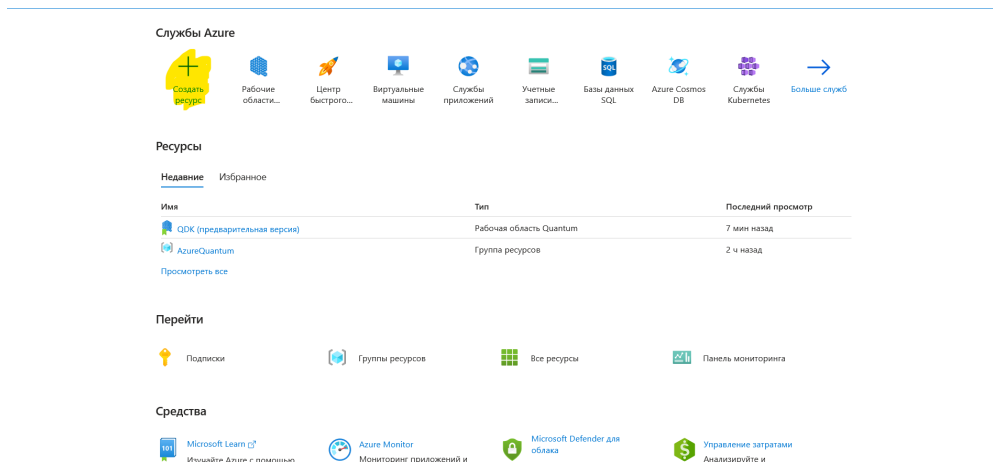


Рис. 2.1.17: Создание нового ресурса в Microsoft Azure Quantum.

Выберем квантовый компьютер Microsoft Azure Quantum.

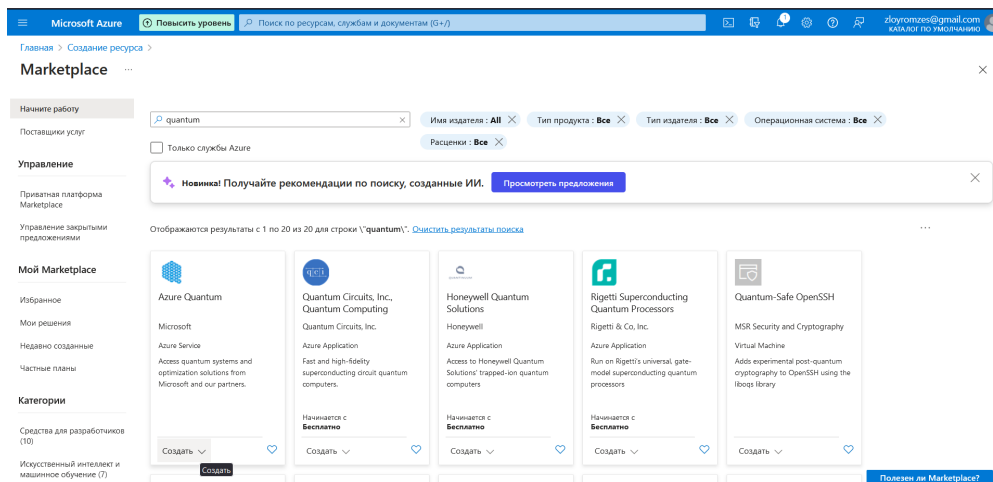


Рис. 2.1.18: Выбор квантового компьютера в Microsoft Azure Quantum.

Затем создадим новую рабочую область.

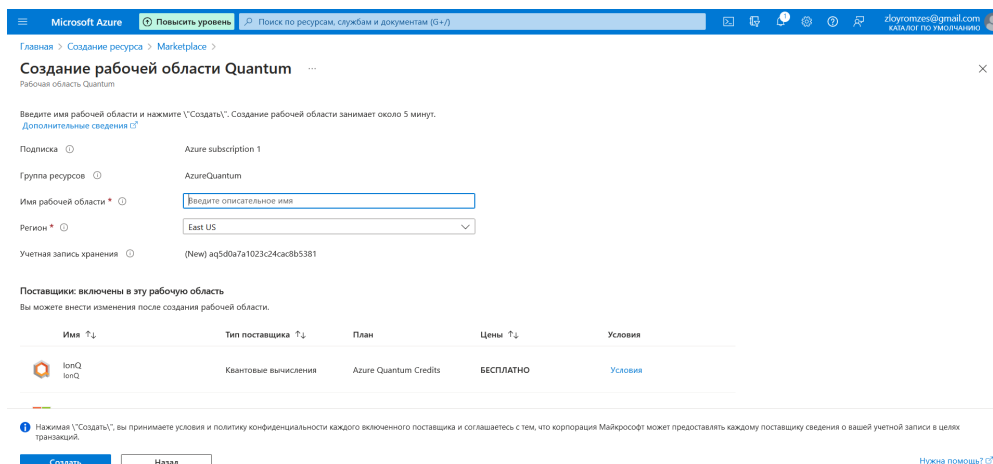


Рис. 2.1.19: Создание новой рабочей области в Microsoft Azure Quantum.

Дождёмся выполнения развёртывания.

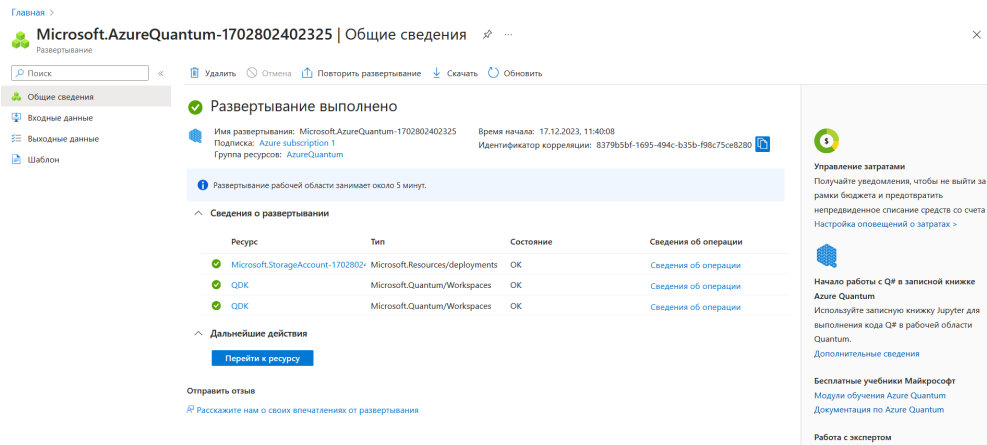


Рис. 2.1.20: Развёртывание новой рабочей области в Microsoft Azure Quantum завершено.

Далее создадим новый Jupiter Notebook и напомним программу на Q# для генерации 500 случайных бит.

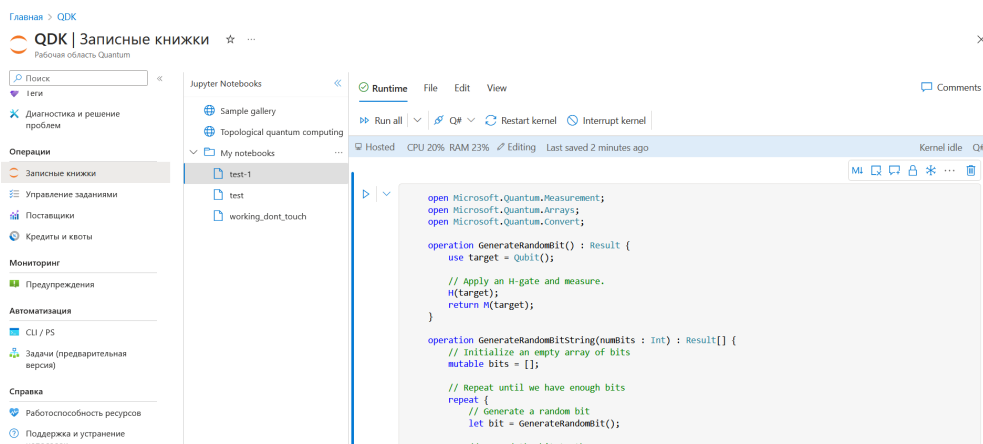
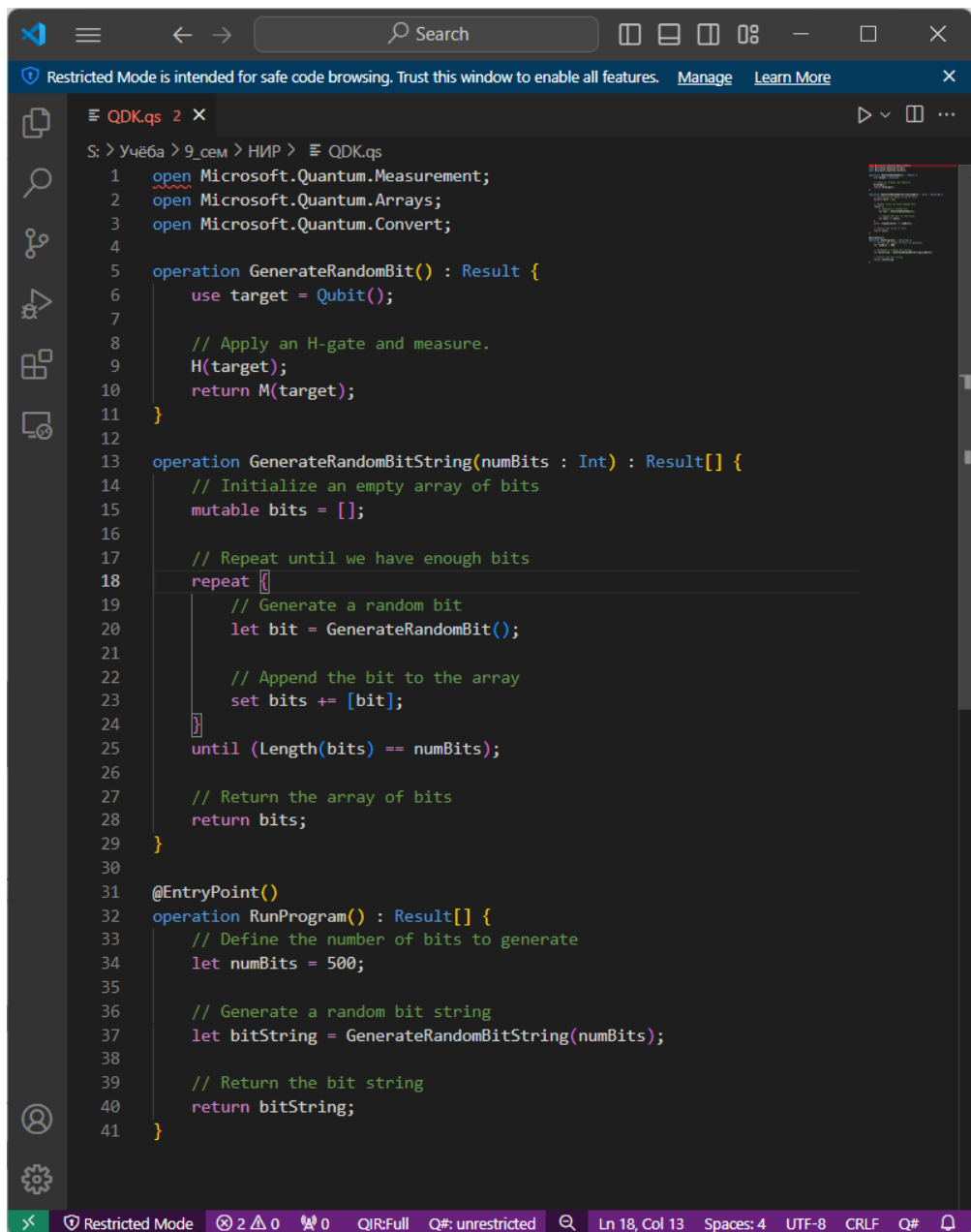


Рис. 2.1.21: Программа на Q#.



The image shows a screenshot of a Q# code editor window. The window title is "QDK.qs 2 X". The code is written in Q# and is contained within a file named "QDK.qs". The code defines two operations: "GenerateRandomBit" and "GenerateRandomBitString". The "GenerateRandomBit" operation takes no arguments and returns a "Result". It uses a "Qubit" target, applies an "H" gate, and then measures the target. The "GenerateRandomBitString" operation takes a "numBits" argument of type "Int" and returns a "Result[]" of type "Result[]". It initializes an empty array of bits, then enters a "repeat" loop that generates a random bit, appends it to the array, and repeats until the array has the desired number of bits. Finally, it returns the array of bits. The code is wrapped in an "@EntryPoint()" block, which defines a "RunProgram" operation that generates a random bit string of 500 bits and returns it. The status bar at the bottom indicates "Restricted Mode", "2 0", "QIR:Full", "Q#: unrestricted", and the current cursor position is "Ln 18, Col 13".

```
1 open Microsoft.Quantum.Measurement;
2 open Microsoft.Quantum.Arrays;
3 open Microsoft.Quantum.Convert;
4
5 operation GenerateRandomBit() : Result {
6     use target = Qubit();
7
8     // Apply an H-gate and measure.
9     H(target);
10    return M(target);
11 }
12
13 operation GenerateRandomBitString(numBits : Int) : Result[] {
14     // Initialize an empty array of bits
15     mutable bits = [];
16
17     // Repeat until we have enough bits
18     repeat {
19         // Generate a random bit
20         let bit = GenerateRandomBit();
21
22         // Append the bit to the array
23         set bits += [bit];
24     }
25     until (Length(bits) == numBits);
26
27     // Return the array of bits
28     return bits;
29 }
30
31 @EntryPoint()
32 operation RunProgram() : Result[] {
33     // Define the number of bits to generate
34     let numBits = 500;
35
36     // Generate a random bit string
37     let bitString = GenerateRandomBitString(numBits);
38
39     // Return the bit string
40     return bitString;
41 }
```

Рис. 2.1.22: Полный код программы на Q#.

После чего подключимся к квантовому компьютеру Microsoft Azure Quantum.

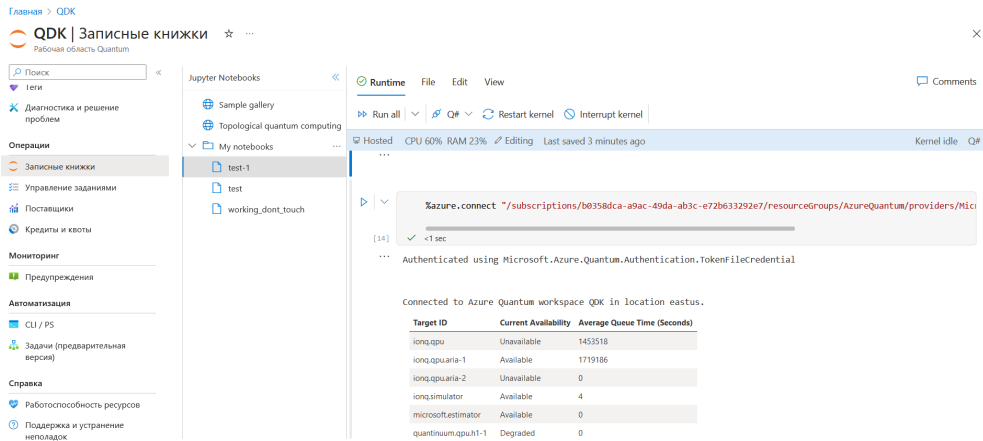


Рис. 2.1.23: Подключение к квантовому компьютеру.

Выберем нужный нам симулятор.

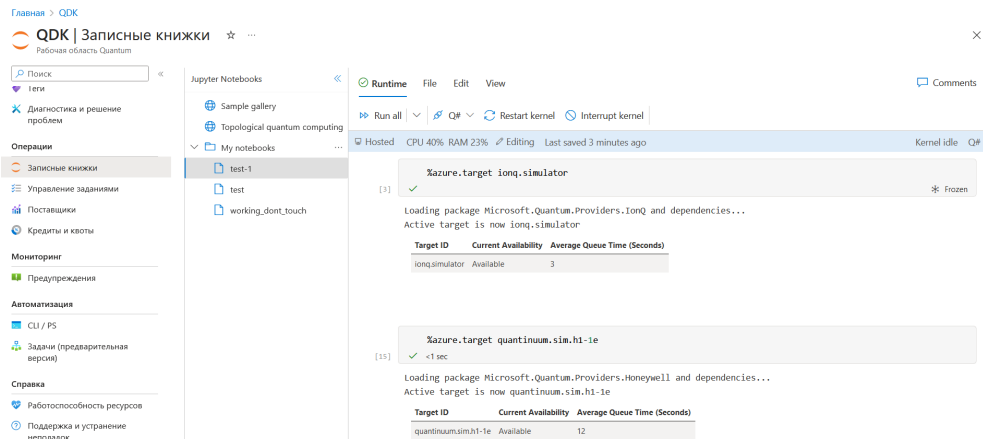


Рис. 2.1.24: Выбор симулятора quantum.sim.h1-1e.

И запустим программу.

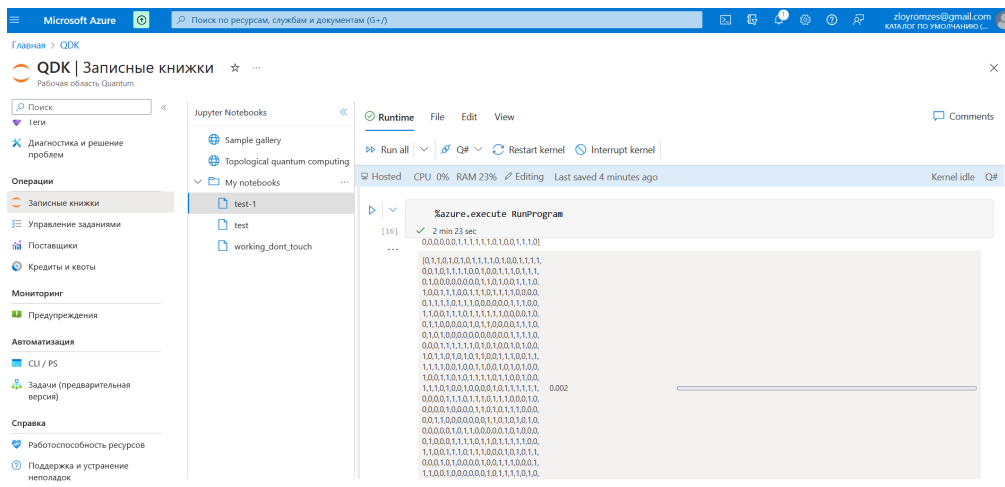


Рис. 2.1.25: Запуск программы.

Вот такую последовательность случайных бит мы получили:

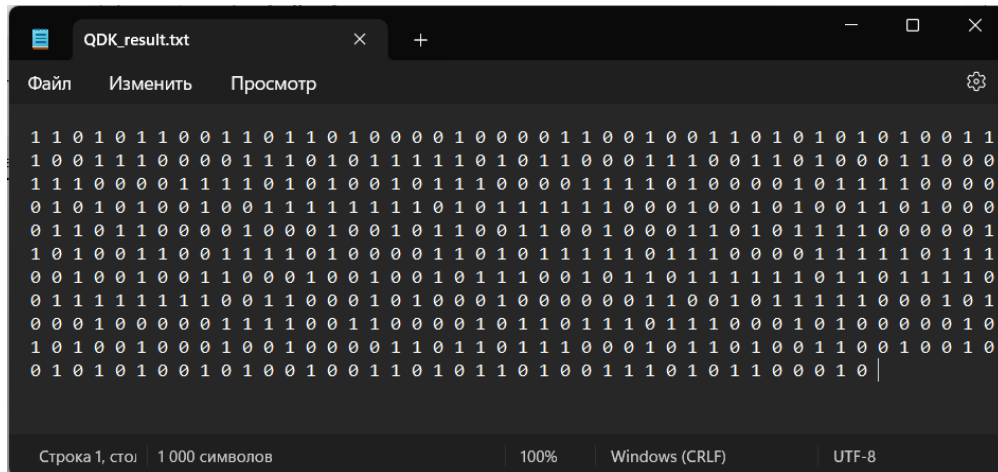


Рис. 2.1.26: Случайные биты, сгенерированные библиотекой QDK.

## 2.2. Статистический анализ полученных результатов

### 2.2.1 Теоретико-математическая основа статистических исследований

#### 1. Среднее значение:

Значение: Среднее значение ( $\mu$ ) – это центр распределения данных, характеризующий его ”типичное” значение.

Формула: 
$$\mu = \frac{1}{n} \sum_{i=1}^n x_i$$

Где:  $n$  – количество элементов в наборе данных;  
 $x_i$  –  $i$ -ое значение в наборе данных.

Интерпретация: При сравнении наборов данных с одинаковыми алфавитами, более высокое среднее значение может указывать на сдвиг распределения вправо. При разных алфавитах интерпретация должна учитывать размерность алфавита.

#### 2. Дисперсия:

Значение: Дисперсия ( $\sigma^2$ ) – это мера того, насколько данные разбросаны вокруг среднего значения.

Формула: 
$$\sigma^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \mu)^2$$

Где:  $\mu$  – среднее значение;  
 $x_i$  –  $i$ -ое значение в наборе данных.

Интерпретация: Большая дисперсия говорит о более широком разбросе данных, а малая – о их концентрации вокруг среднего значения. Сравнение дисперсий корректно только при одинаковых алфавитах.

#### 3. Среднее квадратичное отклонение:

Значение: СКО ( $\sigma$ ) – это квадратный корень дисперсии, характеризующий ”разброс” данных относительно среднего значения.

Формула: 
$$\sigma = \sqrt{\sigma^2}$$

Интерпретация: СКО позволяет интерпретировать разброс данных в контексте их изначальных единиц измерения. Сравнение СКО корректно только при одинаковых алфавитах.

#### 4. Коэффициент асимметрии Пирсона:

Значение: Коэффициент асимметрии Пирсона ( $\gamma_1$ ) показывает, насколько распределение данных асимметрично.

Формула:  $\gamma_1 = \frac{1}{n\sigma^3} \sum_{i=1}^n (x_i - \mu)^3$   
Где:  $\mu$  - среднее значение;  
 $\sigma^3$  - кубический корень СКО.

Интерпретация: Положительное значение  $\gamma_1$  указывает на правостороннюю асимметрию, а отрицательное - на левостороннюю. Значение  $\gamma_1 = 0$  говорит о симметричном распределении.

#### 5. Коэффициент эксцесса Пирсона:

Значение: Коэффициент эксцесса Пирсона ( $\gamma_2$ ) показывает, насколько распределение данных отличается от нормального распределения.

Формула:  $\gamma_2 = \frac{1}{n\sigma^4} \sum_{i=1}^n (x_i - \mu)^4 - 3$   
Где:  $\mu$  - среднее значение;  
 $\sigma^4$  - четвертая степень СКО.

Интерпретация:  $\gamma_2 > 0$  говорит о более «горбатом» распределении, чем нормальное, а  $\gamma_2 < 0$  - о более «плоском».  $\gamma_2 = 0$  соответствует нормальному распределению.

#### 6. Энтропия Шеннона:

Значение: Энтропия Шеннона (H) – это мера неопределенности в данных, характеризующая их «информативность».

Формула:  $H = - \sum_{i=1}^n p_i \log_2(p_i)$   
Где:  $p_i$  – вероятность i-го символа

Интерпретация: Чем выше энтропия, тем больше неопределенность в данных. Сравнение энтропии корректно только при одинаковых алфавитах.

#### 7. Тест Колмогорова-Смирнова:

Значение: Тест Колмогорова-Смирнова – это тест на то, насколько распределение данных соответствует гипотетическому распределению.

Формула:  $D_n = \max_x |F_n(x) - F(x)|$   
 $D_n$  - статистика Колмогорова-Смирнова  
Где:  $F_n(x)$  - эмпирическая функция распределения  
 $F(x)$  - гипотетическая функция распределения

Интерпретация: Тестовая статистика Колмогорова-Смирнова (D) представляет собой максимальную разность между кумулятивной функцией распределения (CDF) данных и кумулятивной функцией распределения (ECDF) гипотетического распределения. Так мы можем выразить соответствие гипотетическому распределению.

#### 8. Тест на серийную корреляцию:

Значение: Тест на серийную корреляцию позволяет проверить, наличие зависимости между соседними значениями в ряду данных.

Формула: 
$$\sum_{i=1}^n \frac{(x_i - \mu)(x_{i-1} - \mu)}{\sigma^2(n-1)}$$

Где:  
 $\mu$  - среднее значение;  
 $\sigma^2$  - дисперсия;  
 $x_i$  - i-ое значение в наборе данных;  
 $n$  - количество элементов в наборе данных.

Интерпретация: р-значение теста показывает, насколько вероятно получить такое же или более экстремальное значение статистики теста при отсутствии серийной корреляции. Малое р-значение (обычно  $< 0.05$ ) говорит о наличии серийной корреляции.

## 2.2.2 Интерпретация результатов

### 1. Среднее значение:

- Cirq: 0.494
- Qiskit: 0.492
- Forest: 3.48
- QDK: 0.496

Среднее значение – это среднее арифметическое всех значений в наборе данных.

Cirq, Qiskit, QDK: значения близки к 0.5, что говорит о симметричном распределении данных.

Forest: значение (3.48) близко к пороговому (3.5), что говорит о возможном сдвиге распределения.

### 2. Дисперсия:

- Cirq: 0.249964
- Qiskit: 0.249936
- Forest: 5.1256
- QDK: 0.249984

Дисперсия показывает, насколько значения в наборе данных отклоняются от среднего значения.

Cirq, Qiskit, QDK: значения близки к 0.25, что говорит о том, что данные сгруппированы вокруг среднего значения.

Forest: значение (5.1256) значительно выше порогового (8.75), что говорит о большом разбросе данных.

### 3. Среднее квадратичное отклонение:

- Cirq: 0.499964
- Qiskit: 0.499936
- Forest: 2.26398
- QDK: 0.499984

Среднее квадратичное отклонение (СКО) – это квадратный корень из дисперсии.

Cirq, Qiskit, QDK: значения близки к 0.5, что согласуется с интерпретацией среднего значения.

Forest: значение (2.26398) выше порогового (2.96), что подтверждает большой разброс данных.

#### 4. Коэффициент асимметрии Пирсона:

- Cirq: 0.0240017
- Qiskit: 0.0320041
- Forest: -0.0209799
- QDK: 0.0160005

Коэффициент асимметрии Пирсона показывает, насколько распределение данных асимметрично.

Во всех случаях значения близки к 0, что говорит о симметричности распределений.

#### 5. Коэффициент эксцесса Пирсона:

- Cirq: -1.99942
- Qiskit: -1.99898
- Forest: -1.22666
- QDK: -1.99974

Коэффициент эксцесса Пирсона показывает, насколько распределение данных отличается от нормального распределения.

Во всех случаях значения близки к -1, что говорит о "плоскости" распределений по сравнению с нормальным.

#### 6. Энтропия Шеннона:

- Cirq: 4.4291
- Qiskit: 4.41117
- Forest: 23.5902
- QDK: 4.44703

Энтропия Шеннона показывает, насколько неопределенны данные.

Cirq, Qiskit, QDK: значения близки к 4.4, что говорит о умеренном уровне неопределённости.

Forest: Энтропия 23.5902 говорит о том, что данные значительно более неопределенны, чем в других наборах.

#### 7. Тест Колмогорова-Смирнова:

- Cirq: 0.998
- Qiskit: 0.998
- Forest: 0.986



- QDK: 0.998

Все значения теста Колмогорова-Смирнова близки к 1, что говорит о том, что данные во всех 4 наборах не отличаются от равномерного распределения.

8. Тест на серийную корреляцию:

- Cirq: -0.030209
- Qiskit: 0.0377647
- Forest: -0.0118664
- QDK: -0.0301263

Все значения теста на серийную корреляцию близки к 0, что говорит о том, что в данных нет серийной корреляции.

Cirq, Qiskit и QDK имеют схожие статистические характеристики, что говорит о схожей природе данных.

Данные в Forest имеют значительно бóльшую вариативность (дисперсия, СКО, энтропия) по сравнению с Cirq, Qiskit и QDK.

### 3. ЗАКЛЮЧЕНИЕ

В данной работе был проведен сравнительный анализ четырех популярных библиотек разработки квантовых схем: Cirq, QISKit, Forest и QDK. Цель работы заключалась в оценке их возможностей и эффективности на примере реализации задачи генерации истинно случайной последовательности.

Для каждой библиотеки были разработаны квантовые схемы генерации случайных чисел, проанализированы их особенности и преимущества.

Результаты исследования показали, что:

- Cirq обладает лаконичным синтаксисом, удобным для опытных пользователей, но может быть сложным для начинающих. Библиотека предоставляет инструменты для оптимизации и компиляции квантовых схем, а также возможность симуляции их работы на различных вычислительных платформах.
- QISKit отличается дружелюбным интерфейсом и обширной документацией, что делает ее привлекательной для новичков. Она предлагает богатый набор инструментов для работы с квантовыми алгоритмами, включая визуализацию схем и симуляцию.
- Forest ориентирована на машинное обучение и квантовые вычисления. Она предоставляет мощные функции для работы с тензорными сетями и оптимизации квантовых алгоритмов.
- QDK разработана Microsoft и тесно интегрируется с Azure Quantum. Библиотека фокусируется на высокопроизводительных квантовых вычислениях и оптимизации для квантовых компьютеров Microsoft.

В целом, данное исследование продемонстрировало, что все четыре библиотеки обладают своими преимуществами и могут быть эффективно использованы для разработки квантовых алгоритмов.

Выбор наиболее подходящей библиотеки зависит от конкретных задач, опыта пользователя и доступных вычислительных ресурсов.

## 4. СПИСОК ЛИТЕРАТУРЫ

1. Лекция №1 по курсу "Квантовая физика"(Глазков В.Н.) - <https://m.youtube.com/watch?v=pgB5APB3JKo>
2. Лекция 1. А.С. Трушечкин, Д.А. Кронберг. Математические основы квантовой криптографии - <https://www.youtube.com/watch?v=7B49Ec5ORpQ>
3. Дмитриев И. С. Я не особо думал об этом (М. Планк и квантовая революция) // ДИС-КУРС. 2019 - [https://discourse.etu.ru/assets/files/02\\_dmitriev.pdf](https://discourse.etu.ru/assets/files/02_dmitriev.pdf)
4. Баскаков П. Е., Хабовец Ю. Ю., Пилипенко И. А., Кравченко В. О., Черкесова Л. В. Инструменты для выполнения и эмуляции квантовых вычислений // Вестник НГУ. Серия: Информационные технологии. 2020 - <https://cyberleninka.ru/article/n/instrumenty-dlya-vypolneniya-i-emulyatsii-kvantovyh-vychisleniy>
5. Ковалев Д.А, Квантовые компьютеры. Что это и зачем они нужны?, 2021 - <https://cyberleninka.ru/article/n/kvantovye-kompyutery-cto-eto-takoe-i-zachem-oni-nuzhny>
6. Колесников П.О., Голубничий А.А. К вопросу о квантовых компьютерах, их развитии и современном состоянии, 2022 - <https://cyberleninka.ru/article/n/k-voprosu-o-kvantovyh-kompyuterah-ih-razvitii-i-sovremennom-sostoyanii>
7. Минобрнауки России, ОмГТУ; сост.: Гончар И.И., М. В. Чушняка М.В., Основы квантовой механики: метод. указания к практ. занятиям и самостоят. работе студентов, Изд-во ОмГТУ, 2017 - [https://omgtu.ru/general\\_information/faculties/radio\\_engineering\\_department/departament\\_of\\_quot\\_physics\\_quot/lib\\_p fys/280402-280302/Gonchar\\_MU\\_OsnKvMex.pdf](https://omgtu.ru/general_information/faculties/radio_engineering_department/departament_of_quot_physics_quot/lib_p fys/280402-280302/Gonchar_MU_OsnKvMex.pdf)
8. Гольдин Л.Л., Новикова Г.И., Квантовая физика. Вводный курс, 2002 - <http://booksshare.net/books/physics/goldin-ll/2002/files/kvantovayafizikavvodniykurs2002.pdf>
9. Джеммер М., Эволюция понятий квантовой механики, 1985 - <https://t-library.net/read/7394>
10. Thomas Wangler, Development of a Vendor Independent Quantum Computing Transpiler, 2020 - [https://elib.uni-stuttgart.de/bitstream/11682/11236/1/master\\_thesis.pdf](https://elib.uni-stuttgart.de/bitstream/11682/11236/1/master_thesis.pdf)
11. Ryan LaRose, Overview and Comparison of Gate Level Quantum Software Platforms, 2018 - <https://quantumcomputingreport.com/wp-content/uploads/2018/06/Overview-and-Comparison-of-Gate-Level-Quantum-Software-Platforms-Final-June-21-2018.pdf>
12. Hidary, J. D., Quantum Computing: An Applied Approach, Alphabet X, 2019 - <https://dokumen.pub/quantum-computing-an-applied-approach-9783030239213-9783030239220.html>
13. S. Aaronson. The Limits of Quantum Computers. Scientific American, March 2008 - [https://www.cs.virginia.edu/~robins/The\\_Limits\\_of\\_Quantum\\_Computers.pdf](https://www.cs.virginia.edu/~robins/The_Limits_of_Quantum_Computers.pdf)
14. Marie Salm, Johanna Barzen, Frank Leymann, Benjamin Weder, Karoline Wild, Automating the Comparison of Quantum Compilers for Quantum Circuits, 2021 - [https://www.iaas.uni-stuttgart.de/publications/Salm2021\\_CompilerComparison.pdf](https://www.iaas.uni-stuttgart.de/publications/Salm2021_CompilerComparison.pdf)