UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Marcus Vinícius Torres Silva

Qiskit: Um estudo comparativo da programação quântica e clássica

Uberlândia, Brasil

2025

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Marcus Vinícius Torres Silva

Qiskit: Um estudo comparativo da programação quântica e clássica

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, como parte dos requisitos exigidos para a obtenção título de Bacharel em Ciência da Computação.

Orientador: Prof. Ma. Giullia Rodrigues de Menezes

Coorientador: Prof. Dr. João Henrique de Souza Pereira

Universidade Federal de Uberlândia – UFU

Faculdade de Computação

Bacharelado em Ciência da Computação

Uberlândia, Brasil

2025

Marcus Vinícius Torres Silva

Qiskit: Um estudo comparativo da programação quântica e clássica

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, como parte dos requisitos exigidos para a obtenção título de Bacharel em Ciência da Computação.

Trabalho aprovado. Uberlândia, Brasil, 09 de maio de 2025:

Prof. Ma. Giullia Rodrigues de Menezes
Orientador

Prof. Dr. João Henrique de Souza Pereira
Coorientador

Prof. Dra. Maria Adriana Vidigal de Lima
Convidado 1

Prof. Dr. Rodrigo Sanches Miani
Convidado 2

Uberlândia, Brasil
2025

Resumo

Este trabalho tem o objetivo de explorar, de forma introdutória, a programação no universo quântico, pois, por mais que a maioria das previsões de tendências para o futuro coloque a computação quântica como uma das principais na lista, muitas pessoas ainda não conhecem seus fundamentos e seu potencial. Assim, este trabalho busca desenvolver um estudo comparatório inicial entre computação quântica e clássica, unindo teoria e prática por meio do kit de desenvolvimento de software Qiskit (do inglês *software development kit*, SDK). Inicialmente, são feitas definições básicas e essenciais para entender o funcionamento de um programa quântico, além de benefícios, surgimento e evolução dessa área. Além disso, é feita uma comparação conceitual entre a computação clássica e a computação quântica, introduzindo o conceito de qubit, que pode ser entendido como um conceito análogo ao de um bit no paradigma convencional. No desenvolvimento deste trabalho, é realizada uma comparação entre a computação quântica e a computação clássica, na qual é utilizado o kit de desenvolvimento qiskit da IBM e a linguagem de programação clássica C para exemplificar essa comparação em um nível básico de lógica de programação, apresentando exemplos de programas com estruturas condicionais e de repetição nos dois paradigmas. Além de comparar a estrutura dos programas, é comparado também um exemplo prático sobre o problema de fatoração. Foram aplicados um algoritmo clássico de fatoração em C e o algoritmo de Shor em qiskit, em que foi utilizado um simulador de um computador quântico e uma entrada fixa (N = 15), a fim de obter o tempo de execução dos programas. O resultado foi mais favorável ao algoritmo em C, pois o algoritmo quântico de Shor executa mais operações e pode levar mais tempo em entradas menores do que um algoritmo clássico. Todo o passo a passo para utilizar o qiskit é descrito na seção de desenvolvimento, desde a instalação dos pacotes até a execução dos programas. Os programas utilizados neste trabalho podem ser encontrados no seguinte repositório: https://github.com/marcustorres02/qiskit-c.

Palavras-chave: Programação, Qiskit, Quântica.

Lista de ilustrações

Lista de abreviaturas e siglas

CQ	Computação Quântica
CX	Porta CNOT
DH	Diffie-Hellman
ECC	Elliptic Curve Cryptography
H	Porta Hadamard
IA	Inteligência Artificial
IBM	International Business Machines Corporation
RSA	Rivest-Shamir-Adleman
SDK	Software Development Kit
UFU	Universidade Federal de Uberlândia
VS Code	Visual Studio Code
X	Porta NOT

Sumário

1 Introdução

A Computação Quântica está surgindo como uma das áreas mais promissoras para o futuro, como foi a Inteligência Artificial há alguns anos atrás e ainda se estabelece até hoje. De acordo com o artigo "Intelligent computing: the latest advances, challenges, and future" Zhu et al. (2023), os computadores clássicos estão atingindo o limite do seu poder computacional e cada vez se espera mais velocidade no processamento de dados. Para isso, a computação quântica pode ser uma vantagem para resolver problemas complexos, como problemas que envolvem fatoração, por exemplo.

Algumas empresas líderes de tecnologia estão desenvolvendo soluções utilizando computação quântica há alguns anos, seja criando computadores quânticos, seja desenvolvendo linguagens e ferramentas capazes de desenvolver programas quânticos. Entre essas empresas de referência, estão a IBM, Google, Microsoft e a Intel Advisors (2024). A IBM, por exemplo, vem desenvolvendo processadores quânticos mais robustos ano após ano, a fim de diminuir os erros por interferência externa, fenômeno comum na quântica e que é estudado pelos cientistas para entender certos comportamentos das partículas.

Por mais que a Computação Quântica seja objeto de estudo há alguns anos e algumas empresas apostem nessa nova tecnologia, muitas pessoas, principalmente no Brasil, não possuem conhecimento algum sobre esse tipo de computação. E isso inclui as pessoas que estão em instituições acadêmicas que oferecem cursos de computação, pois a maioria destes cursos no Brasil, por muitas vezes, não explora o assunto.

Assim, este trabalho tem como objetivo apresentar conceitos introdutórios e aplicações da Computação Quântica, além de desenvolver uma comparação entre a Computação Quântica e a Computação Clássica. A comparação será feita utilizando o kit de desenvolvimento Qiskit, da empresa IBM, e a linguagem de programação clássica C. Os tópicos abordados serão introdutórios, para começar a introduzir uma pessoa sem conhecimento algum sobre quântica nesse novo universo.

1.1 Objetivos Gerais

Desenvolver um estudo comparatório inicial entre a Computação Quântica e a Computação Clássica, com foco nas definições iniciais sobre a quântica e suas aplicações. Além disso, busca introduzir uma pessoa sem conhecimento algum em Computação Quântica a fim de aprender os conceitos básicos e iniciar o estudo em uma das ferramentas quânticas mais utilizadas no mundo atualmente, a ferramenta Qiskit, da IBM.

1.2 Objetivos Específicos

- Comparar, de forma teórica e prática, a Computação Quântica e a Computação Clássica.
- Apresentar os benefícios da Computação Quântica em determinadas áreas.
- Conceituar os princípios quânticos.
- Guia de instalação do kit de desenvolvimento Qiskit.
- Como executar um programa quântico em Qiskit.
- Introduzir a forma de como escrever um programa em Qiskit.
- Como realizar Estruturas Condicionais e de Repetição em Qiskit.
- Realizar comparação com a linguagem clássica C.

1.3 Organização da Monografia

Este trabalho está organizado em seis capítulos, com alguns destes contendo subseções. O Capítulo 1 é dedicado à introdução, em que é feita uma contextualização sobre o tema. No Capítulo 2, são apresentados os principais conceitos e a linha do tempo sobre Computação Quântica. No Capítulo 3, apresentam-se os principais trabalhos relacionados na literatura, contendo um resumo sobre estes. No Capítulo 4, é abordada a metodologia utilizada para o desenvolvimento do trabalho. O Capítulo 5 é referente ao desenvolvimento e aos resultados obtidos. O último capítulo, Capítulo 6, é responsável por realizar as considerações finais, apresentando as conclusões do trabalho e propostas de trabalhos futuros.

2 Fundamentação Teórica

2.1 Comparação entre Computação Quântica e Computação Convencional

A Computação Quântica pode ser vista como uma área diferente da computação atual, que podemos chamar de Computação Convencional. Na Quântica, tem-se um grande entrelaçamento de três grandes ciências: Ciência da Computação, Física e Matemática, e utiliza as propriedades da mecânica quântica para criar suas próprias regras. De acordo com Damasco (2019), a "computação quântica é uma nova forma de computação, baseada em fenômenos da mecânica quântica". Além disso, é comentado que "a computação quântica não é só uma computação mais rápida ou eficiente, ela é uma nova maneira de fazer computação que requer um modelo diferente para representar os problemas e solucioná-los usando fenômenos da mecânica quântica".

Em relação à Computação Convencional, é essa a computação que é utilizada atualmente e que surgiu há algumas décadas atrás. Nela, é utilizado o conceito de bit, que é definido como a unidade básica de informação. O bit pode ser representado por dois valores (0 ou 1) e podemos definir o "0" como algo "falso/desligado", e o "1" como "verdadeiro/ligado".

Enquanto na Computação Convencional existe o conceito de bit, na Computação Quântica tem-se o qubit. McMahon (2008) apresenta que de maneira análoga ao conceito de bit, podemos representar uma unidade básica de informação na Computação Quântica, que é chamada de qubit. Conforme apresentado na Figura 1, o qubit pode existir no valor ou estado "0" ou "1", mas também pode existir como uma combinação dos dois estados (0 e 1), esse fenômeno é chamado de superposição. Entretanto, quando ocorre a medição desse qubit em algum momento, ele só poderá ser encontrado no estado "0" ou no estado "1".

Carvalho, Lavor e Motta (2007) mostra algumas representações para o estado de um qubit na literatura, e uma das notações mais famosas é a notação de Dirac. Nessa forma, um estado quântico é representado por vetores, ao invés de apenas 0 e 1, como na computação convencional

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

Outra diferença importante entre os dois paradigmas é que a Computação Clássica é caracterizada por ser determinística. Dada a mesma entrada e um algoritmo, na maioria dos casos será produzida a mesma saída, a não ser que haja manipulação para simular uma aleatoriedade. Mesmo assim, a natureza da Computação Convencional segue sendo determinística. Por sua vez, a Computação Quântica tende, por sua natureza, a ser um paradigma probabilístico. Ou seja, uma mesma entrada pode gerar uma saída diferente a cada execução, e o que se mede é a probabilidade de tal saída ocorrer.

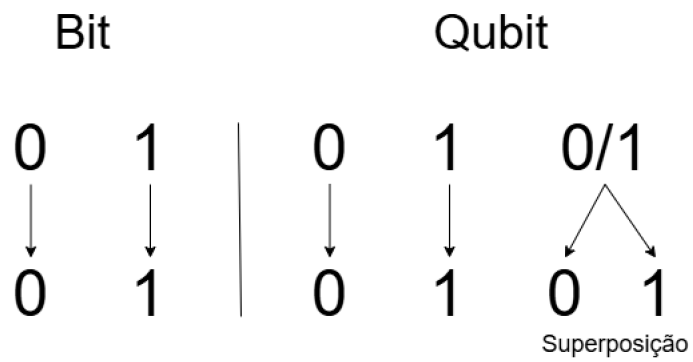

Figura 1 – Estados de um bit e de um qubit (Adaptado de Graph (2023)).

2.2 Avanços da Computação Quântica

Explicada a diferença entre a Computação Quântica e a Computação Convencional, é necessário entender por que esse novo tipo de computação vem sendo amplamente debatido e considerado um paradigma que irá coexistir com a computação clássica. Para isso, as próximas subseções serão dedicadas às aplicações da quântica no cenário atual.

2.2.1 Medicina

No campo da medicina, já foram iniciadas pesquisas com a computação quântica e os resultados têm sido motivadores, mostrando vantagens em relação aos melhores computadores clássicos atuais na atuação em determinadas áreas. Rasool et al. (2023) comenta que o uso da computação quântica poderá acelerar o diagnóstico e tratamento de doenças, descobrir novos medicamentos por meio de simulações moleculares e realizar avanços no tratamento do câncer por radioterapia.

O estudo de Flöther (2023) apresenta que a utilização do modelo computacional quântico irá acelerar exponencialmente os avanços da medicina e saúde. O autor divide o uso da CQ (Computação Quântica) em três áreas principais de casos de uso. A primeira área é a genômica, responsável por entender o funcionamento do genoma humano, realizando simulações em nível de moléculas. O segundo tópico comentado é sobre o diagnóstico de doenças, a fim de obter resultados mais assertivos e com rapidez. E o último

tema é comentado sobre os tratamentos e o aumento da eficiência da radioterapia por meio da computação quântica.

O autor Shahwar et al. (2022) traz uma pesquisa sobre a combinação entre a computação clássica e quântica, em que redes neurais clássicas atuam em processadores quânticos para detectar automaticamente pacientes com a doença de Alzheimer. Além de acelerar o diagnóstico, foi mostrado que, com o uso da quântica, houve um aumento na precisão de classificação de 97,2%, em comparação aos 92% do modelo tradicional sem utilizar a computação quântica.

2.2.2 Criptografia

Em relação à atuação da quântica na criptografia, Mattielo et al. (2012) comenta sobre a estimativa do tempo em que um número leva para ser fatorado. Podemos citar o algoritmo de criptografia assimétrica *Rivest-Shamir-Adlemar* (RSA), que apenas é considerado seguro atualmente devido ao problema de fatorar grandes números. Com a Computação Quântica, estima-se que esses números poderão ser fatorados utilizando o Algoritmo de Shor, e então, o RSA não seria considerado mais seguro.

Ainda sobre criptografia, Scholten et al. (2024) fala em uma seção, dos algoritmos de criptografia de chave pública, *Rivest-Shamir-Adlemar* (RSA), *Diffie-Hellman* (DH) e *Elliptic Curve Cryptography* (ECC). O autor comenta que, em um futuro, com o uso de algoritmos quânticos, esses algoritmos de criptografia atuais poderão ser "quebrados" facilmente e, então, se estabeleceria uma nova era da criptografia.

2.2.3 Outras áreas

IBM (2024) além de trazer as definições iniciais de Computação Quântica e os fenômenos físicos que a envolvem, a empresa IBM, uma das empresas pioneiras em pesquisas em computação quântica e construção de computadores quânticos, traz em uma seção algumas áreas onde a computação quântica está sendo utilizada atualmente, as quais podemos citar a área de veículos elétricos com a Mercedes-Benz, de energias renováveis com a ExxonMobile e aplicações no avanço de pesquisas nucleares com a organização europeia CERN.

Damasco (2019) traz, além da criptografia, aplicações da quântica nas áreas de comunicação, citando o recorde estabelecido em 2018 de maior distância na transmissão quântica via satélite que ocorreu entre a Áustria e a China, com 7.600 km. Além disso, comenta-se sobre o foco da quântica para aperfeiçoar cada vez mais o uso da Inteligência Artificial, mais especificamente em aprendizado de máquina.

A Computação Quântica também está começando a atuar em conjunto com a Inteligência Artificial, é o que diz no artigo Zhu et al. (2023). O trabalho cita que está

sendo utilizada o que ele chama de "IA Quântica", em pesquisas como síntese de drogas e no tratamento de algumas reações químicas, mas o uso da combinação da IA e Quântica ainda se encontra em estágio inicial.

2.3 Nascimento e Evolução da Computação Quântica

2.3.1 Século XX

A computação quântica só foi proposta devido aos estudos em uma área da Física, chamada de mecânica quântica. Tal ideia como hoje é entendida, foi desenvolvida com estudos de vários físicos, como o físico austríaco Erwin Schrödinger (1887-1961), famoso pelo experimento mental que ficou conhecido como o "Gato de Schrödinger", e o físico alemão Werner Heisenberg (1901-1976) Piza (2003). As ideias da mecânica quântica foram derivadas dos estudos do também físico alemão Max Planck (1858-1947) Falcão e Molinari (2016), o qual definiu que a energia é emitida e absorvida em "pacotes", que chamou de *quanta*. A palavra tem derivação do latim *quantum*, que tem o significado de "quantidade" ou "quanto".

A ideia de um dispositivo que utilizasse princípios da mecânica quântica começou a ser pensada já na década de 1970, devido a "Lei de Moore" e o problema de em algum momento ocorrer uma limitação dos circuitos em chips de silício, como descrito em Capoferri et al. (2021).

Capoferri et al. (2021) também comenta que foi a partir da década de 1980 que o assunto começou a ganhar forma e diversos pesquisadores começaram a publicar os primeiros trabalhos. Os físicos estadunidenses Paul Benioff (1930 - 2022) e Richard Feynman (1918 - 1988), e o físico israelense David Deutsch, foram os pioneiros a trabalhar a ideia de um computador quântico.

Em relação aos primeiros algoritmos quânticos descritos, foi na década de 1990 que ocorreram os primeiros passos. Como destacado em Insider (2020), em 1994 o matemático estadunidense Peter Shor apresentou um algoritmo que hoje é conhecido como "Algoritmo de Shor". Esse algoritmo permite que um computador quântico consiga fatorar grandes números, chegando a superar os melhores cálculos feitos em um computador clássico, e assim, começou-se a olhar com mais cautela para a segurança da criptografia. Um outro algoritmo notório, é o algoritmo de busca quântica de Grover, proposto pelo cientista indiano Lov Grover em 1996. Esse algoritmo utiliza paralelismo quântico para encontrar as soluções em um problema de busca.

2.3.2 Século XXI

A partir do início do século XXI, outros fatos importantes aconteceram na computação quântica até os dias de hoje. De acordo com Nehal, Farhan e Sunad (2020), a empresa IBM demonstrou em 2001, a fatoração do número 15 (fatores 3 e 5) usando um computador quântico de 7 qubits.

Além disso, teve-se um avanço na área de Hardware, principalmente na construção de computadores quânticos comerciais. Falcão e Molinari (2016) comenta que em 2011, a empresa canadense D-Wave afirmou ter produzido o primeiro computador quântico comercial, chamado de D-Wave One, uma máquina que poderia processar até 128 qubits.

Em 2019, a empresa IBM criou um computador quântico que foi batizado como "Q System One". A máquina foi disponibilizada para ser acessada em nuvem e possui sistema avançado de refrigeração Capoferri et al. (2021). O processador inicial desse computador trabalhava com 20 qubits ACM (2022), e o objetivo seria executar mais algoritmos quânticos.

Um outro marco também de 2019, como descrito em Capoferri et al. (2021), foi a empresa Google ter anunciado que alcançou a "supremacia quântica", que indica o potencial da computação quântica na resolução de problemas em tempo bem mais razoável do que a computação convencional, como por exemplo, um problema em que um computador clássico resolveria em muitos anos, um computador quântico resolveria em segundos. O anúncio do Google foi feito por meio da utilização do seu processador "Sycamore" de 53 qubits, que levou 200 segundos para realizar uma operação, e estima-se que o supercomputador mais potente da época, o "Summit", desenvolvido pela IBM, levaria algo em torno de 10 mil anos para realizar Caçula (2024). A IBM contestou esse anúncio do Google afirmando que poderia simular tal tarefa em um computador clássico levando 2,5 dias.

Atualmente, tem-se tido avanços significativos na computação quântica, tal como o surgimento de novos computadores quânticos, como os computadores lançados pela IBM, o "Eagle" em 2021, com 127 qubits Capoferri et al. (2021), e o "Osprey" lançado em 2022, com os surpreendentes 433 qubits Hossain (2023), entrando em uma nova era de dispositivos mais potentes e mais desafiadores.

Na Internet, há vários materiais com conteúdos para aprender sobre computação quântica, como é o caso dos cursos que a própria IBM disponibiliza, sendo um deles de forma gratuita para aprender sobre o Qiskit, criada pela empresa IBM (2025e). Além de diversos conteúdos disponíveis em plataformas como o Youtube, também há materiais de aprendizagem da linguagem Q#, criada pela Microsoft Microsoft (2025), e diversos cursos presentes na plataforma EDx, ofertados pela Universidade de Chicago EDx (2025).

2.4 Definições de Linguagem de Programação

Uma linguagem de programação na computação convencional pode ser definida como um método padronizado, que segue determinadas regras a fim de instruir um computador para realizar operações que consistem em um programa Gotardo (2015).

Na computação quântica, o termo linguagem quântica é debatido pois existem recursos que exploram a computação quântica e são construídos em cima de uma linguagem de programação clássica, como o Qiskit, da IBM, que é importado como uma biblioteca do Python e é definido como um kit de desenvolvimento de software. E existem recursos como o Q#, construído pela Microsoft, que é definido como uma linguagem de programação quântica Microsoft (2025).

Esses recursos utilizam princípios da mecânica quântica, como a superposição e o entrelaçamento, não seguindo modelos determinísticos como a computação convencional, mas um modelo mais probabilístico, realizando medições dos qubits, que representam a unidade básica de informação, comparável ao bit na computação clássica José, Piqueira e Lopes (2013).

Existem várias linguagens de programação na computação convencional, cada uma em seu determinado paradigma. Por exemplo, tem-se as linguagens C, C++, C#, Java, JavaScript, Python, entre várias outras.

De acordo com Ferreira e Campos (2025), de forma parecida com o que ocorre nas linguagens de programação na computação clássica, podemos dividir a atuação das ferramentas da computação quântica nas categorias das linguagens convencionais.

2.4.1 Linguagens Imperativas

As linguagens de programação imperativas caracterizam-se por apresentarem a noção de descrever passo a passo as operações que o computador deve executar. Nesse tipo de paradigma, o foco está em como um programa desenvolvido deve realizar uma determinada tarefa.

Na computação convencional, temos como representação as linguagens imperativas C, C++ e Java. Já na computação quântica, temos, por exemplo, as ferramentas QCL, QASM, Silq, Q e Ket.

2.4.2 Linguagens Funcionais

As linguagens ditas como funcionais trabalham com funções no sentido mais matemático, preocupando-se mais em o que deve ser feito, e não em como deve ser feito, e por isso uma linguagem funcional também é dita como declarativa. Nesse tipo de paradigma,

concentra-se mais em passar qual deve ser o resultado esperado, não se preocupando em como chegará a esse resultado.

Podemos citar alguns exemplos de linguagens de programação funcionais na computação clássica, como Haskell, Lisp e Elixir. Na computação quântica , alguns exemplos de tecnologias que podemos citar são: QPL, QFC, QML e Quipper.

2.4.3 Linguagens Multiparadigmas

Quando se diz que uma linguagem é multiparadigma, significa que tal linguagem suporta mais de um paradigma de programação. Com uma linguagem multiparadigma, o desenvolvedor tem liberdade de escolher qual ou quais paradigmas ele poderá utilizar para desenvolver um programa, aproveitando a eficiência que cada paradigma pode oferecer.

Em relação à computação clássica, pode-se citar como exemplos de linguagens multiparadigmas o Python e C#. Na quântica, temos como exemplo as ferramentas Qiskit, Q# e Cirq.

2.5 Princípios Quânticos

A Computação Quântica consiste nos mesmos princípios quânticos da Mecânica Quântica, os quais serão explicados alguns principais, como a superposição, emaranhamento ou entrelaçamento e a interferência da medição.

2.5.1 Superposição

Essa propriedade infere que, antes de uma medição, um determinado qubit pode se encontrar em uma combinação de estados ao mesmo tempo. Um exemplo desse princípio é a experiência denominada de “Gato de Schrödinger”, em que um gato está em uma caixa com um frasco de veneno, e ele pode estar em uma superposição de estados entre vivo e morto, até que ele seja observado.

2.5.2 Emaranhamento

O princípio do emaranhamento ou entrelaçamento quântico descreve o fenômeno em que dois elementos estão tão conectados que, independente da distância entre eles, cada um exerce influência no comportamento do outro, de forma que só perderá essa propriedade caso o par venha a ser perturbado de alguma forma.

2.5.3 Interferência da Medição

A propriedade de interferência da medição, ou apenas medição, é o estado em que, caso um qubit seja observado, ele irá sair da superposição e irá assumir algum valor, seja 0 ou 1.

2.6 Portas comuns na Computação Quântica

De forma análoga à computação clássica, a computação quântica possui portas que são utilizadas para transformar um qubit. As portas mais comuns na Quântica são as portas Hadamard (H), NOT (X) e CNOT (CX).

De acordo com o artigo "Present landscape of quantum computing" Hassija et al. (2020), a porta Hadamard é utilizada para deixar o qubit em estado de superposição com a probabilidade igual do qubit colapsar para 0 ou 1. A porta NOT é utilizada para transformar o qubit em 0 se antes este era 1 e vice-versa. A terceira porta, CNOT, utiliza dois qubits como entrada, um qubit é chamado de controle e o outro qubit é chamado de alvo, e a operação nessa porta é realizada sob as seguintes condições: caso o qubit de controle for medido como 0, nada acontece; caso contrário, é aplicada a porta NOT no qubit alvo.

3 Trabalhos Relacionados

No trabalho do autor Jesus et al. (2021), ele apresenta o experimento do uso da ferramenta de computação quântica Qiskit, a fim de levar o ensino de computação quântica para alunos de graduação em Física e áreas afins, com o objetivo de esclarecer os conceitos básicos e fundamentais de computação quântica. Para as aulas, foi utilizado o pacote Qiskit, criado pela IBM, juntamente com a linguagem Python, para a criação de algoritmos quânticos. Com o uso do Qiskit, os resultados se mostraram positivos, constatando que a ferramenta é acessível para iniciantes e eficaz para implementar algoritmos quânticos.

O autor Angara et al. (2021) também utiliza a ferramenta da IBM, Qiskit, para ensinar computação quântica para jovens do Ensino Médio por meio de workshops. O trabalho tem o propósito de popularizar a computação quântica, a fim de obter uma força de trabalho qualificada na área. Nas aulas, foi apresentado aos alunos o plano *IBM Quantum Experience*, com Qiskit e Jupyter Notebook. Ao final do programa, os alunos avaliaram as aulas com uma média geral de 4,2 em 5, e 60% dos entrevistados disseram que indicariam o workshop a um amigo.

Nesse próximo artigo, do autor Liu (2023), são expostas algumas comparações entre a computação quântica e a computação convencional. São levantados dois assuntos em que a quântica leva vantagem ou até mesmo provoca repensar algoritmos que até então funcionam bem em computadores clássicos. O primeiro assunto é a fatoração de números primos, que é utilizado no algoritmo criptográfico *Rivest-Shamir-Adleman* (RSA). Na quântica, o Algoritmo de Shor é responsável por resolver problemas de fatoração e, de acordo com as melhorias nos computadores quânticos, pode ser que no futuro tenha que se repensar o algoritmo RSA. O segundo assunto tratado foi sobre a particularidade da computação quântica em trabalhar com simulações de moléculas. O autor conclui que é necessário reconhecer o potencial que a computação quântica possui, mesmo ainda estando em um estágio inicial, mas promissor.

No estudo sobre o uso de ferramentas que auxiliam a programação na computação quântica Ferreira e Campos (2025), o autor busca investigar o uso de uma série de ferramentas, por meio de uma pesquisa com 251 participantes que utilizam algum tipo de tecnologia de programação quântica. Com essa pesquisa, conclui-se que o kit de desenvolvimento Qiskit é o mais utilizado pelos programadores quânticos, seguido pelas ferramentas Cirq e Q#. Além disso, tais ferramentas são mais utilizadas em situações de pesquisa e aprendizado.

No seguinte artigo Sharma, Singh e Kumar (2022), o autor busca realizar um estudo comparativo entre modelos de aprendizado de máquina da computação clássica e modelos da computação quântica para análise de sentimentos a partir de resenhas de filmes. Na pesquisa, utilizou-se os algoritmos *Decision Tree*, *Support Vector Machine* e *Gradiente Boosting Classifier* para a computação clássica. Em relação à computação quântica, foi utilizado o algoritmo *Quantum Support Vector Machine*. Ao final, os resultados mostraram-se mais favoráveis ao algoritmo quântico, que classificou um conjunto de dados com 40.000 resenhas com maior precisão e acurácia do que os algoritmos clássicos.

4 Metodologia

O método utilizado no projeto consiste nos seguintes passos:

- Introduzir sobre o tema de Computação Quântica, explicando conceitos básicos.
- Apresentar o kit de desenvolvimento Qiskit, mostrando como instalar e preparar o ambiente para realizar os primeiros programas.
- Comparar a lógica do Qiskit com a lógica de programação convencional, utilizando a linguagem C, abordando tópicos como Estruturas Condicionais e de Repetição.
- Trazer um exemplo prático, como o problema de fatoração, comparando o tempo de execução entre um algoritmo clássico e quântico.

Para desenvolver esse trabalho, foi preciso pesquisar em trabalhos acadêmicos e livros sobre a computação quântica em geral, como por exemplo, definições básicas, surgimento e evolução até os dias atuais, áreas em que a quântica já é aplicada e os benefícios trazidos por ela.

Além disso, foi preciso aprender sobre o kit de desenvolvimento Qiskit, da empresa IBM, e para isso, predominantemente, foram utilizadas como referências os materiais presentes no site oficial da IBM, como documentação do Qiskit, guias e trilhas de treinamento disponibilizadas de forma gratuita.

Para realizar a comparação com a computação clássica na prática, foi escolhida a linguagem C. Tal linguagem foi escolhida porque geralmente é ensinada nos cursos de Computação nas primeiras disciplinas de Introdução à Computação e Estrutura de Dados, e, para complementar, ela é uma linguagem de alto nível, ou seja, mais intuitiva para o entendimento do ser humano.

Essa metodologia foi adotada com o foco em introduzir alguém que não tem conhecimento nenhum em computação quântica, ou que possui algum conhecimento sobre, mas não conhece as aplicações, como pode aplicar, princípios básicos, entre outros tópicos abordados no trabalho. Ao final, tal pessoa terá conhecimento teórico e prático e poderá ter uma base para prosseguir avançando nos estudos desse novo paradigma de computação.

5 Desenvolvimento

No artigo sobre o estudo do uso das tecnologias de desenvolvimento de programação quântica Ferreira e Campos (2025), o autor identificou e descreveu 37 ferramentas, com a criação destas variando entre os anos de 1997 e 2021. De acordo com o levantamento do autor, o qual contou com 251 participantes, as ferramentas quânticas mais utilizadas são Qiskit, seguido por Cirq, OpenQASM e Q#. Como o software Qiskit está configurado como uma das ferramentas mais utilizadas na computação quântica, ele foi escolhido como o objeto de estudo deste trabalho.

O kit de desenvolvimento Qiskit foi criado pela empresa IBM, a qual realiza grandes investimentos na computação quântica. De acordo com a informação da própria IBM IBM (2025a), o início dessa evolução se deu no ano de 2016, quando a IBM lançou o *IBM Quantum Experience* e começou a disponibilizar um computador quântico por meio da Internet. Essa iniciativa fez com que aumentasse o alcance da computação quântica entre as pessoas. Em março de 2017, foi lançada a primeira versão do Qiskit (Qiskit v0.1), que vem sendo continuamente melhorada com o lançamento de outras versões, principalmente com contribuições externas, por ser um software de código aberto (*open-source*).

No artigo sobre a computação quântica com Qiskit Javadi-Abhari et al. (2024), publicado em 2024, o autor informa que o Qiskit foi instalado mais de 6 milhões de vezes, mais de 500 pessoas contribuíram para o desenvolvimento do software e mais de 2000 trabalhos científicos usando Qiskit foram postados na plataforma *arXiv*. O autor ainda afirma com grande margem que Qiskit é o software de computação quântica mais adotado.

5.1 Qiskit

Nessa seção e nas suas subseções, serão abordados tópicos referentes à instalação das ferramentas necessárias para utilizar o Qiskit. É importante saber que, de forma técnica, o Qiskit é um kit de desenvolvimento de software (SDK) presente na linguagem Python. Assim, a programação de algoritmos quânticos é realizada em Python, utilizando as classes e métodos disponibilizados pela biblioteca Qiskit, que é acessada por meio de sua importação no código. Foi utilizada como guia a documentação oficial do Qiskit IBM (2025d). Os programas presentes neste capítulo podem ser encontrados no seguinte repositório: https://github.com/marcustorres02/qiskit-c.

5.1.1 Instalação

Para utilizar o Qiskit, é necessário realizar algumas instalações de recursos e pacotes. Os tópicos a seguir apresentam a instalação da linguagem Python, do recurso de desenvolvimento Visual Studio Code e dos pacotes do Qiskit.

5.1.1.1 Instalar o Python

O primeiro passo para ter acesso ao Qiskit é instalar a linguagem de programação Python. É importante notar que a versão atual do Qiskit é compatível com as versões do Python 3.9 para cima, informação que consta no material Pypi (2025) e mostrada na Figura 2.

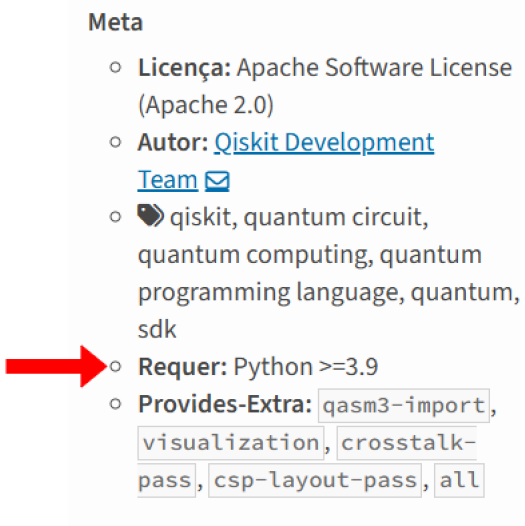

Figura 2 – Requerido Python a partir da versão 3.9 (Fonte: captura de tela do site Pypi (2025), acesso em 18 de fevereiro de 2025).

Caso queira verificar se o Python está instalado na máquina, ou verificar qual versão do Python está instalada, no caso do sistema operacional *Windows*, siga as instruções abaixo.

Pressione simultaneamente as teclas `Windows + R`. Após pressionadas as teclas, aparecerá a janela "Executar". Nessa janela, insira o comando "cmd" e clique no botão "OK", conforme a Figura 3.

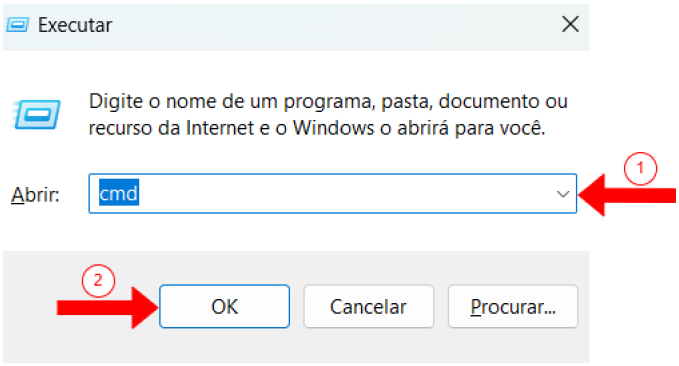

Figura 3 – Atalho do Windows para o Prompt de Comando (Fonte: autor).

Na janela do *Prompt de Comando*, digite o comando: `python --version`. No exemplo da Figura 4, a versão do Python instalada é a 3.12.0.

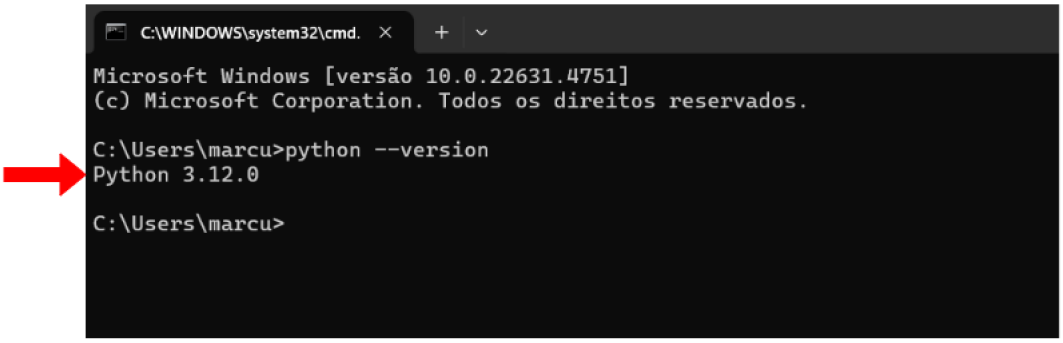

Figura 4 – Versão do Python no Prompt de Comando (Fonte: autor).

Caso a linguagem Python esteja instalada, basta seguir para o próximo tópico. Agora, caso a linguagem não esteja instalada, ou esteja instalada em uma versão não compatível com o Qiskit, o próximo passo é realizar a instalação de uma versão compatível.

No navegador, acesse a página por meio do link: Download Python | Python.org. Na página principal de instalação, pode-se instalar a versão mais recente do Python para Windows, que aparece no início da página, conforme Figura 5, seta marcada com o número 1, ou pode-se instalar versões anteriores. Caso o usuário esteja utilizando um sistema operacional diferente do Windows, na página principal perto do botão de instalar o Python para o Windows, têm-se instruções para instalar para diferentes sistemas operacionais, Figura 5, seta marcada com o número 2.

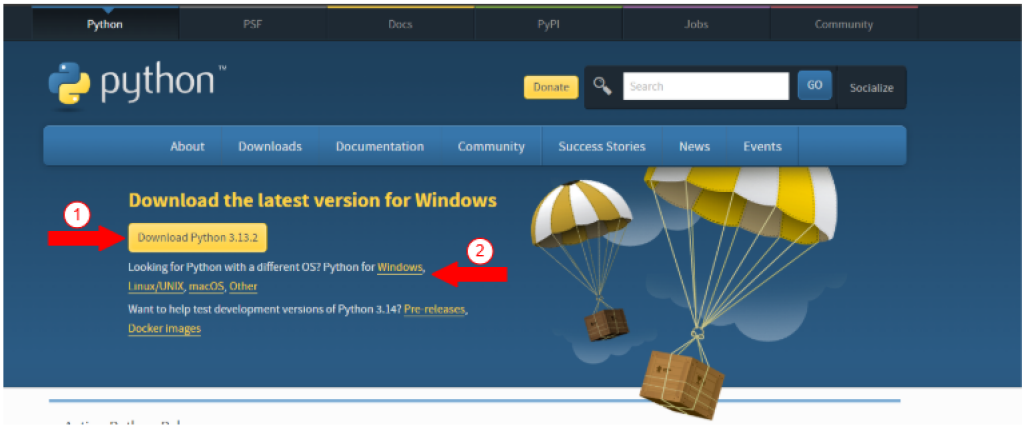

Figura 5 – Página de download da linguagem Python (Fonte: captura de tela do site Download Python | Python.org, acesso em 04 de março de 2025).

Após instalar o executável, abra o arquivo para iniciar o processo de instalação. Como mostrado na Figura 6, habilite os campos "Use admin privileges when installing py.exe" e "Add python.exe to PATH". Caso seja um usuário avançado, é possível personalizar a instalação clicando no botão "Customize installation". Caso contrário, clique em "Install Now" e o Python será instalado e estará pronto para ser utilizado na máquina.

Figura 6 – Processo de instalação do Python (Fonte: captura de tela realizada pelo autor durante o processo de instalação do Python 3.13.2).

Para confirmar se o Python foi instalado corretamente, siga os passos das Figuras 3 e 4.

5.1.1.2 Instalar o Visual Studio Code

O ambiente de desenvolvimento que foi utilizado nos experimentos é o Visual Studio Code, o qual é gratuito e de fácil uso. Para instalar o VS Code, basta acessar o seguinte link no navegador: Download Visual Studio Code, e realizar o download de acordo com o sistema operacional instalado, conforme a Figura 7.

Figura 7 – Página de download Visual Studio Code (Fonte: captura de tela do site Download Visual Studio Code, acesso em 13 de março de 2025).

5.1.1.3 Instalar o Qiskit

Com o Python instalado na máquina, utilize um terminal para criar um ambiente virtual (.venv) para trabalhar com o Qiskit, separando-o de outras aplicações. Para criar um ambiente virtual, insira no terminal o comando: `python -m venv .venv`.

Depois de criado o novo ambiente, ative-o com o seguinte comando no PowerShell: `.venv\Scripts\Activate.ps1`. O próximo passo é instalar os pacotes do Qiskit com o comando: `pip install qiskit`.

É recomendável trabalhar com Qiskit utilizando Jupyter Notebook, instale os pacotes do Jupyter com o comando `pip install jupyter`. Para melhor visualização dos circuitos, também é recomendável instalar as dependências com o comando `pip install qiskit[visualization]`.

Caso queira executar as tarefas em um hardware quântico, também instale o "Qiskit Runtime": `pip install qiskit-ibm-runtime`. Posteriormente, crie uma conta na IBM Quantum Platform para ter acesso aos computadores quânticos disponíveis. Há uma opção para criar uma conta gratuita. Essa conta dá acesso a 10 minutos por mês para usar os computadores quânticos. Após criada a conta, será associado a ela um *API Token*, que será necessário para conectar ao serviço da IBM. A seção de instalação pode ser conferida na documentação oficial do Qiskit IBM (2025c).

Para salvar sua conta em um ambiente confiável, para que não precise passar toda vez a *API Token*, crie um código com a mesma estrutura apresentada abaixo, em que `seu_token` deverá ser substituído pelo token que está na IBM Quantum Platform.

```
from qiskit_ibm_runtime import QiskitRuntimeService
QiskitRuntimeService.save_account(
  token=seu_token,
  channel="ibm_quantum"
)
```

Assim, toda vez que precisar de autenticação, não precisa passar a credencial, apenas inserir o seguinte comando que está abaixo, que já irá carregar a credencial salva.

```
service = QiskitRuntimeService()
```

Na IBM Quantum Platform, a *API Token* estará no painel da aba Dashboard. Logo abaixo do painel, é possível visualizar quanto já se utilizou dos serviços da IBM. Caso a conta seja a opção gratuita, é importante reforçar que é possível utilizar um computador quântico disponível apenas 10 minutos por mês.

5.2 Linguagem C

De acordo com o livro "The C programming language" de Kernighan e Ritchie Kernighan e Ritchie (1988), a linguagem C teve influência da linguagem BCPL, que é uma linguagem criada nos anos 60 e tem a característica de não ter tipos de dados, diferentemente da linguagem C, que contém tipos de dados como caracteres, números inteiros e de ponto flutuante.

Também, os autores comentam que a linguagem C estrutura bem os programas devido aos recursos de controle de fluxo, como as estruturas de condição (`if`, `else`, `switch-case`) e estruturas de repetição (`for`, `while`, `break`, `continue`).

5.3 Comparação do Qiskit com a Linguagem C

Nessa seção, será feita uma comparação entre o kit de desenvolvimento Qiskit e a linguagem de programação clássica C, abordando alguns tópicos apresentados na seção 5.1. A linguagem C foi escolhida por ser uma das linguagens mais utilizadas para aprender lógica de programação atualmente, além de ser uma linguagem que influenciou outras posteriormente, como as linguagens C++ e C#.

5.3.1 Introdução

Para demonstrar um exemplo básico de um programa no Qiskit, mostraremos um exemplo que está presente na documentação oficial na página da IBM.

```
from qiskit import QuantumCircuit
circuito_quantico = QuantumCircuit(2)
```

```
circuito_quantico.h(0)
circuito_quantico.cx(0, 1)
circuito_quantico.draw(output="mpl")
```

Como resultado visível, a Figura 8 mostra o conceito chamado de estado de Bell, em que acontece o emaranhamento quântico. Esse exemplo pode ser comparado a um "Hello, World!" em linguagens de programação da computação clássica, pois na quântica não há a preocupação de exibir texto na tela, mas sim de realizar operações que medem qubits. De forma semelhante a um circuito lógico na computação clássica, um circuito quântico consiste na aplicação de portas quânticas em qubits, de acordo com um algoritmo definido.

No programa mostrado, as operações realizadas são:

1. Cria um novo circuito com dois qubits.
2. Aplica a porta Hadamard ao qubit 0.
3. Aplica a porta CNOT no qubit 1, controlada pelo qubit 0.
4. Desenho do circuito.

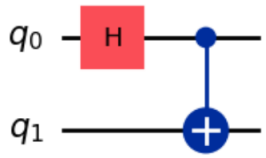

Figura 8 – Circuito quântico com dois qubits (Fonte: execução do código adaptado de IBM (2025b)).

Diferente de um programa quântico, que por muitas vezes não possui um método nativo para apresentar algo na tela pois não é seu objetivo, na computação clássica as linguagens possuem funções que conseguem apresentar uma determinada mensagem e informação para o usuário. Com isso, por diversas vezes, o programa mais básico que se aprende quando se está aprendendo uma linguagem de programação clássica é imprimir uma mensagem na tela, e isso é válido para a linguagem C.

```
#include <stdio.h>
int main() {
    printf("Hello World!");
    return 0;
}
```

No programa acima, é um exemplo de um programa simples na linguagem C em que é possível apresentar a mensagem "Hello World!" na tela por meio da função `printf` que está disponível na biblioteca `stdio.h`.

Além disso, C é uma linguagem conhecida por exigir tipos de dados nas declarações de variáveis, e têm-se quatro tipos de dados mais conhecidos, conhecidos como primitivos, que são: char (caracteres); int (números inteiros); float (números de ponto flutuante com precisão simples); double (números de ponto flutuante com maior precisão).

```
#include <stdio.h>
int main() {
    char c = 'a';
    int numero_inteiro = 0;
    float numero_real_simples = 0;
    double numero_real_dupla = 0;
    printf("Digite um caractere: ");
    scanf("%c", &c);
    printf("Digite um numero inteiro: ");
    scanf("%d", &numero_inteiro);
    printf("Digite um numero real (float): ");
    scanf("%f", &numero_real_simples);
    printf("Digite um numero real (double): ");
    scanf("%lf", &numero_real_dupla);
    printf("\n");
    printf("Caractere digitado: %c\n", c);
    printf("Inteiro digitado: %d\n", numero_inteiro);
    printf("Numero real (float) digitado: %.10f\n", numero_real_
       simples);
    printf("Numero real (double) digitado: %.10f\n", numero_real_
       dupla);
    return 0;
}
```

O resultado do código acima está presente na Figura 9, em que se mostra o comportamento dos tipos primitivos da linguagem C.

```
Digite um caractere: m
Digite um numero inteiro: 10
Digite um numero real (float): 2.1234567890
Digite um numero real (double): 2.1234567890

Caractere digitado: m
Inteiro digitado: 10
Numero real (float) digitado: 2.1234567165
Numero real (double) digitado: 2.1234567890
```

Figura 9 – Tipos primitivos linguagem C (Fonte: autor).

5.3.2 Estrutura Condicional

Em relação à implementação de estruturas condicionais dentro do Qiskit, existem alguns métodos, como `c_if`, `if_test`, `if_else`, `IfElseOp`, `switch/case`, entre outros que estão sendo desenvolvidos para versões futuras. Para exemplificar, abordaremos os métodos `c_if`, `if_test` e `IfElseOp`.

O método `c_if` é responsável por fazer um controle de fluxo no programa, atuando como um `if` clássico que normalmente conhecemos, porém com o uso de apenas uma condição. Sua estrutura é caracterizada pela seguinte forma `c_if(registrador clássico, valor)`.

```
from qiskit import QuantumCircuit, ClassicalRegister,
   QuantumRegister
qr = QuantumRegister(1, "q")
cr = ClassicalRegister(1, "c")
qc = QuantumCircuit(qr, cr)
qc.h(qr[0])
qc.measure(qr[0], cr[0])
qc.x(qr[0]).c_if(cr, 1)
```

No trecho de código acima, o uso do `c_if` é aplicado para realizar a condição em que, caso o registrador clássico seja igual a 1, então será aplicada a porta X no qubit quântico. O método possui esse uso em específico, sendo aplicável para uma única porta, com uma condição e sem suporte para o comando `else`.

O método `if_test` também tem suporte para uma única condição, entretanto, pode-se colocar mais operações e tem suporte para uma estrutura com o comando `else`. Sua declaração é feita na forma `with if_test((registrador clássico, valor))`.

```
from qiskit import QuantumCircuit, ClassicalRegister,
   QuantumRegister
qr = QuantumRegister(1, "q")
cr = ClassicalRegister(1, "c")
qc = QuantumCircuit(qr, cr)
qc.h(qr[0])
qc.measure(qr[0], cr[0])
with qc.if_test((cr, 1)) as else_:
    qc.x(qr[0])
    qc.h(qr[0])
with else_:
    qc.h(qr[0])
```

O programa acima demonstra o uso do `if_test`. Caso o registrador clássico for medido como 1, são aplicadas as operações de porta X e Hadamard no qubit. Caso con-

trário, ou seja, o registrador clássico for medido como 0, será aplicada apenas a porta Hadamard no qubit.

Um outro recurso que pode ser usado é o `IfElseOp`, em que podemos passar em uma única linha o recurso `IfElseOp`, geralmente com três parâmetros `IfElseOp(condição, instruções caso verdadeiro, instruções caso falso (opcional))`.

```
from qiskit import QuantumCircuit, ClassicalRegister,
   QuantumRegister
from qiskit.circuit import IfElseOp
qr = QuantumRegister(1, "q")
cr = ClassicalRegister(1, "c")
qc = QuantumCircuit(qr, cr)
qc.h(qr[0])
qc.measure(qr[0], cr[0])
escopo_true = QuantumCircuit(1)
escopo_true.x(0)
escopo_true.h(0)
escopo_false = QuantumCircuit(1)
escopo_false.h(0)
exemplo = IfElseOp((cr, 1), escopo_true, escopo_false)
qc.append(exemplo, [qr[0]])
```

O trecho de código acima exemplifica o uso do `IfElseOp`. No programa, foram criados dois circuitos que possuem o prefixo "escopo". Caso o registrador clássico seja medido como 1, as operações presentes em `escopo_true` são aplicadas no qubit. Caso contrário, são aplicadas as operações em `escopo_false`. Ao final, realiza-se a operação `append`, para que as operações sejam aplicadas no circuito.

Como o Qiskit é executado dentro da linguagem Python, pode-se manipular o fluxo com comandos do Python, como o programa abaixo mostra.

```
from qiskit import QuantumCircuit, ClassicalRegister,
   QuantumRegister
valor = int(input("Digite 1 para aplicar porta x no circuito ou
   digite 2 para aplicar porta hadamard"))
while valor != 1 and valor != 2:
    print("Valor invalido\n")
    valor = int(input("Digite 1 para aplicar porta x no circuito
       ou digite 2 para aplicar porta hadamard"))
qr = QuantumRegister(1, "q")
cr = ClassicalRegister(1, "c")
qc = QuantumCircuit(qr, cr)
if valor == 1:
```

```
    qc.x(qr[0])
elif valor == 2:
    qc.h(0)
qc.measure(qr[0], cr[0])
qc.draw(output="mpl")
```

Nesse caso, pede-se para o usuário digitar 1 caso queira que aplique a porta x no circuito, ou digitar 2 caso queira que aplique a porta hadamard. Com o valor digitado pelo usuário, é utilizado o comando `if` do Python para ter esse controle da seleção da operação realizada no circuito.

Na linguagem C, para realizar condições, há algumas funções disponíveis como `if`, `else`, e `switch-case`. Para esse tópico, serão abordadas as funções `if` e `else`. Além desses recursos, também tem-se o auxílio de operadores lógicos e relacionais.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main() {
    srand(time(NULL));
    int resultado = rand() % 2;
    if (resultado == 0) {
        printf("Resultado: Cara\n");
    } else {
        printf("Resultado: Coroa\n");
    }
    return 0;
}
```

O programa acima simula o lançamento de uma moeda, em que gera um número inteiro aleatório (`rand()`) e divide esse valor por 2. O resto da divisão será armazenado na variável inteira `resultado`, que terá o valor 0 ou 1. Caso o `resultado` seja igual a 0, mostrar na tela `Resultado: Cara`, caso contrário, mostrar `Resultado: Coroa`.

O comando `if` é caracterizado por seguir a sintaxe `if (condição)`, nesse caso, `condição` é igual a `resultado == 0`, em que `==` é um operador relacional que significa igualdade no sentido de comparação. Assim, se o resultado for igual a 0, será executado o trecho de código que estiver dentro das chaves do comando `if`. Caso a condição não seja satisfeita no `if`, será executado o trecho de código dentro das chaves do comando `else`.

Pode-se ter vários `if` dentro do programa, e também vários `else`, sendo que para utilizar o comando `else` deve ter sido utilizado pelo menos um `if` anteriormente. Além disso, pode-se ter essa estrutura de forma aninhada, ou seja, um bloco de `if else` dentro de um outro bloco com `if else`.

5.3.3 Estrutura de Repetição

Para tratativa de estrutura de repetição no Qiskit, temos alguns recursos, como `for_loop`, `while_loop`, `ForLoopOp` e `WhileLoopOp`. Para exemplificar o uso de um loop em um programa, usaremos os controles `ForLoopOp` e `WhileLoopOp`.

O `ForLoopOp` possui uma estrutura da seguinte forma `ForLoopOp(quantidade de vezes que o loop irá repetir, contador interno, operações)`.

```
from qiskit import QuantumCircuit, QuantumRegister
from qiskit.circuit import ForLoopOp
qr = QuantumRegister(1, "q")
qc = QuantumCircuit(qr)
operacoes = QuantumCircuit(1)
operacoes.x(0)
operacoes.h(0)
exemplo = ForLoopOp(range(5), None, operacoes)
qc.append(exemplo, [qr[0]])
```

O programa acima mostra o funcionamento do `ForLoopOp`. No exemplo, o bloco chamado de "operacoes" contém as operações de aplicar a porta X e Hadamard, e essas operações são realizadas cinco vezes no qubit e é feita a adição ao circuito.

A classe `WhileLoopOp` tem um formato um pouco diferente do `ForLoopOp`.

```
from qiskit import QuantumCircuit, ClassicalRegister,
   QuantumRegister
from qiskit.circuit import WhileLoopOp
qr = QuantumRegister(1, "q")
cr = ClassicalRegister(1, "c")
qc = QuantumCircuit(qr, cr)
qc.h(qr[0])
qc.measure(qr[0], cr[0])
operacoes = QuantumCircuit(1, 1)
operacoes.x(0)
operacoes.h(0)
operacoes.measure(0, 0)
loop_while = WhileLoopOp((cr, 1), operacoes)
qc.append(loop_while, [qr[0]], [cr[0]])
```

O algoritmo acima mostra o uso do `WhileLoopOp`. Geralmente, `WhileLoopOp` tem a seguinte estrutura `WhileLoopOp(condição, operacoes)`. O primeiro parâmetro é uma condição (`(cr, 1)`), que significa, enquanto o registrador clássico for medido como 1. O segundo parâmetro representa as operações que estão presentes no qubit "operacoes" e que serão realizadas caso a condição se mantenha verdadeira. O programa ficará em

loop até que o registrador clássico seja medido como 0. Ao final, é feita a atualização do circuito.

Para começar uma estrutura de repetição na linguagem C, existem comandos como o `for`, `while` e também o `do while`, que são executados até que uma certa condição seja atendida. Para exemplificar, serão utilizados os comandos `for` e `while`.

O exemplo abaixo mostra o funcionamento do uso do `for` no mesmo código do cara e coroa. O comando possui a estrutura `for(inicialização; condição; atualização)`.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main() {
    srand(time(NULL));
    int resultado = 0;
    for(int i = 0; i < 10; i++) {
        printf("Rodada %d\n", i + 1);
        resultado = rand() % 2;
        if (resultado == 0) {
            printf("Resultado: Cara\n");
        } else {
            printf("Resultado: Coroa\n");
        }
    }
    return 0;
}
```

Nesse exemplo, as instruções que estão dentro do `for` serão executadas até que a condição não seja satisfeita, ou seja, 10 vezes. A cada iteração, a variável de controle `i`, geralmente chamada de contador, está sendo incrementada em um de seu valor. Quando `i`, que começou em 0, chegar ao valor 10, o programa sairá do `loop` e encerrará.

O próximo exemplo será o mesmo código, porém utilizando o comando `while`.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main() {
    srand(time(NULL));
    int i = 0, resultado = 0;
    while (i < 10) {
        printf("Rodada %d\n", i + 1);
        resultado = rand() % 2;
        if (resultado == 0) {
```

```
            printf("Resultado: Cara\n");
        } else {
            printf("Resultado: Coroa\n");
        }
        i++;
    }
    return 0;
}
```

Como apresentado no programa acima, o `while` tem uma estrutura diferente do `for`, em que é passada apenas a condição entre parênteses `while (condição)`. Entretanto, ainda tem-se as características do `for`, com a variável de controle `i` sendo inicializada antes do `while` e sendo atualizada como a última instrução dentro do `while`. Essa atualização é importante para que não se tenha o que é chamado de `loop infinito`, o que significaria que as instruções seriam executadas por tempo e quantidade indeterminadas, até acontecer um `overflow` de memória, por exemplo.

5.3.4 Algoritmo de Fatoração

Para um exemplo prático de comparação entre as duas ferramentas, Qiskit e C, utilizaremos o problema da fatoração de números inteiros. Esse exemplo foi escolhido por ser um dos problemas que a computação quântica propõe evoluir, com o chamado algoritmo de Shor, e o que torna o algoritmo criptográfico RSA seguro atualmente. O objetivo será comparar o tempo de execução dos algoritmos aplicados em cada paradigma, com uma mesma entrada, que será o número inteiro 15 (fatores 3 e 5), a fim de realizar uma comparação da abordagem clássica e quântica.

No Qiskit, foi utilizado o algoritmo de Shor, que de forma simplificada, consiste em encontrar o período r de uma função $f(x) = a^x \bmod N$, onde a é um número inteiro aleatório e x é um número inteiro. Para implementar o algoritmo de Shor no Qiskit, foi utilizado como base o código presente no repositório do Qiskit no GitHub e que pode ser encontrado pela seguinte url: Repositório algoritmo de Shor Qiskit. Foram realizadas pequenas adaptações para a versão 1.4.1 do Qiskit, além de inserir a biblioteca `time` para medir o tempo de execução do algoritmo. Para deixar mais equiparado com a computação clássica, um simulador de computador quântico foi usado para executar o algoritmo, que é capaz de reduzir ruídos e tratar erros, tendo mais consistência e velocidade do que um computador quântico real disponível de forma gratuita pela IBM.

Na linguagem C, o algoritmo utilizado foi o chamado divisão por tentativa, porém otimizado para esse problema. Primeiro, ele remove todos os fatores iguais a 2, em seguida, testa fatores ímpares sucessivos, e por fim, se restar um número maior que 2, este é um fator primo final.

```
void fatorar(int n) {
    printf("Fatores primos do numero %d: ", n);
    while (n % 2 == 0) {
        printf("2 ");
        n = n/2;
    }
    for (int i = 3; i * i <= n; i += 2) {
        while (n % i == 0) {
            printf("%d ", i);
            n = n/i;
        }
    }
    if (n > 2) {
        printf("%d", n);
    }
    printf("\n");
}
```

Em relação ao resultado final, tanto em Qiskit quanto em C, foram encontrados os fatores 3 e 5 para o número inteiro 15 de entrada. O tempo de execução no Qiskit implementando o algoritmo de Shor foi de 0.5 segundos. Já utilizando o algoritmo de divisão por tentativa em C, foi de 0.002 segundos.

5.3.5 Resultados

Ao apresentar como um programa é estruturado utilizando o kit de desenvolvimento Qiskit e utilizando a linguagem clássica C, nota-se diferenças marcantes nas estruturas dos programas, devido à diferença de paradigmas. Por exemplo, no Qiskit, o objeto de estudo é o qubit e é importante declarar a quantidade de qubits que serão utilizados e as operações que serão executadas no circuito.

Entretanto, também é possível notar semelhanças, pois na maioria dos casos, os registradores clássicos, que servem para medir o qubit, são utilizados nos programas. Por mais complexo que possa ser o fenômeno da superposição, ao final, o qubit deverá colapsar para um estado ao ser observado.

Outro ponto de destaque é que em alguns casos, um programa quântico pode se assemelhar muito a um programa de linguagem clássica, aplicando operações lógicas, seleção e repetição.

Em relação à comparação das soluções para o problema de fatoração de inteiros para os dois paradigmas, foi analisado que, para a situação de fatorar o número inteiro 15, mesmo com simulador quântico, o algoritmo de Shor levou mais tempo para encontrar

os fatores primos (0.5 segundos) do que o algoritmo clássico em C (0.002 segundos). Isso pode ser justificado pela quantidade de operações que o algoritmo de Shor realiza ser maior do que o algoritmo aplicado em C, maior custo computacional a fim de simular circuitos quânticos, e também, a linguagem C pode oferecer melhor desempenho por ser uma linguagem compilada e não interpretada como Python.

6 Conclusão

A computação quântica surge como uma nova vertente de se fazer computação, utilizando princípios da Mecânica Quântica para superarmos limites da computação convencional. Isso não quer dizer que ela substituirá a computação que conhecemos atualmente, mas sim que as duas irão coexistir para ajudar na resolução de problemas complexos de nossas vidas.

É importante, para o crescimento do conhecimento sobre a computação quântica, discutir mais os tópicos que a envolvem, e mostrar os avanços que estão acontecendo atualmente no mundo devido a ela. Isso pode começar pelos cursos de computação das instituições brasileiras ao incluir o ensino de Computação Quântica nas grades curriculares dos alunos, até mesmo como uma disciplina optativa, mas disponível, para despertar o interesse e difundir o ensino.

É notável observar que a pesquisa em Computação Quântica está cada vez mais acessível, com ferramentas disponíveis de forma gratuita para implementar o que é descrito na teoria. Antes, o que era desenvolvido de forma fechada e restrita a somente algumas pessoas e organizações, hoje pode-se ter acesso a computadores quânticos reais e simuladores para resolver problemas.

Como uma forma de apresentar o tema, este trabalho apresentou conceitos introdutórios para iniciar o estudo em computação quântica, e incentivar a quem tenha interesse a avançar nos próximos desafios. O trabalho se apresenta como uma forma de guia para quem tem conhecimento em programação na linguagem C e está ingressando no universo da computação quântica.

A referência utilizada para introduzir na programação de computadores quânticos foi o kit de desenvolvimento Qiskit. Tal ferramenta, criada e atualizada pela empresa IBM, é uma das principais ferramentas de computação quântica utilizadas atualmente. Como demonstrado, é preciso instalar a linguagem Python e, posteriormente, estará apto a instalar as dependências do Qiskit. Além disso, o trabalho demonstrou como é fácil obter acesso a um computador quântico real da IBM presente na nuvem de forma gratuita na seção sobre o Qiskit, caso opte em utilizar no lugar de um simulador.

Para se aproximar e fazer uma relação com a computação clássica, foi realizada uma comparação do Qiskit e da linguagem clássica C, a fim de demonstrar os objetivos de cada uma e como elas são utilizadas em cada paradigma. Também, pode-se perceber semelhanças entre as lógicas dos paradigmas, como mostradas na seção de estrutura condicional e estrutura de repetição de cada uma das ferramentas. Por fim, foi exposta a comparação de um algoritmo clássico e quântico atuando no problema da fatoração, com

o objetivo de apresentar o tempo de execução de cada algoritmo.

6.1 Propostas para Trabalhos Futuros

Como proposta para trabalho futuro, está a criação de uma disciplina optativa nos cursos de Computação da Universidade Federal de Uberlândia (UFU) sobre Computação Quântica, a fim de começar a difundir o conhecimento sobre essa área da computação. Outra proposta é incentivar trabalhos explorando mais sobre as ferramentas de programação quântica, apresentando outras tecnologias além do Qiskit.

Referências

ACM, C. of the. **Assessing the Quantum-Computing Landscape**. 2022. Acessado em 09 de dezembro de 2024. Disponível em: <https://cacm.acm.org/research/assessing-the-quantum-computing-landscape/>. Citado na página 14.

ADVISORS, S. S. G. **Quantum Computing: Evolution or Revolution?** 2024. Acessado em 03 de junho de 2025. Disponível em: <https://www.ssga.com/library-content/assets/pdf/global/equities/2024/quantum-computing-evolution-or-revolution.pdf>. Citado na página 8.

ANGARA, P. P.; STEGE, U.; MACLEAN, A.; MÜLLER, H. A.; MARKHAM, T. Teaching quantum computing to high-school-aged youth: A hands-on approach. **IEEE Transactions on Quantum Engineering**, IEEE, v. 3, p. 1–15, 2021. Citado na página 18.

CAÇULA, M. E. F. Computação quântica: Estado atual e aplicações futuras. **Revista Caparaó**, v. 6, n. 1, p. e93–e93, 2024. Citado na página 14.

CAPOFERRI, B. C.; RAMEH, G. D.; FREZZATTI, H. M.; MAKUTA, L. S. A história da computaçao quântica. 2021. Citado 2 vezes nas páginas 13 e 14.

CARVALHO, L.; LAVOR, C.; MOTTA, V. Caracterização matemática e visualização da esfera de bloch: ferramentas para computação quântica. **Trends in Computational and Applied Mathematics**, v. 8, n. 3, p. 351–360, 2007. Citado na página 10.

DAMASCO, R. Computação quântica. **Revista LIFT papers**, v. 2, n. 2, 2019. Citado 2 vezes nas páginas 10 e 12.

EDX. **Learn quantum computing with online courses and programs**. 2025. Acessado em 16 de janeiro de 2025. Disponível em: <https://www.edx.org/learn/quantum-computing>. Citado na página 14.

FALCÃO, G. B. N.; MOLINARI, R. Histórico, estado e perspectivas da tecnologia da computação quântica. **UNILUS Ensino e Pesquisa**, v. 13, n. 31, p. 88–100, 2016. Citado 2 vezes nas páginas 13 e 14.

FERREIRA, F.; CAMPOS, J. An exploratory study on the usage of quantum programming languages. **Science of Computer Programming**, v. 240, p. 103217, 2025. ISSN 0167-6423. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0167642324001400>. Citado 3 vezes nas páginas 15, 18 e 21.

FLÖTHER, F. F. The state of quantum computing applications in health and medicine. **Research Directions: Quantum Technologies**, v. 1, p. e10, 2023. Citado na página 11.

GOTARDO, R. Linguagem de programação. **Rio de Janeiro: Seses**, p. 34, 2015. Citado na página 15.

GRAPH, M. T. **O que é computação quântica: O futuro do processamento de informações**. 2023. Acessado em 29 de maio de 2025. Disponível em: <https://mindthegraph.com/blog/pt/o-que-e-computacao-quantica/>. Citado 2 vezes nas páginas 4 e 11.

HASSIJA, V.; CHAMOLA, V.; SAXENA, V.; CHANANA, V.; PARASHARI, P.; MUMTAZ, S.; GUIZANI, M. Present landscape of quantum computing. **IET Quantum Communication**, Wiley Online Library, v. 1, n. 2, p. 42–48, 2020. Citado na página 17.

HOSSAIN, K. A. The potential and challenges of quantum technology in modern era. **Scientific Research Journal**, v. 11, n. 6, 2023. Citado na página 14.

IBM. **O que é computação quântica**. 2024. Acessado em 20 de novembro de 2024. Disponível em: <https://www.ibm.com/br-pt/topics/quantum-computing>. Citado na página 12.

_____. **Celebrating 7 years of Qiskit**. 2025. Acessado em 30 de janeiro de 2025. Disponível em: <https://www.ibm.com/quantum/qiskit/history>. Citado na página 21.

_____. **Hello World**. 2025. Acessado em 25 de junho de 2025. Disponível em: <https://docs.quantum.ibm.com/guides/hello-world>. Citado 2 vezes nas páginas 4 e 27.

_____. **Install Qiskit**. 2025. Acessado em 30 de março de 2025. Disponível em: <https://docs.quantum.ibm.com/guides/install-qiskit>. Citado na página 25.

_____. **Introduction to Qiskit**. 2025. Acessado em 23 de março de 2025. Disponível em: <https://docs.quantum.ibm.com/guides>. Citado na página 21.

_____. **Qiskit**. 2025. Acessado em 16 de janeiro de 2025. Disponível em: <https://www.ibm.com/quantum/qiskit>. Citado na página 14.

INSIDER, Q. **The History of Quantum Computing You Need to Know [2024]**. 2020. Acessado em 03 de dezembro de 2024. Disponível em: <https://thequantuminsider.com/2020/05/26/history-of-quantum-computing/>. Citado na página 13.

JAVADI-ABHARI, A.; TREINISH, M.; KRSULICH, K.; WOOD, C. J.; LISHMAN, J.; GACON, J.; MARTIEL, S.; NATION, P. D.; BISHOP, L. S.; CROSS, A. W. et al. Quantum computing with qiskit. **arXiv preprint arXiv:2405.08810**, 2024. Citado na página 21.

JESUS, G. F. d.; SILVA, M. H. F. da; DOURADO, T. G.; GALVÃO, L. Q.; SOUZA, F. G. de O.; CRUZ, C. Computação quântica: uma abordagem para a graduação usando o qiskit. **Revista Brasileira de Ensino de Física**, SciELO Brasil, v. 43, p. e20210033, 2021. Citado na página 18.

JOSÉ, M. A.; PIQUEIRA, J. R. C.; LOPES, R. d. D. Introdução à programação quântica. **Revista Brasileira de Ensino de Física**, SciELO Brasil, v. 35, p. 1–9, 2013. Citado na página 15.

KERNIGHAN, B. W.; RITCHIE, D. M. **The C programming language**. [S.l.]: prentice-Hall, 1988. Citado na página 26.

LIU, Q. Comparisons of conventional computing and quantum computing approaches. **Highlights in Science, Engineering and Technology**, v. 38, p. 502–507, 2023. Citado na página 18.

MATTIELO, F.; SILVA, G. G.; AMORIM, R. G. G. de et al. Decifrando a computação quântica. **Biblioteca Digital Link School of Business**, 2012. Citado na página 12.

MCMAHON, D. **Quantum computing explained**. [S.l.]: John Wiley & Sons, 2008. Citado na página 10.

MICROSOFT. **Introduction to the quantum programming language Q**. 2025. Acessado em 16 de janeiro de 2025. Disponível em: <https://learn.microsoft.com/en-us/azure/quantum/qsharp-overview>. Citado 2 vezes nas páginas 14 e 15.

NEHAL, A. S.; FARHAN, M.; SUNAD, Y. Quantum cryptography-breaking rsa encryption using quantum computing with shor's algorithm. **International Journal For Technological Research in Engineering**, v. 8, n. 1, 2020. Citado na página 14.

PIZA, A. F. R. de T. **Mecânica quântica**. [S.l.]: Edusp, 2003. Citado na página 13.

PYPI. **qiskit**. 2025. Acessado em 18 de fevereiro de 2025. Disponível em: <https://pypi.org/project/qiskit/>. Citado 2 vezes nas páginas 4 e 22.

RASOOL, R. U.; AHMAD, H. F.; RAFIQUE, W.; QAYYUM, A.; QADIR, J.; ANWAR, Z. Quantum computing for healthcare: A review. **Future Internet**, MDPI, v. 15, n. 3, p. 94, 2023. Citado na página 11.

SCHOLTEN, T. L.; WILLIAMS, C. J.; MOODY, D.; MOSCA, M.; HURLEY, W.; ZENG, W. J.; TROYER, M.; GAMBETTA, J. M. et al. Assessing the benefits and risks of quantum computers. **arXiv preprint arXiv:2401.16317**, 2024. Citado na página 12.

SHAHWAR, T.; ZAFAR, J.; ALMOGREN, A.; ZAFAR, H.; REHMAN, A. U.; SHAFIQ, M.; HAMAM, H. Automated detection of alzheimer's via hybrid classical quantum neural networks. **Electronics**, MDPI, v. 11, n. 5, p. 721, 2022. Citado na página 12.

SHARMA, D.; SINGH, P.; KUMAR, A. A comparative study of classical and quantum machine learning models for sentimental analysis. **arXiv e-prints**, p. arXiv–2209, 2022. Citado na página 19.

ZHU, S.; YU, T.; XU, T.; CHEN, H.; DUSTDAR, S.; GIGAN, S.; GUNDUZ, D.; HOSSAIN, E.; JIN, Y.; LIN, F. et al. Intelligent computing: the latest advances, challenges, and future. **Intelligent Computing**, AAAS, v. 2, p. 0006, 2023. Citado 2 vezes nas páginas 8 e 12.