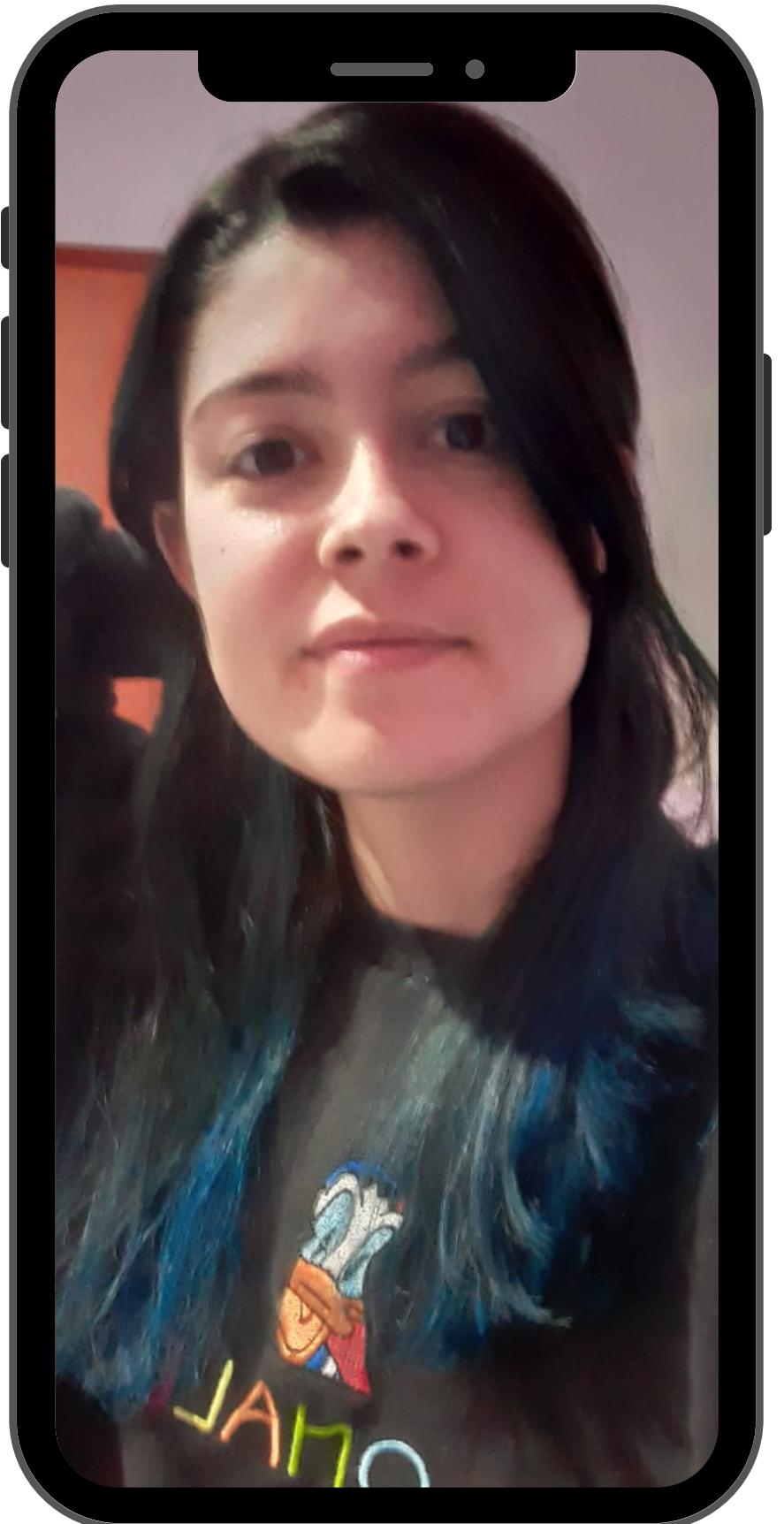


*Primeiros Passos na*

# COMPUTAÇÃO QUÂNTICA

Amanda G. Valério



# Quem sou eu (ou como cheguei aqui)

- Aluna do 6º período de Ciência da Computação no IFSuldeMinas
- Iniciação Científica de 2019 à 2020 (um ano e meio)  
Análise e Comparação da Complexidade de Algoritmos de Busca utilizando os Paradigmas Clássico e Quântico - Valério, A.; Brant-Ribeiro, T. (2019)
- Membra co-fundadora do PyLadies Sul de Minas

CONTATO:



[amanda.valerio@alunos.ifluminense.edu.br](mailto:amanda.valerio@alunos.ifluminense.edu.br)



Mas o que é

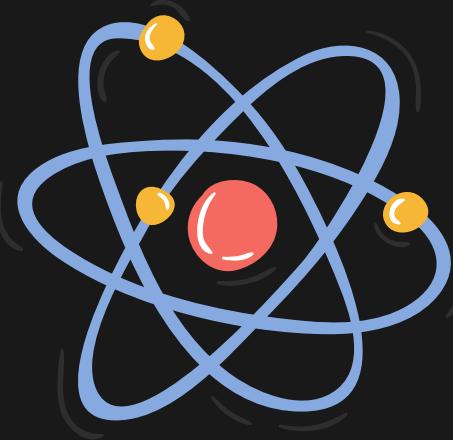
# **COMPUTAÇÃO QUÁNTICA?**



Mas o que é

# **COMPUTAÇÃO QUÂNTICA?**

É um novo paradigma de computação que utiliza as propriedades da física quântica para aumentar a velocidade de processamento

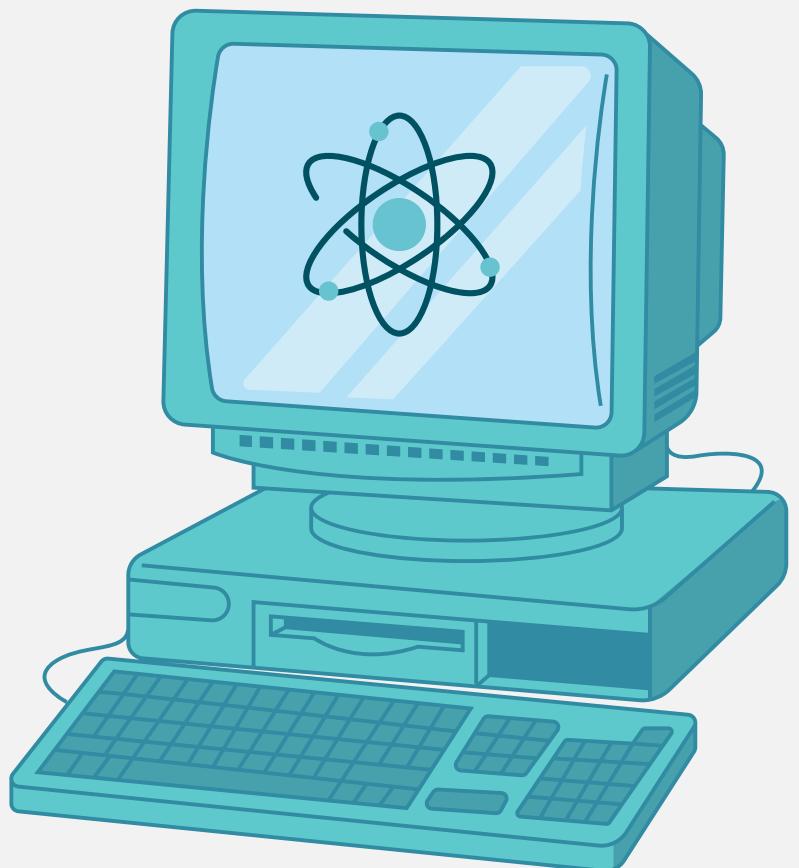


E DE ONDE  
SURGIU?



# Um Breve Resumo

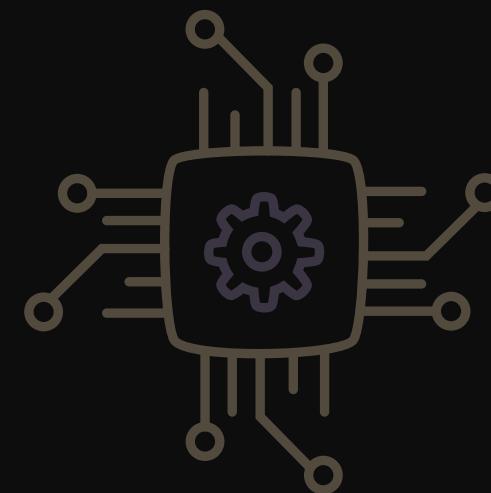
- Surgimento da Mecânica Quântica
- Simulação de experimentos
- Lei de Moore
- Ausência de tecnologia suficiente na época
- Interferência nos processadores atuais
- "Corrida" pela supremacia



Vamos entender um pouco melhor tudo isso

Começando pelos

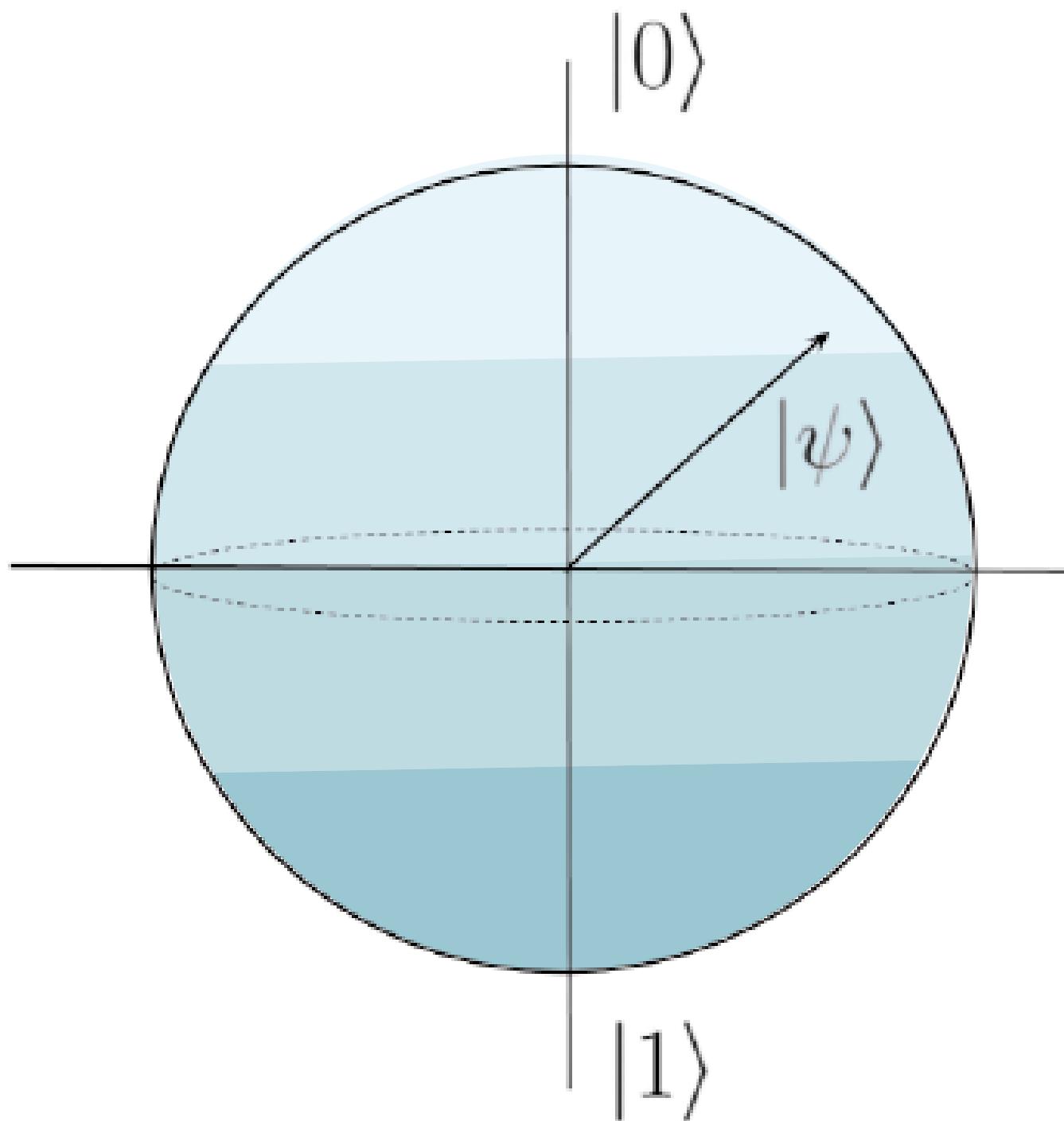
**QUBITS**



# *Bit* x *QuBit*

- Pulses de frequências elétricas
  - Valor: 0 ou 1
  - Propriedades clássicas
- 
- Partículas
  - Probabilidade de estados
  - 0 e 1
  - Propriedades quânticas

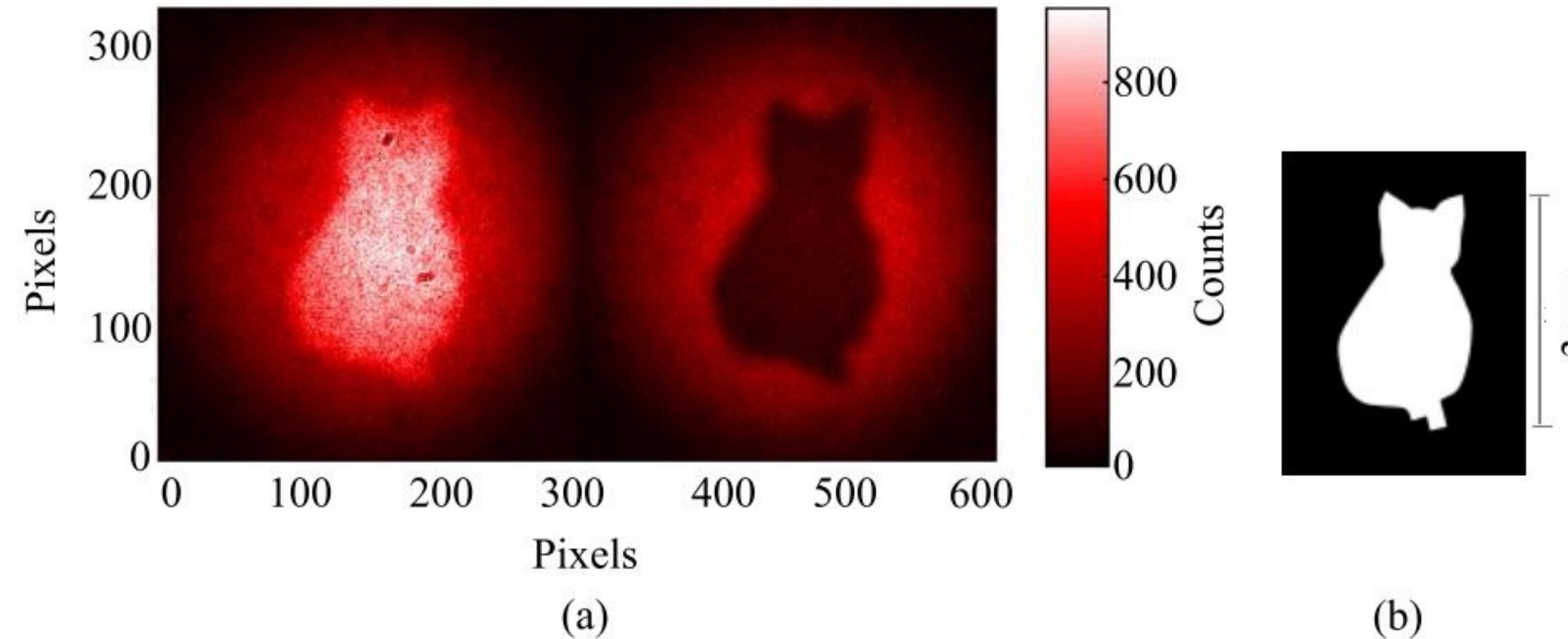
# O QuBit



- Natureza probabilística
- Superposição
- Medidas colapsam o sistema

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

# Emaranhamento



Fonte: Imagem extraída de:  
LEMOS, Gabriela Barreto et al.  
Quantum imaging with  
undetected photons. Nature, v.  
512, n. 7515, p. 409-412, 2014.

$$|\psi\rangle = \alpha |00\rangle + \beta |01\rangle + \gamma |10\rangle + \delta |11\rangle$$

# Aplicando o Entrelaçamento

$|0\rangle$

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$|1\rangle$

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

# Aplicando o Entrelaçamento

$|0\rangle$

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$|1\rangle$

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} =$$

# Aplicando o Entrelaçamento

$|0\rangle$

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$|1\rangle$

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 & \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} \\ 1 & \end{pmatrix}$$

# Aplicando o Entrelaçamento

$|0\rangle$

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$|1\rangle$

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 & \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} \\ 1 & \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

# Aplicando o Entrelaçamento

$|0\rangle$

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$|1\rangle$

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

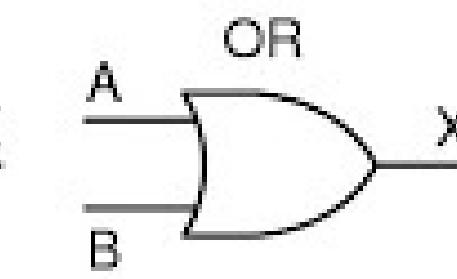
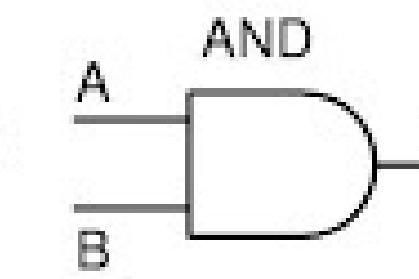
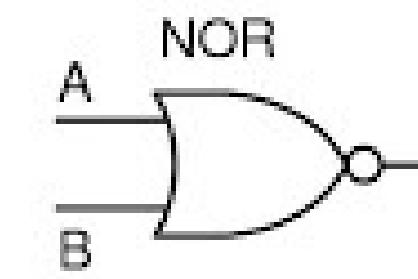
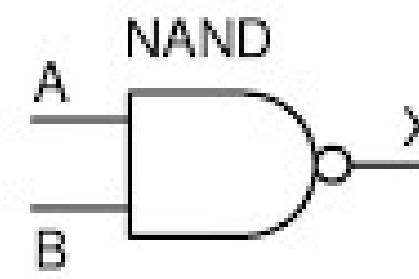
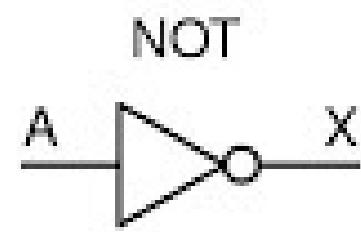
$$\begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 & \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} \\ 1 & \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

Combinação  
Linear dos  
estados

# Portas Lógicas & Portas Quânticas



# Portas Lógicas Clássicas



A	X
0	1
1	0

(a)

A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

(b)

A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

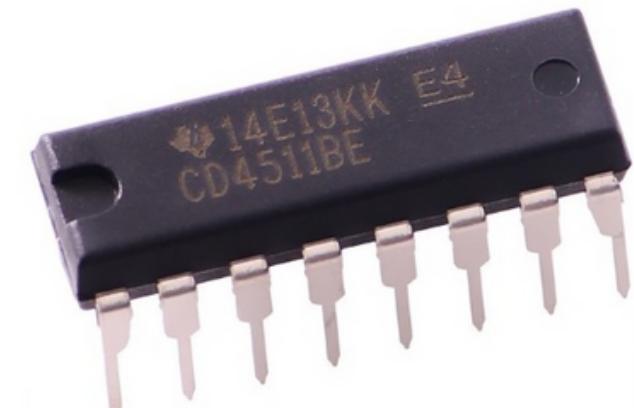
(c)

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

(d)

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

(e)

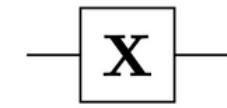


Fonte: [http://www.dpi.inpe.br/~carlos/Academicos/Cursos/ArqComp/aula\\_5bn1.html](http://www.dpi.inpe.br/~carlos/Academicos/Cursos/ArqComp/aula_5bn1.html)

# Portas Quânticas

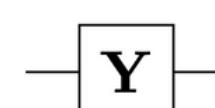
Pauli-X

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$



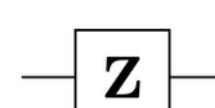
Pauli-Y

$$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$$



Pauli-Z

$$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$



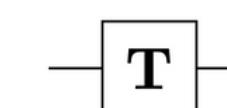
Hadamard

$$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$



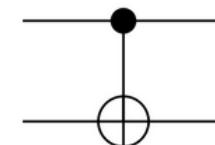
Pi/8

$$\begin{pmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{4}} \end{pmatrix}$$



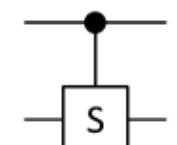
Controlled-Not

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$



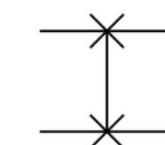
Controlled-Phase

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & i \end{pmatrix}$$



Swap

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



# Aplicando as Portas

$|0\rangle$

$|1\rangle$

$\begin{pmatrix} 1 \\ 0 \end{pmatrix}$

$\begin{pmatrix} 0 \\ 1 \end{pmatrix}$

# Aplicando as Portas

$$|0\rangle \quad |1\rangle \quad -\boxed{\mathbf{X}}-\quad \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} =$$
$$\begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

# Aplicando as Portas

$$\begin{matrix} |0\rangle & |1\rangle \\ \left( \begin{matrix} 1 \\ 0 \end{matrix} \right) & \left( \begin{matrix} 0 \\ 1 \end{matrix} \right) \end{matrix}$$

$$\xrightarrow{\text{---} \boxed{\mathbf{X}} \text{---}} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

# Aplicando as Portas

$$\begin{array}{c} |0\rangle \quad |1\rangle \\ \left( \begin{array}{c} 1 \\ 0 \end{array} \right) \quad \left( \begin{array}{c} 0 \\ 1 \end{array} \right) \end{array} \xrightarrow{\text{---} \boxed{\mathbf{X}} \text{---}} \begin{array}{c} \left( \begin{array}{cc} 0 & 1 \\ 1 & 0 \end{array} \right) \left( \begin{array}{c} 1 \\ 0 \end{array} \right) = \left( \begin{array}{c} 0 \\ 1 \end{array} \right) \\ \left( \begin{array}{cc} 0 & 1 \\ 1 & 0 \end{array} \right) \left( \begin{array}{c} 0 \\ 1 \end{array} \right) = \end{array}$$

# Aplicando as Portas

$$\begin{array}{cc} |0\rangle & |1\rangle \\ \left(\begin{matrix} 1 \\ 0 \end{matrix}\right) & \left(\begin{matrix} 0 \\ 1 \end{matrix}\right) \end{array} \xrightarrow{\text{---} \boxed{\mathbf{X}} \text{---}} \begin{array}{c} \left(\begin{matrix} 0 & 1 \\ 1 & 0 \end{matrix}\right) \left(\begin{matrix} 1 \\ 0 \end{matrix}\right) = \left(\begin{matrix} 0 \\ 1 \end{matrix}\right) \\ \left(\begin{matrix} 0 & 1 \\ 1 & 0 \end{matrix}\right) \left(\begin{matrix} 0 \\ 1 \end{matrix}\right) = \left(\begin{matrix} 1 \\ 0 \end{matrix}\right) \end{array}$$

# Aplicando as Portas

$$\begin{array}{c} |0\rangle \quad |1\rangle \\ \left(\begin{matrix} 1 \\ 0 \end{matrix}\right) \quad \left(\begin{matrix} 0 \\ 1 \end{matrix}\right) \end{array} \xrightarrow{\text{---} \boxed{\mathbf{X}} \text{---}} \begin{array}{l} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\ \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \end{array}$$

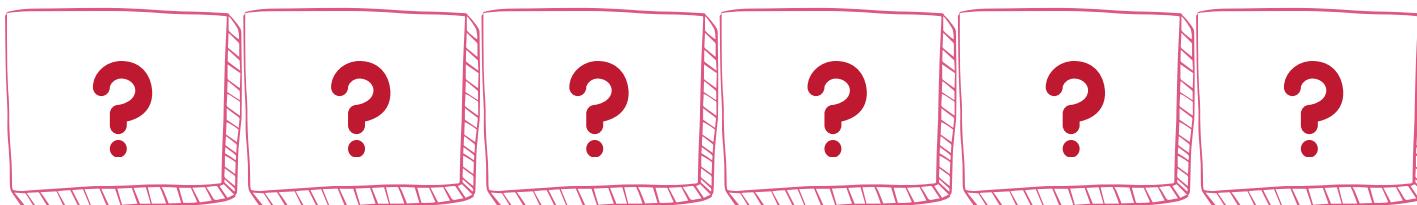
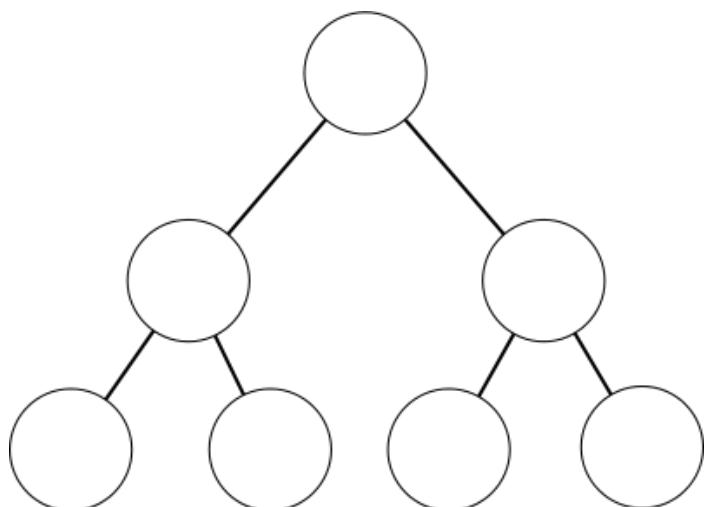
A porta Pauli-X funciona como NOT

# *Algoritmos Quânticos*



# Algoritmo de Grover

- Desenvolvido por Lov Grover em 1996
- Algoritmo de busca em bancos desordenados
- Comparação entre os algoritmos clássicos e o de Grover
- Probabilístico com custo de  $O(\sqrt{N})$

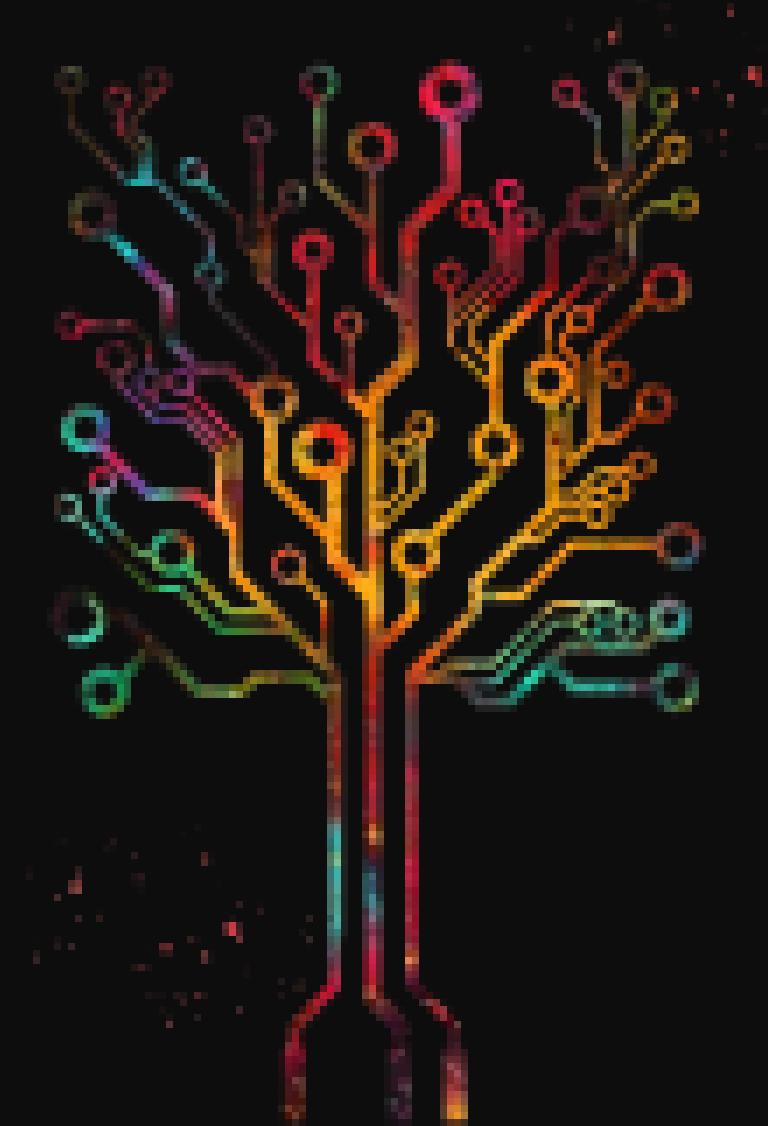


# Algoritmo de Shor

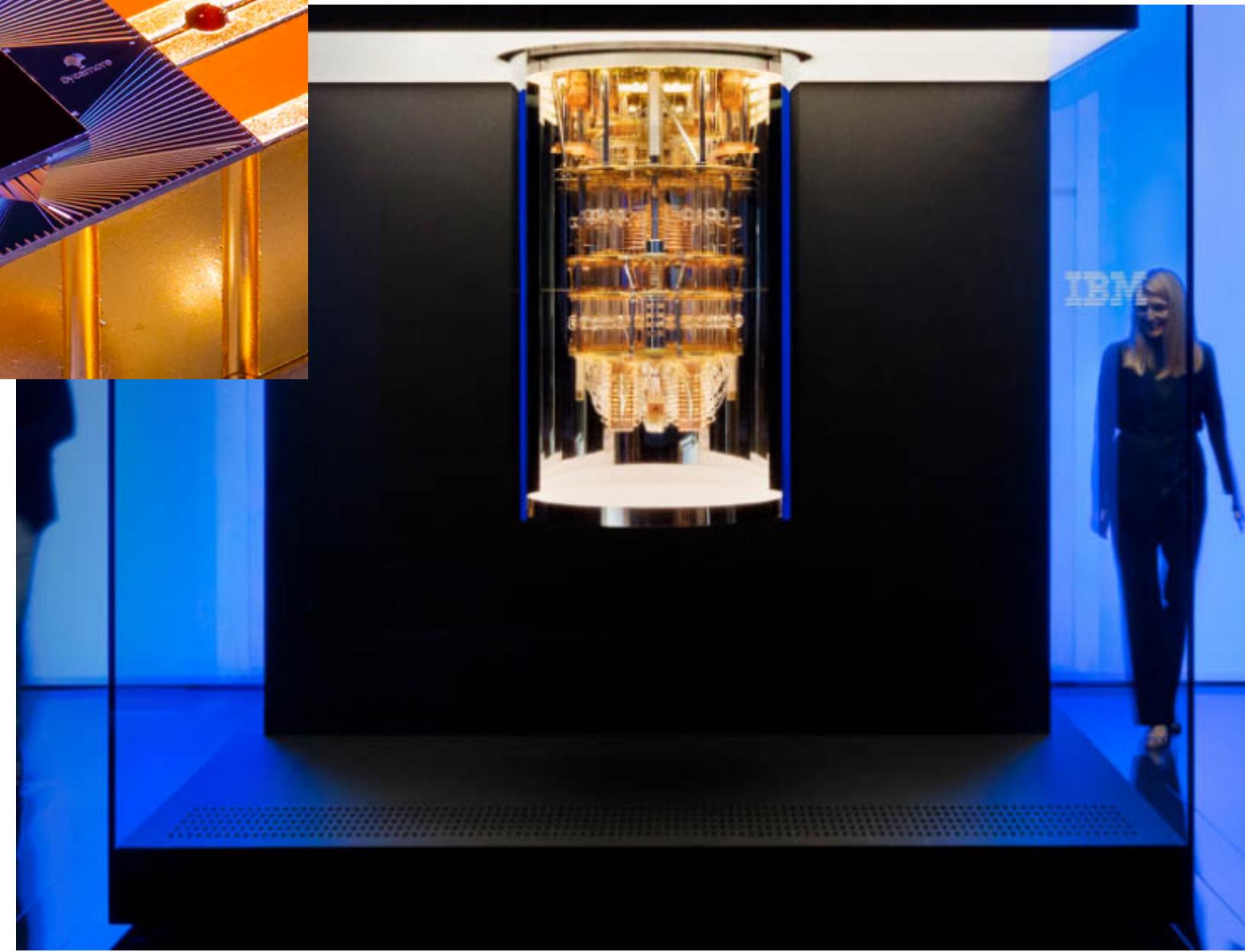
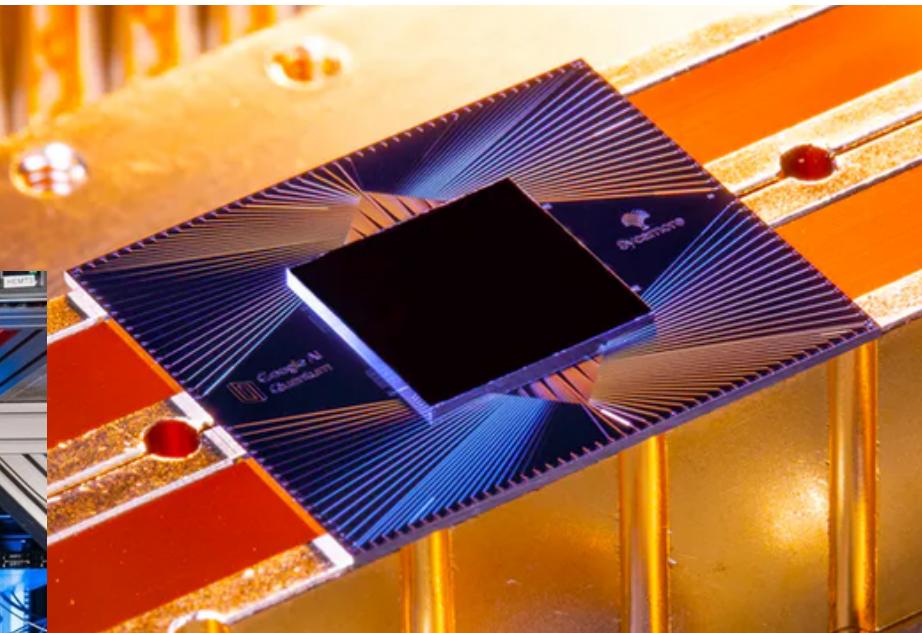
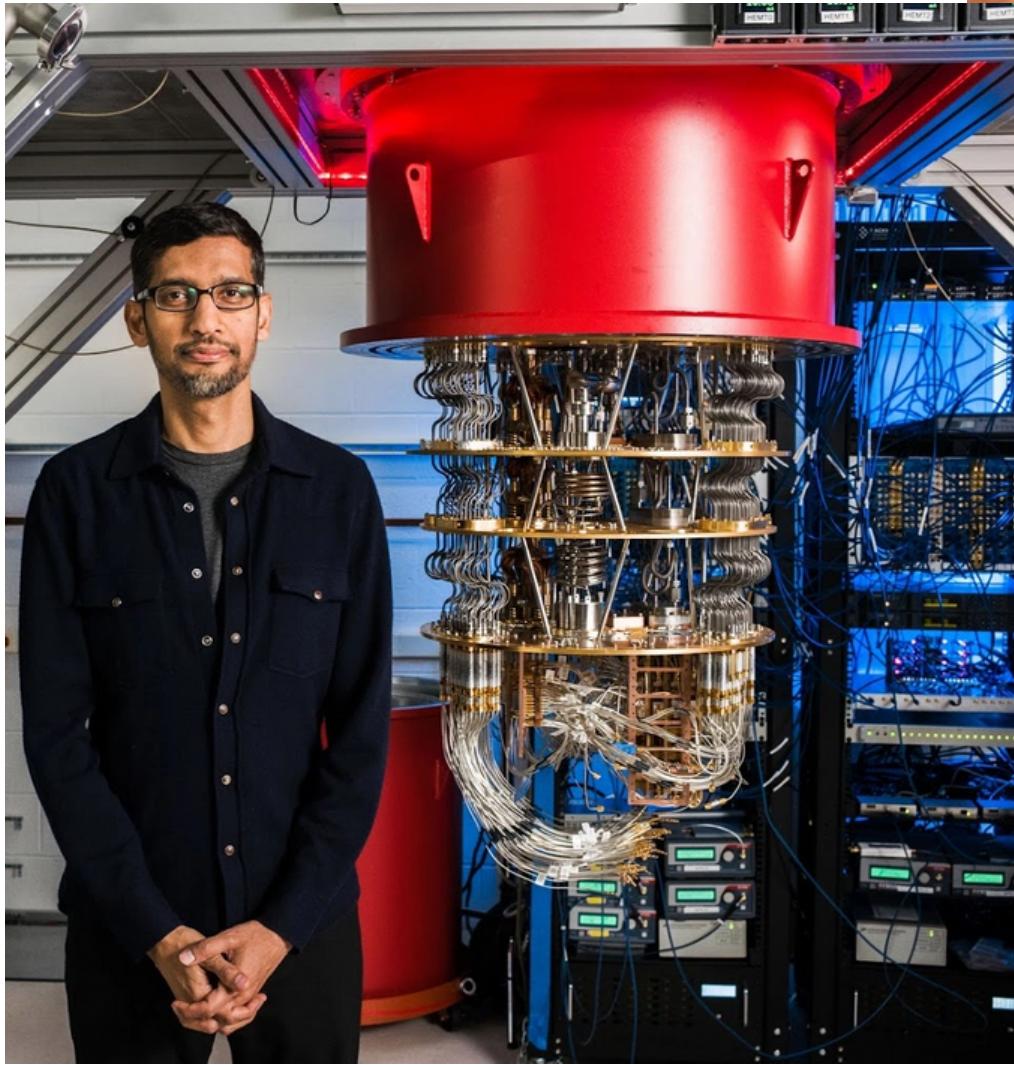
- Desenvolvido por Peter Shor em 1996
- Algoritmo de fatoração em números primos
- Quebra da criptografia atual
- Computadores atuais ainda não o executam



# A Computação Quântica Atualmente

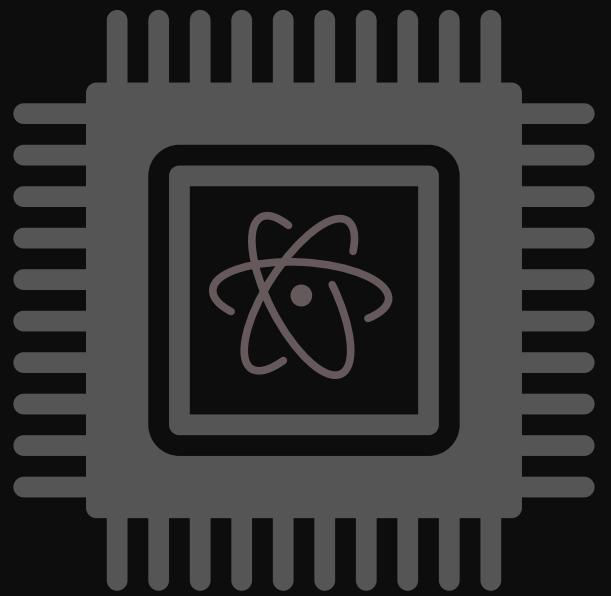


# Computadores Quanticos Atuais



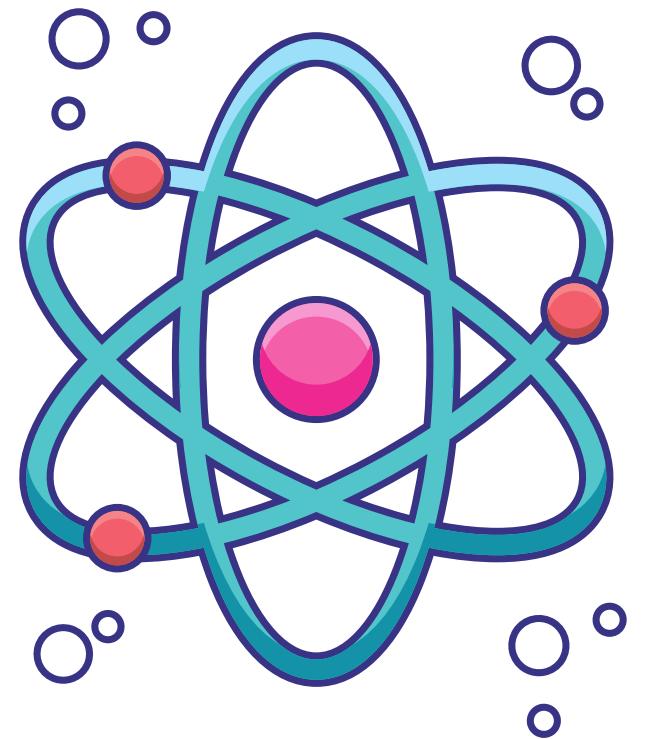
# *Mão na massa*

Simuladores e Bibliotecas



# Plataforma Quirk

- Disponível online
- Desenvolvido em JavaScript
- Conta com circuitos pré-programados
- A programação é feita "arrastando blocos"
- Código-fonte disponível no GitHub (Open Source)
- Tutorial disponível no YouTube



# Plataforma Quirk

Version 2.3

**Toolbox**

Probes	Displays	Half Turns	Quarter Turns	Eighth Turns	Spinning	Formulaic	Parametrized	Sampling	Parity
	Density Bloch	Z Swap	S S <sup>-1</sup>	T T <sup>-1</sup>	Z <sup>t</sup> Z <sup>-t</sup>	Z <sup>f(t)</sup> Rz(f(t))	Z <sup>A/2^n</sup> Z <sup>-A/2^n</sup>	Z  0> Z 0>	[Z] par
0><0   1><1	Chance Amps	Y	Y <sup>1/2</sup> Y <sup>-1/2</sup>	Y <sup>1/4</sup> Y <sup>-1/4</sup>	Y <sup>t</sup> Y <sup>-t</sup>	Y <sup>f(t)</sup> Ry(f(t))	Y <sup>A/2^n</sup> Y <sup>-A/2^n</sup>	Y  0> Y 0>	[Y] par
O ●		H	X <sup>1/2</sup> X <sup>-1/2</sup>	X <sup>1/4</sup> X <sup>-1/4</sup>	X <sup>t</sup> X <sup>-t</sup>	X <sup>f(t)</sup> Rx(f(t))	X <sup>A/2^n</sup> X <sup>-A/2^n</sup>	X  0> X 0>	[X] par

Circuit Diagram:

Local wire states (Chance/Bloch)      Final amplitudes

**Toolbox<sub>2</sub>**

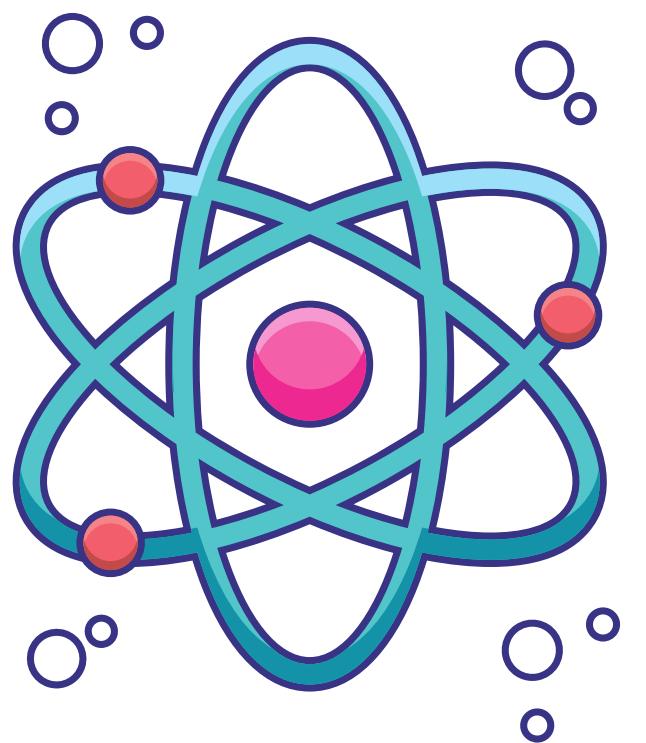
$\ominus$	$\oplus$	$+[t]$	$-[t]$	QFT	$QFT^\dagger$	input A	$A = \#$ default	+1	-1	$\oplus A < B$	$\oplus A > B$	+1 mod R	-1 mod R	...	0	Oracle
$\otimes$	$\otimes$	Reverse				input B	$B = \#$ default	+A	-A	$\oplus A \leq B$	$\oplus A \geq B$	+A mod R	-A mod R	-		
$ +>( + )$	$ -\rangle\langle - $			Grad <sup>1/2</sup>	Grad <sup>-1/2</sup>	input R	$R = \#$ default	+AB	-AB	$\oplus A = B$	$\oplus A \neq B$	$\times A$ mod R	$\times A^{-1}$ mod R	i	-i	

**Toolbox<sub>2</sub>**

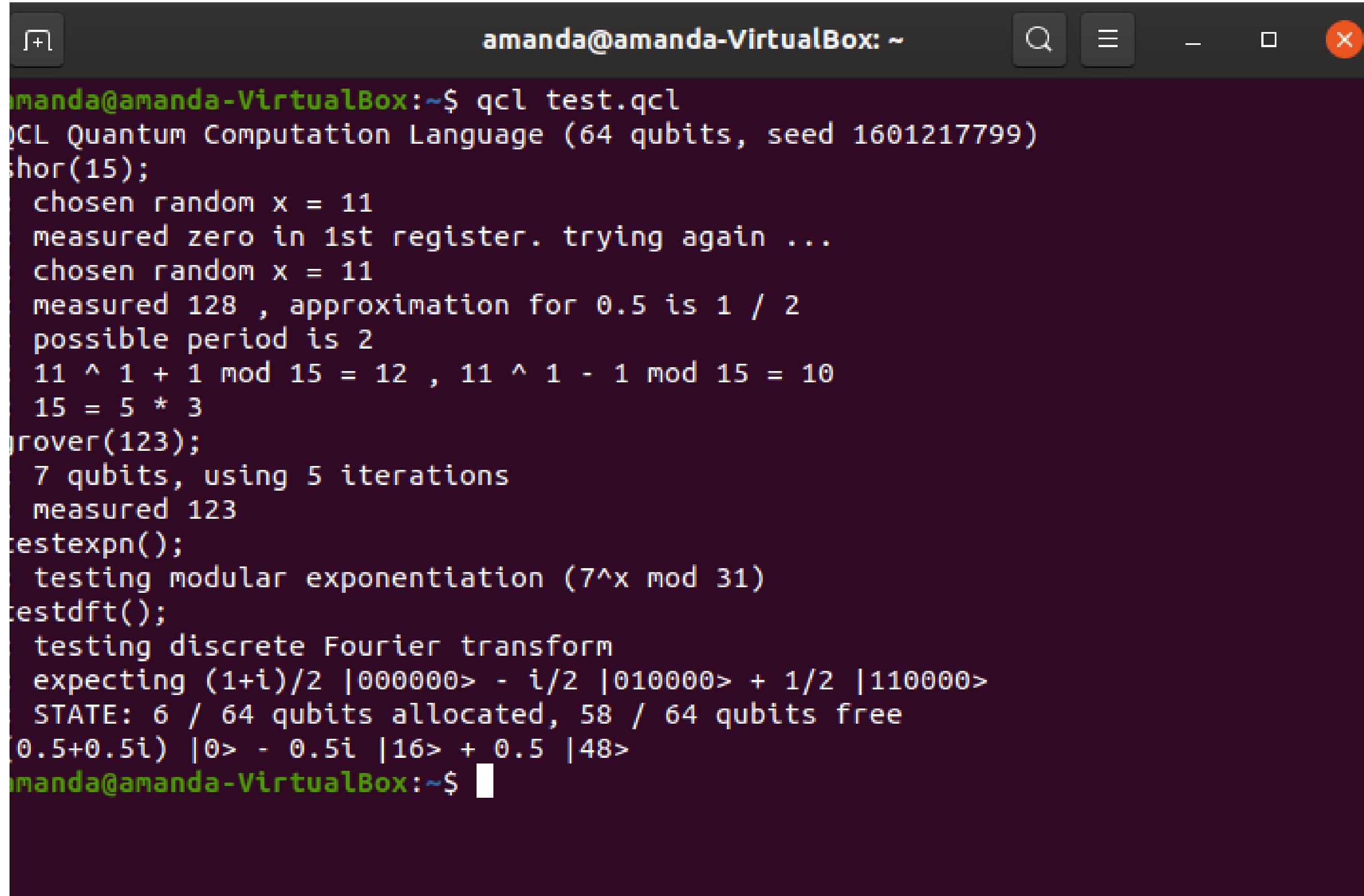
$\ominus$	$\oplus$	$+[t]$	$-[t]$	QFT	$QFT^\dagger$	input A	$A = \#$ default	+1	-1	$\oplus A < B$	$\oplus A > B$	+1 mod R	-1 mod R	...	0	Oracle
$\otimes$	$\otimes$	Reverse				input B	$B = \#$ default	+A	-A	$\oplus A \leq B$	$\oplus A \geq B$	+A mod R	-A mod R	-		
$ +>( + )$	$ -\rangle\langle - $			Grad <sup>1/2</sup>	Grad <sup>-1/2</sup>	input R	$R = \#$ default	+AB	-AB	$\oplus A = B$	$\oplus A \neq B$	$\times A$ mod R	$\times A^{-1}$ mod R	i	-i	

# Simulador QCL

- Open Source
- Desenvolvido em C para sistemas Linux
- Operações e algoritmos pré-programados
- Programação utilizando a linguagem QCL
- Open Source
- Documentação no site do desenvolvedor



# Simulador QCL

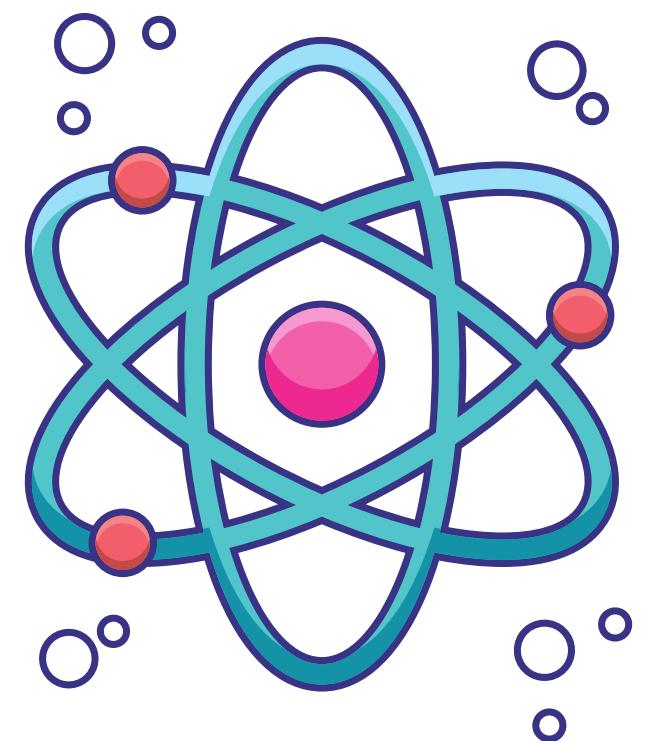


The screenshot shows a terminal window with a dark background and light-colored text. The title bar reads "amanda@amanda-VirtualBox: ~". The command entered is "qcl test.qcl". The output displays the execution of a QCL script named "test.qcl". The script includes several quantum operations like "shor(15)", "grover(123)", and "testexpn()", along with modular exponentiation and discrete Fourier transform tests. It also shows the current state of the qubits.

```
amanda@amanda-VirtualBox:~$ qcl test.qcl
QCL Quantum Computation Language (64 qubits, seed 1601217799)
shor(15);
    chosen random x = 11
    measured zero in 1st register. trying again ...
    chosen random x = 11
    measured 128 , approximation for 0.5 is 1 / 2
    possible period is 2
    11 ^ 1 + 1 mod 15 = 12 , 11 ^ 1 - 1 mod 15 = 10
    15 = 5 * 3
grover(123);
    7 qubits, using 5 iterations
    measured 123
testexpn();
    testing modular exponentiation (7^x mod 31)
testdft();
    testing discrete Fourier transform
    expecting (1+i)/2 |000000> - i/2 |010000> + 1/2 |110000>
    STATE: 6 / 64 qubits allocated, 58 / 64 qubits free
    |0.5+0.5i) |0> - 0.5i |16> + 0.5 |48>
amanda@amanda-VirtualBox:~$
```

# Microsoft Q# e QDK

- Software Livre
- Conjunto de bibliotecas numéricas, para machine learning e química
- É compatível com Python, C# e VB.NET
- Jupyter, Visual Code e Visual Studio



# Microsoft Q# e QDK

The screenshot shows the Microsoft Quantum Development Kit for Visual Studio Code extension page. The top navigation bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The title bar says "Extension: Microsoft Quantum Development Kit for Visual Studio Code - Visual Studio Code". The main content area displays the extension details: "Microsoft Quantum Development Kit for Visual Studio Code" by Microsoft DevLabs, version 31.261, with a 5-star rating. It provides support for developing quantum algorithms in the Q# programming language. Buttons for Disable and Uninstall are present, along with a note that the extension is enabled globally. Below this, a section titled "Microsoft Quantum Development Kit Preview" contains text about the kit's purpose, a note about the simulator requiring a 64-bit system, and a link to the GitHub repository. A "Feedback" section encourages users to provide feedback via GitHub. A "Components and Documentation" section is also mentioned. The sidebar on the left contains icons for file operations like Open, Save, Find, and others.

File Edit Selection View Go Run Terminal Help

Extension: Microsoft Quantum Development Kit for Visual Studio Code - Visual Studio Code

Extension: Microsoft Quantum Development Kit for Visual Studio Code X

**Microsoft Quantum Development Kit for Visual Studio Code** quantum.quantum-devkit-vscode

Microsoft DevLabs | 31.261 | ★★★★★ | Repository | v0.12.20082513

Microsoft Quantum Development Kit for Visual Studio Code provides support for developing quantum algorithms in the Q# programming language.

Disable Uninstall This extension is enabled globally.

Details Feature Contributions

## Microsoft Quantum Development Kit Preview

Thank you for your interest in Microsoft's Quantum Development Kit for Visual Studio Code preview. The Quantum Development Kit contains the tools you'll need to build your own quantum computing programs and experiments. Assuming some experience with Visual Studio Code, beginners can write their first quantum program, and experienced researchers can quickly and efficiently develop new quantum algorithms.

To jump right in, take a look at our [Getting Started guide](#). Use [Quickstart](#) - your first computer program to learn about the structure of a Q# project and how to write the quantum equivalent of "Hello, world!" -- a quantum teleport application.

**NOTE: The simulator included with the Quantum Development Kit requires a 64-bit operating system to run.**

The source code for this extension can be found on [our GitHub repository](#).

### Feedback

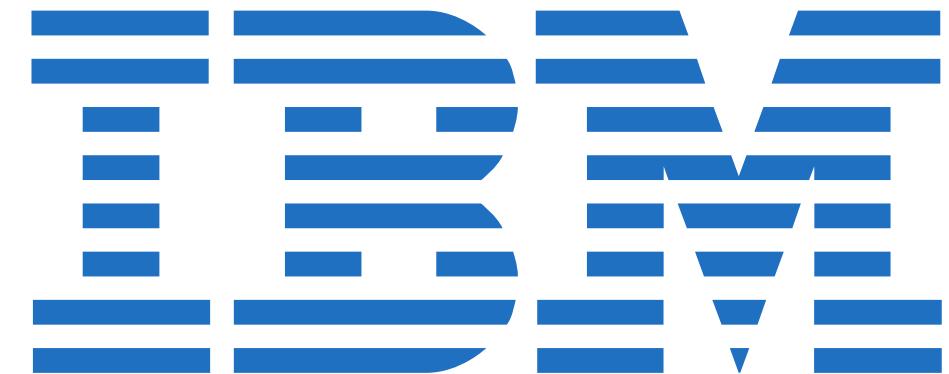
Your feedback about all parts of the Quantum Development Kit is important. Please go to [our GitHub repository](#) to provide feedback on the Q# compiler and language extensions, or to learn more about where to give feedback on other parts of the Quantum Development Kit.

### Components and Documentation

The Quantum Development Kit preview provides a complete development and simulation environment. We refer to [our documentation](#) for further information.

# IBM QisKit

- Programa Open Source
- Pode ser utilizado localmente ou online
- Ferramenta gráfica para programar circuitos quânticos
- Utiliza Python
- Usando a Quantum Experience, é possível executar os circuitos em computadores quânticos da IBM\*\*



# Cirq

- Open Source
- Utilizado para pequenos circuitos
- Biblioteca de Python
- Suporte para programar computadores quânticos da Google

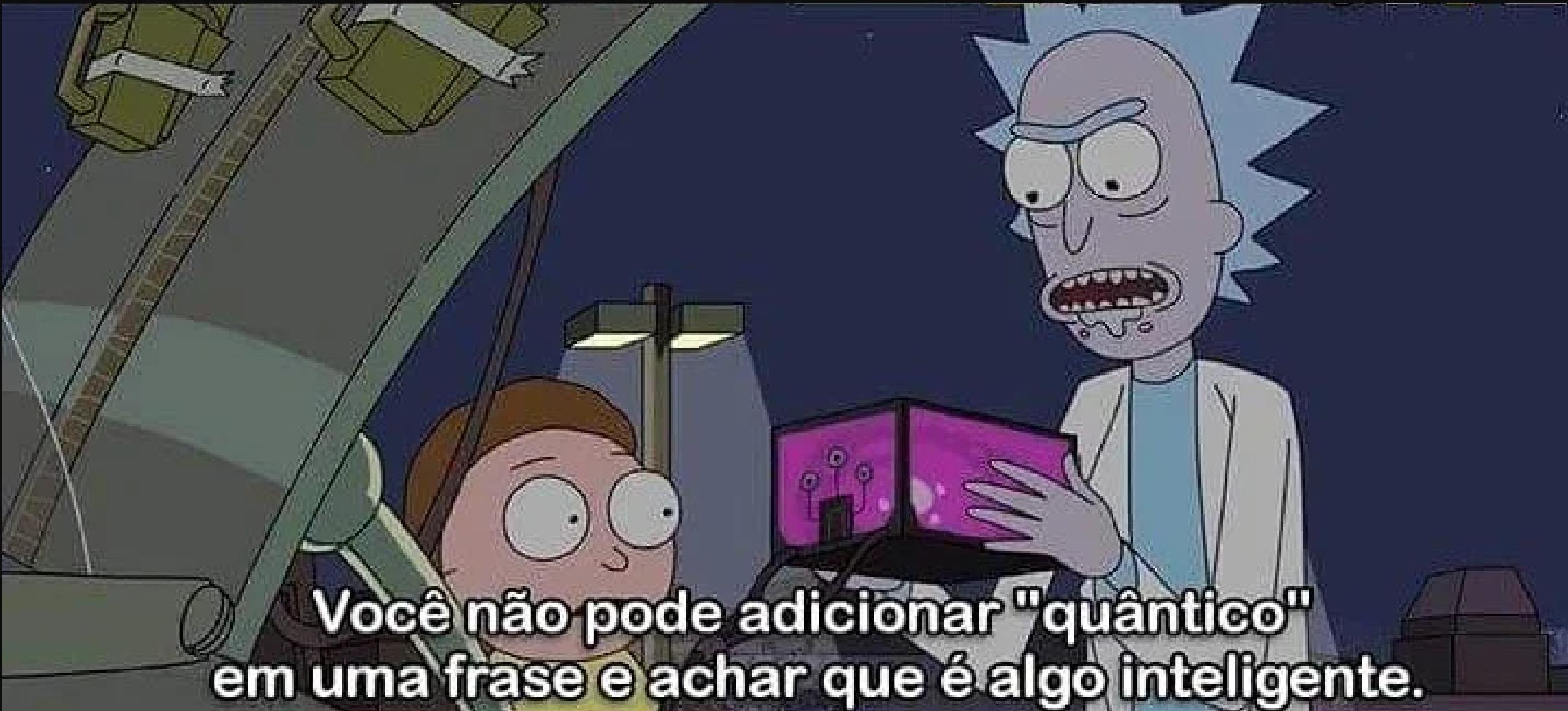


# E muitas outras opções...



Você pode acessar a lista de simuladores e bibliotecas em:

<https://www.quantiki.org/wiki/list-qc-simulators>



**Você não pode adicionar "quântico"  
em uma frase e achar que é algo inteligente.**

Obs. final: a quântica não vai melhorar a sua vida, ela só afeta partículas e objetos microscópios. Desconfie de tudo que você ouvir com este termo, verifique as fontes e sempre estude.

THANK  
YOU

DÚVIDAS?

amanda.valerio@alunos.if sulde minas.edu.br