PROGRAMMING IN QUANTUM COMPUTERS

BLERTA LEKA (MOÇKA)[1], DANIEL LEKA[2]

[1]Department of Mathematics and Informatics, Faculty of Economics and Agribusiness, Agricultural University of Tirana, Albania

[2]Special Court of First Instance for Corruption and Organized Crime, 1016, Tirana, Albania

e-mail: blerta.mocka@fshn.edu.al

Abstract

Quantum computers have emerged as a transformative technology, challenging the traditional binary language of classical computers. Since their inception in the 1980s, quantum computers have revolutionized data processing by utilizing quantum bits (qubits) capable of existing in superpositions, representing both 0 and 1. This unique property enables quantum computers to excel in a wide range of computational tasks, especially those demanding substantial processing power. Consequently, corporations, governments, and academic institutions are heavily investing in the development of quantum hardware, software, and applications. This article provides a survey of quantum computers, explains the programming languages involved in the field, and explores the integration of Python into quantum computers. Even through the challenges in years of quantum computers, we will identify the beginning but also the challenges in that and theirs.

Key words: *Quantum computers, qubits, Python, Qiskit.*

Përmbledhje

Kompjuterët kuantikë kanë dalë si një teknologji transformuese, sfiduese për gjuhën binare tradicionale të kompjuterëve klasikë. Që nga themelimi i tyre në vitet 1980, kompjuterët kuantikë kanë revolucionarizuar procesimin e të dhënave duke përdorur bitët kuantikë (qubit) të cilët mund të ekzistojnë në supozime duke u shfaqur si 0 dhe 1. Kjo cilësi e veçantë u mundëson kompjuterëve kuantikë të jenë mjaft të përshtatshëm në një gamë të gjerë të detyrave kompjuterike, sidomos atyre që kërkojnë fuqi të madhe të procesimit. Si rezultat, koorporatat, qeveritë dhe institucionet akademike po investojnë shumë në zhvillimin e hardware-ve e, software-ve dhe

aplikacioneve kuantike. Ky artikull ofron një studim të kompjuterëve kuantikë, shpjegon gjuhët e programimit të përdorura në këtë fushë dhe hulumton integrimin e Python në kompjuterët kuantikë. Gjithashtu me anë të zhvillimit në vite të kompjuterav kuantike, do të identifikojmë përparësitë por edhe sfidat në përdorimin dhe zhvillimin e tyre.

***Fjalët kyçe**: Kompjuterat kuantikë, qubit-et, Python, Qiskit.*

Introduction

The concept of a quantum computer was first proposed in the early 1980s by the physicist Richard Feynman, who suggested that quantum computers could simulate complex quantum systems that are too difficult to simulate by classical computers (Richard, 1982). However, the first experimental realization of a quantum computer was not achieved until the late 1990s. In 1998, a team led by Isaac Chuang and Neil Gershenfeld at the Massachusetts Institute of Technology (MIT) demonstrated a two-qubit quantum computer using nuclear magnetic resonance (NMR) techniques (Gershenfeld & Chuang, (1998). This system was able to perform simple quantum algorithms, such as the Deutsch-Jozsa algorithm, which is designed to determine whether a function is constant or balanced. There have been many advances in quantum computing, with researchers developing a variety of technologies for creating and manipulating qubits, including superconducting circuits, trapped ions, and topological qubits. Classical computers, reliant on binary bits, face exponential costs in simulating complex physical systems. Quantum computers, manipulating qubits based on quantum mechanical components, offer a paradigm shift. Richard Feynman's early 1980s proposal paved the way for circumventing the computational limitations of classical computers (Demmer et al.).

While current quantum computers are still limited in size and their performance, they have already demonstrated the potential to solve some problems much faster than classical computers and to simulate quantum systems that are too complex to be handled by classical computers (Rijmenam, 2022). Quantum computers are being discussed more and more at the scientific level and are being implemented in many companies. Figure 1 shows a timeline of their development.

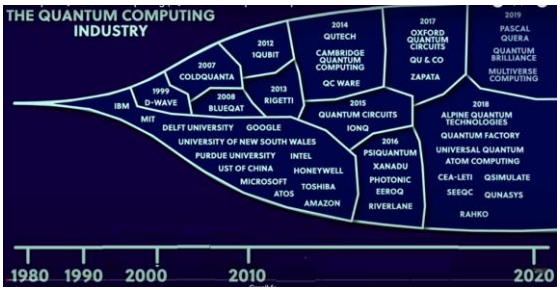

Figure 1 The Quantum Computer Industry (Domain of Science, 2021)

Quantum computers have the potential to solve many specific problems faster than conventional or classical computers (Lee, 2021). They can also be used for optimization problems, such as finding the shortest path between multiple points or optimizing resource allocation, and for simulating quantum systems, such as chemical reactions or the behavior of materials at the atomic scale. Through the speed of calculation on a very large amount of data, quantum computers will change the operation in telecommunications, finance, medicine, cyber security and many other fields. These computers have the potential to replace classic computers in almost every field and will have a successful future (Sharma, 2020).

How quantum computers work

Quantum computers work differently from classical computers. In classical computers, everything was based on binary language, with 0 and 1. Quantum computers are not limited to these two states (Figure 2). They can have a much larger number of states at the same time (Bourzac, 2019).

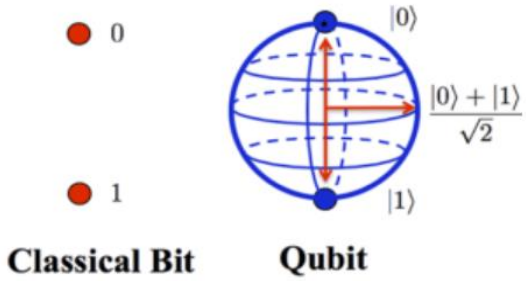

Figure 2 Comparison of classical Bit with Qubit (Quantum states)

In order to understand this large number of states, we must first understand three main characteristics: superposition, interference and entanglement

(Qiskit Development Team, 2022). In classical computers a state is called a bit. In quantum computers it is called quantum bits or qubits. These conditions work in very different ways. The classic bit, if we will call it that, functions as a switch which can have the state 0 for off and 1 for on. These two conditions are independent of each other and well defined. But Qubit doesn't work the same way, it's more complex.

We can think of it as an arrow which can point in a 3-dimensional space. If the mark above is the value 1 and if the mark below is the value 0. But, in addition to these, we also have the superstates, which are combinations of 0 and 1. Whether it will take the value 1 or 0 will be decided by the highest percentage in which the arrow is located. (Figure 3)

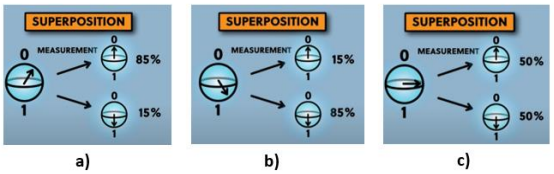

Figure 3 a) value 0; b) value 1; c) superposition can take the value 0 or 1 (Hajjar, 2022).

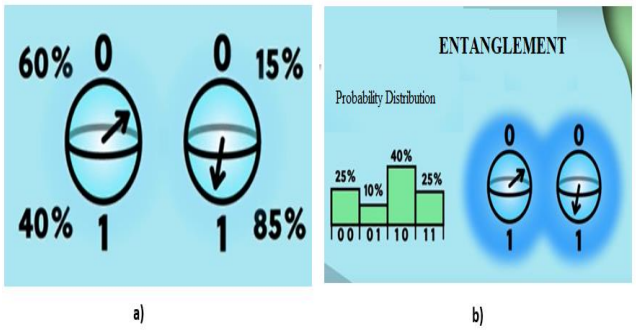

Figura 1 Entanglement

Qubits are not independent of each other; they are all part of a larger state. This is true regardless of the number of qubits we have. Based on the number of the qubit we calculate the number of possible states which is 2 to the power of the number of the qubit (Table 1)

Number of qubit	**Number of states**
1	2
2	4
…	…
n	2^n

Table 1 Number of states depending on the number of qubits (StackExchange, 2018).

Interference

The quantum wave function can be represented as a sphere (Qiskit Development Team, 2022). Several qubits are several wave functions added together to give a multifunction. Although there can be multiple states that some qubits can be in, when we measure it, we only have one state at a given time. In computers we must use constructive interference to get the correct answer and use destructive interference to reduce the number of wrong answers. (Figure 5)

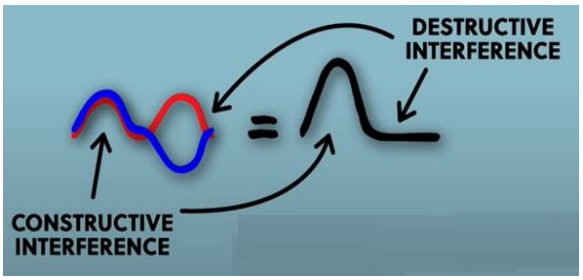

Figure 5 Constructive and destructive interference

Entanglement

In classical computers, state 0 is independent from state 1. While in quantum computers, these two states are dependent on other things or intertwined together. When we analyze the state of a single qubit, it is determined by the

percentage. If we look at the state of two qubits, they will have several combinations of states (Figure 4 a). If the arrow on one of the qubits changes it will affect all the other states. This is the entanglement property, which is observed when we have more than one qubit. This mode of operation of quantum computers allows them to process massive and complex data sets more efficiently than classical computers. Current quantum computers usually isolate qubits from their environment as best they can, but as the number of qubits multiplies, this isolation becomes extremely difficult to maintain (Ball, 2021).

Quantum Computing Challenges

In the dynamic landscape of quantum computing, characterized by the emergence of quantum processors exemplified by Google's Sycamore, the trajectory of computing power is undergoing a paradigm shift, ushering in the era of quantum supremacy (Tepanyan, 2023). This transformative phase, rooted in three decades of interdisciplinary breakthroughs, holds the promise of substantial disruption in the technological landscape. However, the realization of quantum computing's full potential is intricately entangled with a myriad of limitations and challenges.

As highlighted by Clark et al., stakeholders in quantum computing must grapple with cryptographic vulnerabilities, navigate technological uncertainties, and effectively harness quantum advancements for societal benefit. The multifaceted challenges in quantum computing demand comprehensive attention from various quarters. Quantum decoherence emerges as a critical hurdle, with quantum systems proving highly sensitive to environmental disturbances, leading to qubit entanglement issues (Banafa, 2023; Priya, 2023; UGI and Swayne, 2023). The inherent fragility of quantum systems, necessitating costly refrigeration, and the limited coherence time of qubits further complicate their practical implementation.

Geopolitical competition introduces concerns about potential divisions in quantum networks, posing challenges to global governance (Brooks, 2021). Security risks, particularly in quantum decryption, underscore the imperative of ethical considerations and the development of post-quantum cryptography. Policymakers face the intricate task of navigating institutional transitions into the quantum age, with initiatives like the National Quantum Initiative Act aimed at addressing these challenges. Collaboration between private sector leaders is deemed essential for progress. On a technical front, challenges span scalable and stable quantum hardware, noise management,

the development of efficient quantum algorithms, error correction, quantum communication advancement, and seamless integration with classical technology (Swayne and Priya, 2023). The shortage of skilled professionals, lack of standardization, and high expenses present additional hurdles. Despite the formidable nature of these challenges, ongoing efforts, increased funding, and collaborative initiatives provide hope for overcoming these obstacles and unlocking the transformative potential inherent in quantum computing.

Programming in quantum computers

Quantum computers are able to reduce the machine learning process from several hundreds of years to seconds (Zyga, 2015). Major technologies such as IBM and Google are investing in and producing quantum computers (Lee, 2021). Quantum programming languages are instructions designed to run on quantum computers. The most suitable quantum languages and which require the order to be learned are (Hajjar, 2022; Muthyala, 2021):

- **QLC (Quantum Computing Language)** is a redevelopment of the C language for quantum computers. It is the first Quantic language implemented.
- **QMASM (Quantum Macro Assembler)** or quantum low-level language was released in 2016. This language and from the name itself have features of Assemble language where programmers need to know hardware specifications in detail and write code is performed by writing instructions in a low-level abstraction.
- **SILQ** is a high-level language released in 2020. It is written in the D language. The D language has 482 stars, 10 contributions on github and is a language that updates regularly.

Quantum Computing with Python

Python is becoming an increasingly popular language for quantum computing research and development. There are several reasons for this:

- Easy to learn and use: Python is known for its simplicity and ease of use, making it a good choice for researchers and developers who are new to quantum computing.
- Large community and ecosystem: Python have a large community of users and developers, and a rich ecosystem of libraries and tools,

making it easy to find resources and get help with quantum computing projects.

- Versatility: Python is a versatile language that can be used for a wide range of tasks, from data analysis to machine learning to web development, making it well suited for quantum computing research and development.
- Integration with quantum software and hardware: Many quantum software and hardware providers offer Python APIs and software development kits (SDKs), allowing developers to easily integrate quantum functionality into their Python applications.

There are several Python development tools designed specifically for quantum computing, such as **Qiskit**, **PyQuil** and **Cirq**. These tools provide a range of functionality, from simulating quantum circuits to interfacing with real quantum hardware. The characteristics of the most commonly used Python tools for quantum coding are shown in Table 2.

Table 2 Comparison of Python tools (Web: Qiskit, Cirq, PyQuil; Metwalli, 2022)

Qiskit	Cirq	PyQuil
It was developed by IBM and provides a complete set of tools and libraries for quantum computing.	Cirq is developed by Google and focuses on providing a clean and simple API for building quantum circuits.	PyQuil was developed by Rigetti Computing and is specifically designed for programming and simulating quantum computers.
It supports a wide range of quantum operations and gates, making it suitable for complex quantum circuit construction.	It offers a wide range of quantum gates and operations, making it suitable for various quantum algorithms and simulations.	It provides a high-level interface for quantum programming, making it accessible to users without prior experience in quantum computing.
Qiskit allows users to run their circuits on real quantum hardware provided by IBM Quantum.	Cirq provides simulators for circuit simulation, including the ability to simulate measurement results.	PyQuil supports a variety of quantum gates and operations, enabling users to build complex quantum circuits.

It provides a rich set of simulators, including the state vector simulator, which allows users to simulate quantum state evolution and derive measurement results.	It supports a variety of backends, including Google's Quantum hardware, allowing users to run their circuits on real hardware.	It provides simulators, such as Wavefunction Simulator, that allow users to simulate quantum circuits and obtain wavefunction or measurement results.
Qiskit has a large and active community, with extensive documentation and tutorials available.	Cirq has a lower approach compared to Qiskit, which makes it suitable for users who prefer finer control over their quantum circuits.	PyQuil can connect to Rigetti's Quantum Virtual Machine (QVM) or the Forest platform to run programs on their quantum hardware.

Installing and using Qiskit, Cirq and PyQuil

All three SDK are installed using the pip (Python package manager) command when we have Python pre-installed on our computer. Related commands:

pip install qiskit

pip install cirq

pip install PyQuil

Qiskit provides various modules to work with different aspects of quantum computing, including quantum circuits, simulators, and supports for running circuits on quantum hardware. Cirq provides a variety of classes and methods for working with quantum circuits.

Creating quantum circuits:

Code in Qiskit:

from qiskit import QuantumCircuit, execute, Aer

 # Creating a quantum circuit with two qubits

circuit = QuantumCircuit(2)

circuit.h(0)

""" Application of the CX gate that checks the first qubit and the second qubit with circuit.cx(0, 1)."""

```
circuit.cx(0, 1)
# Circuit simulation using the Aer simulator
simulator = Aer.get_backend('qasm_simulator')
job = execute(circuit, simulator)
result = job.result()
# Numbering of measurement results
counts = result.get_counts(circuit)
print(counts)
```

Code in Cirq:

```
import cirq
# Creating a quantum circuit
circuit = cirq.Circuit()
qubits = cirq.LineQubit.range(2)
circuit.append(cirq.H(qubits[0]))
circuit.append(cirq.CNOT(qubits[0], qubits[1]))
# Circuit simulation
simulator = cirq.Simulator()
result = simulator.simulate(circuit)
# Numbering of measurement results
measurements = result.measurements
print(measurements)
```

Code in PyQuil:

```
from pyquil import Program
from pyquil.api import QVMConnection
# Creating a quantum circuit
```

```
program = Program()
program += program.h(0)
program += program.cnot(0, 1)
# Simulating the program using QVM (Quantum #Virtual Machine)
qvm = QVMConnection()
result = qvm.run_and_measure(program, [0, 1], trials=10)
# Numbering of measurement results
print(result)
```

All three languages have similar procedures. The main difference lies in the specific classes and methods they provide for working with quantum circuits and simulating quantum algorithms. Qiskit is developed by IBM, Cirq by Google and PyQuil by Rigetti, so the choice is based on preference or the quantum hardware the user will be using. Qiskit is the most popular and widely used quantum programming library.

Using the Qiskit library in the quantum teleportation algorithm

Quantum teleportation is a foundational concept in the field of quantum information, enabling the transfer of an unknown quantum state from one qubit (referred to as the sender qubit, or user qubit A) to another qubit (designated as the receiver qubit, or user qubit B) through the utilization of quantum entanglement and classical communication (Rieffel & Polak, 2011). This phenomenon carries significant implications for the domains of quantum communication and computational tasks.

In contrast to classical data transfer methods, quantum teleportation offers several intriguing advantages. Traditional methods rely on transmitting information through classical channels, which can be subject to various limitations such as bandwidth constraints, signal degradation, and eavesdropping risks. In contrast, quantum teleportation leverages the unique properties of quantum entanglement to transfer quantum states without exposing the actual state itself during transmission. The key distinction lies in the principles of superposition and entanglement, which allow quantum teleportation to transmit quantum information in a way that is not achievable by classical means. Quantum teleportation provides a secure and efficient means of transferring quantum states across large distances, making it a

promising candidate for quantum communication and cryptography applications.

The code presented below was developed and executed using the IBM Quantum Platform. The chosen account type is a free subscription, which facilitates the simulation of quantum teleportation through the utilization of the Qiskit library.

Code Implementation (Qiskit, Quantum Teleportation):The code implementation (Qiskit, Quantum Teleportation):

```
from qiskit import QuantumCircuit, execute, Aer
from qiskit.visualization import plot_histogram
# Creating a qubit circuit with three qubits
circuit = QuantumCircuit(3, 3)
# A prepares the conditions to be teleported
circuit.h(0)
circuit.cx(0, 1)
circuit.barrier()
# Entanglement of qubit 2 with the state to be teleported
circuit.cx(1, 2)
circuit.h(0)
circuit.barrier()
circuit.measure([0, 1], [0, 1])
circuit.barrier()
circuit.cx(1, 2)
circuit.cz(0, 2)
# Measuring the final teleported qubit
circuit.measure(2, 2)
# Simulate the circuit using the Air simulator
simulator = Aer.get_backend('qasm_simulator')
job = execute(circuit, simulator, shots=1000)
```

result = job.result()

 # We take and draw the measurement results

counts = result.get_counts(circuit)

plot_histogram(counts)

The results of the code are depicted in Figure 6, c, which displays histograms representing the measurement results of the teleported qubit. These histograms illustrate the distributions of measurement outcomes for varying numbers of simulations, specifically, for 10, 100, 1000, and 10,000 runs. Notably, as the number of simulations increases, the observed trend reveals a convergence towards near-equal separations among the measurement outcomes, highlighting the statistical robustness and consistency of the quantum teleportation process as the computational workload scales.

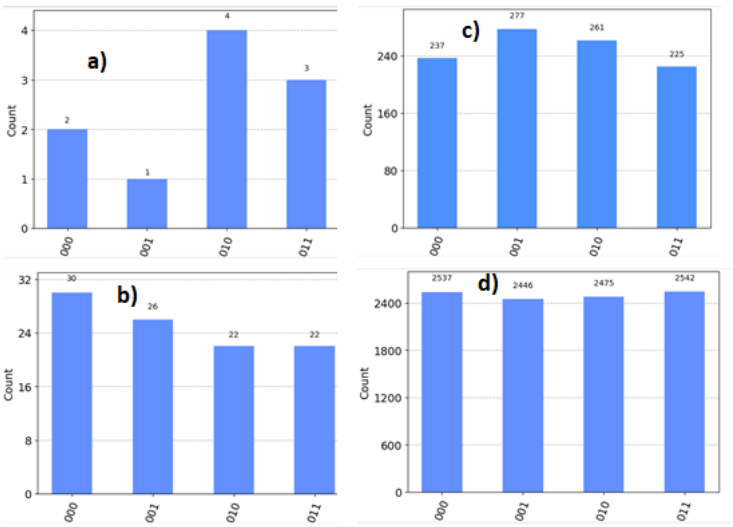

Figure 6 Histograms according to the number of simulations: a for 10, b for 100, c for 1000 and d for 1000 (Souce: Authors simulations)

Moreover, our observation that the measurement outcomes tend to approach equal separations as the number of simulations increases underscores the statistical predictability and stability of quantum teleportation. In contrast,

classical data transfer methods may exhibit variability and susceptibility to external factors that can affect the reliability of transmitted information.

Python, in general, is a versatile and accessible language that is suitable for quantum computing research and development, and its popularity is likely to continue to grow in the future (Skecic & Yakaryılmaz, 2022). By studying and comparing several quantum computing SDKs, we identified that each quantum SDK has its own documentation and provides users with cross-platform support with an easy-to-use syntax. However, programming for quantum computers is still unknown and difficult. Some of the difficulties we may encounter in programming are (Muthyala, 2021; Dilmegani, 2022):

- The difficulty in formulating a universal QC (Quantum Computing) language;
- Incomplete and hidden variables in quantum machines;
- Quantum computers themselves are still in their infancy and are not powerful enough to run complex quantum algorithms.

Conclusions

Quantum computing programming is in its early phases but is advancing rapidly. Leading global IT giants such as Google, IBM, Microsoft, Intel, Amazon, and many others are actively harnessing the potential of quantum computers to advance fields like materials science, medicine, and artificial intelligence applications. As quantum technology progresses, the intricate integration of quantum systems with existing classical computing platforms becomes increasingly challenging. The interest and expertise among programmers, students, and programming enthusiasts in quantum computing are continually expanding.

Quantum programming environments like Qiskit, Cirq, and PyQuil provide developers with tools to navigate the complexities of programming quantum computers and explore real-world applications. These technologies are in a constant state of advancement. In a recent scientific study, Qiskit demonstrated its utility as an open-source tool for investigating quantum teleportation, offering pre-built functions and tools for the construction, simulation, and execution of quantum circuits, simplifying the implementation of quantum protocols such as teleportation. Due to its rapid development, quantum computer programming is poised to progress at an accelerated rate compared to classical computing. This accelerated advancement promises to unlock new innovations and vast opportunities

across a wide array of fields. As time unfolds, programming for quantum computers will continue to evolve and establish itself as a potent tool for conquering intricate computational challenges and fostering groundbreaking applications.

Challenges in the trajectory of quantum computing, such as quantum decoherence, the fragility of quantum systems, geopolitical competition, and security risks, necessitate comprehensive attention from policymakers, researchers, and industry stakeholders, but ongoing efforts and collaborative initiatives offer hope for overcoming these hurdles and realizing the transformative potential of quantum computing.

References

Ball, P. (2021): Major Quantum Computing Strategy Suffers Serious Setback. https://www.quantamagazine.org/major-quantum-computing-strategy-suffers-serious-setbacks-20210929/

Banafa, A., (2023) Trends and Challenges in Quantum Computing, https://www.bbvaopenmind.com/en/technology/digital-world/quantum-computing-trends/

Bourzac, K. (2019): To upgrade quantum computers, researchers look to materials science. https://cen.acs.org/materials/electronic-materials/upgrade-quantum-computers-researchers-look/97/i15

Brooks, H., (2021) Quantum Computers: Opportunities, Risks, and Challenges for Policymakers, American University of Washington DC, https://www.american.edu/sis/centers/security-technology/quantum-computers.cfm

Clark, R., Bartlett, S., Bremner, M., Lam, P. K., & Ralph, T. (2021). QUANTUM COMPUTING. In The impact of quantum technologies on secure communications (pp. 13–16). Australian Strategic Policy Institute. http://www.jstor.org/stable/resrep31261.6

Demmer, M., Fonseca, R. and Koushanfar, F. Richard Feynman: simulating physics with computers, cs294: reading the classics,

https://www.fisica.net/computacaoquantica/richard_feynman_simulating_physics_with_computers.pdf

Dilmegani, C., (2022), Quantum Programming: Languages, SDKs & Algorithms in 2023, https://research.aimultiple.com/quantum-computing-programming

Domain of Science, (2021): The Map of Quantum Computing | Quantum Computers Explained. https://www.youtube.com/watch?v=-UlxHPIEVqA&t=388s

Feynman, Richard (1982): Simulating Physics with Computers. International Journal of Theoretical Physics. 21 (6/7): 467–488.

https://web.archive.org/web/20190108115138/https://people.eecs.berkeley.edu/~christos/classics/Feynman.pdf

Gershenfeld, N & Chuang, I., (1998), Quantum Computing with Molecules Scientific American Vol. 278, , pp. 66-71

Hajjar, A. J. (2022): Quantum Programming in 2022: Languages, SDKs & Algorithms. https://research.aimultiple.com/quantum-computing-programming/

Lee, Ch. (2021): Programmable optical quantum computer arrives late, steals the show. https://arstechnica.com/science/2021/03/programmable-optical-quantum-computer-arrives-late-steals-the-show/

Lee ,M. (2021): Quantum Computing and Cybersecurity.

https://www.belfercenter.org/publication/quantum-computing-and-cybersecurity

METWALLI, S., (2022): Python For Quantum Computers.

https://www.shecancode.io/blog/python-for-quantum-computers

Muthyala, Sh. (2021): Top Quantum Programming Languages to learn in 2022 ". https://www.analyticsinsight.net/top-quantum-programming-languages-to-learn-in-2022-to-know/

Priya, D., (2023) 10 Challenges In Quantum Computing,

https://www.analyticsinsight.net/10-challenges-in-quantum-computing

Qiskit Development Team, (2022): Quantum computing in a nutshell.

https://qiskit.org/documentation/qc_intro.html

Qiskit: Quantum Teleportation, https://learn.qiskit.org/course/ch-algorithms/quantum-teleportation

Rieffel, E & Polak, W (2011): QUANTUM COMPUTING, A Gentle Introduction, The MIT Press, Cambridge, Massachusetts, London, England

Rijmenam, M. van, (2022): HOW QUANTUM COMPUTING WILL CHANGE THE WORLD. https://www.thedigitalspeaker.com/quantum-computing-change-world/

Sharma, P. (2020): A Study on Quantum Computing, Journal of Xi'an University of Architecture & Technology, Volume XII, Issue IV, 2020ISSN No : 1006-7930

Skecic, M. and Yakaryılmaz, A (2022): Comparing Quantum Software Development Kits for Introductory Level Education. Baltic J. Modern Computing, Vol. 10 (2022), No. 1, pp. 87–104 https://doi.org/10.22364/bjmc.2022.10.1.06

StackExchange (2018): What are the models of quantum computation?.

https://quantumcomputing.stackexchange.com/questions/74/what-are-the-models-of-quantum-computation

Swayne, M., (2023) Quantum Computing Business,

https://thequantuminsider.com/2023/03/24/quantum-computing-challenges/

Tepanyan, H. (2023). Google Quantum Computer, https://www.bluequbit.io/google-quantum-computer

UGI (Universal Group of Institutions), The challenges of Quantum Computing, https://universalinstitutions.com/the-challenges-of-quantum/

Zyga, L.(2015) Quantum computers could greatly accelerate machine learning by, Phys.org,https://phys.org/news/2015-03-quantum-greatly-machine.html

Web: Qiskit:https://qiskit.org/documentation/

Cirq: https://quantumai.google/cirq/start/basics

PyQuil: https://pyquil-docs.rigetti.com/en/v2.7.0/migration.html