Più Qubit per tutti!

Dal Quantum Computing al linguaggio Q#

#CodeGen

#dotnetconf

@cloudgen_verona

Main sponsor

Community sponsors

Syncfusion®

Packt

MANNING

elmah.io

Udemy™

JET BRAINS

Local Event


It's FREE!
.NET Conf
Discover the world of .NET
September 12-14, 2018
Tune in: www.dotnetconf.net
.NET

Chi sono

Roberto Albano @dancerjude

- Dev (+ varie cose a contorno)
- Fondatore di dotNET{podcast}
- Senior Technical Analyst @ KPMG
- Consumatore di grappa barricata

  (rigorosamente dopo un buon pasto)

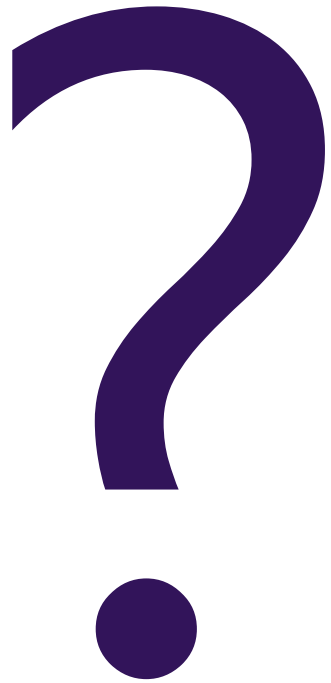
Agenda

- Cos'è il Quantum Computing
- Cos'è un Qubit
- Microsoft Quantum Development Kit
- Il linguaggio Q#

Disclaimer!

- Non sono un fisico
- Non sono un matematico
- Vi spiegherò quello che ho capito
- Ve lo spiegherò come l'ho capito
- Non sparate sul pianista

Cos'è il Quantum Computing



Cos'è il Quantum Computing

Classical Computing

- Elaborazione
- Algoritmi «classici»

Quantum Computing

- Elaborazione simultanea
- Algoritmi «quantistici»

Cos'è il Quantum Computing

Classical Computing

- elaborazione sequenziale
- basati sui bit (binary digit, 0 / 1)
- codice binario
- i computer sono «sistemi binari»
- le informazioni sono organizzate in «registri»
  (32/64 bit in base all'architettura)
- logica booleana (NOT, AND, OR, NAND, NOR, XOR…)

Cos'è il Quantum Computing

Quantum Computing

- elaborazione simultanea
- basati sui qubit (quantum bit, 0 / 1 / 0&1)
- codice binario
- i computer quantistici sono «sistemi binari»
- le informazioni possono organizzate in «registri»
- logica booleana (NOT, AND, OR, NAND, NOR, XOR…)

Pro e contro del Quantum Computing

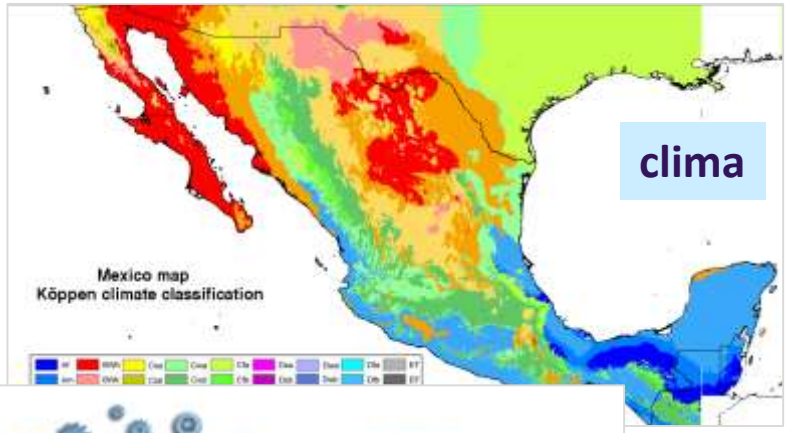
- capacità di calcolo esponenzialmente superiori
- «dramatically improvement» nell'esecuzione di algoritmi «classici» in ambiente «quantico»
- tutti gli ambiti possono essere impattati positivamente (economia, finanza, medicina, ricerca…)
- gli algoritmi di cifratura considerati oggi «a prova di hacker» saranno tutti seriamente a rischio

Quali sono i campi di applicazione

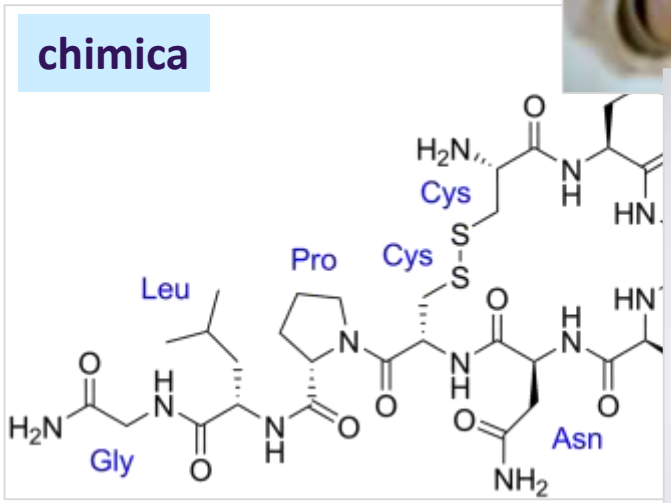
AI

salute

clima



chimica



materiali

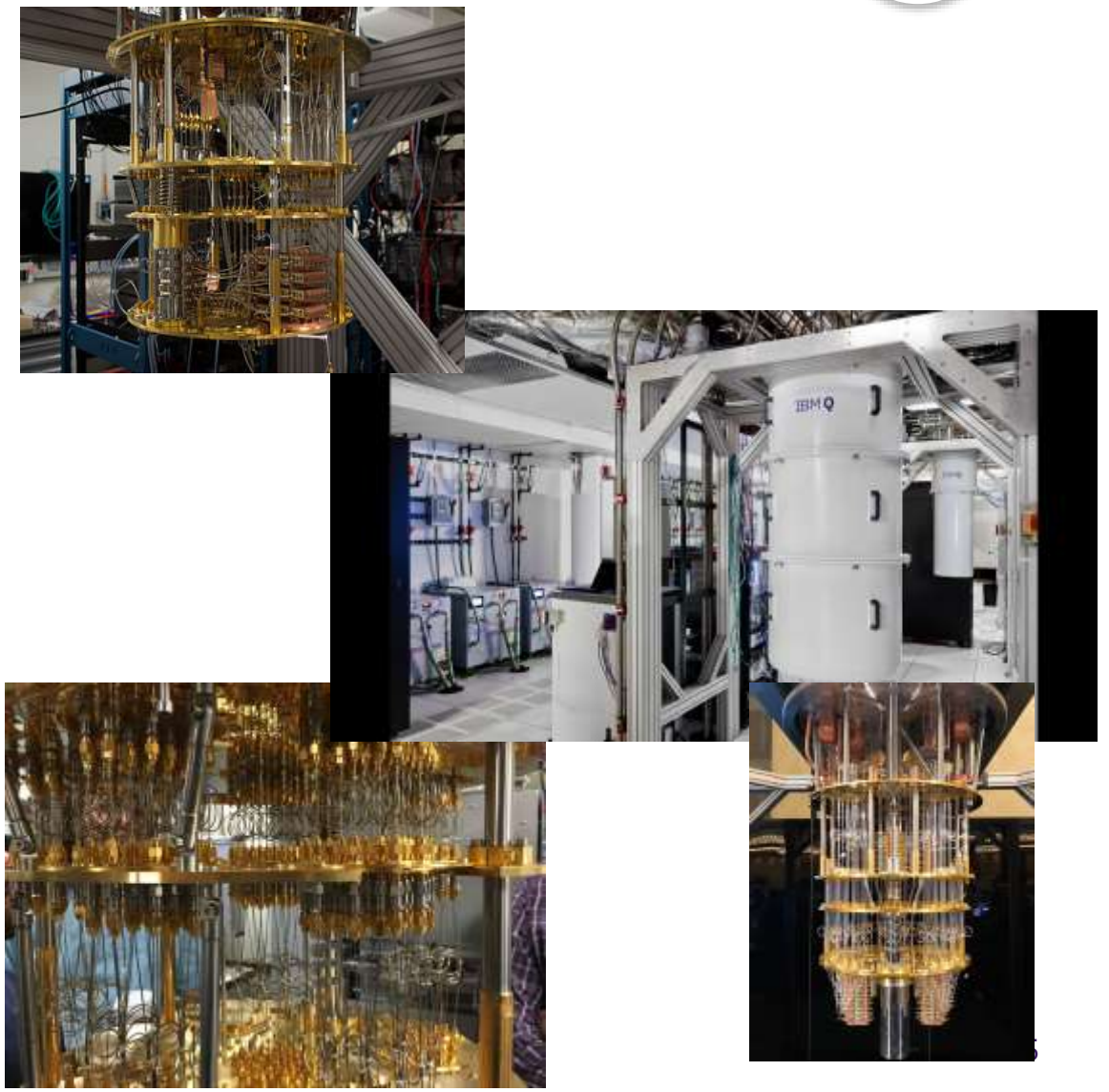
machine learning

... e tanti, tanti altri...

Cos'è il Quantum Computing

Come sono fatti i computer quantistici

- assomigliano a piccole centrali nucleari
- per il corretto funzionamento devono operare a temperature bassissime
- si tratta di una tecnologia ancora in via di sviluppo
- lo scopo futuro è realizzare il «backend» dei nostri servizi su computer quantistici



Cos'è il Quantum Computing

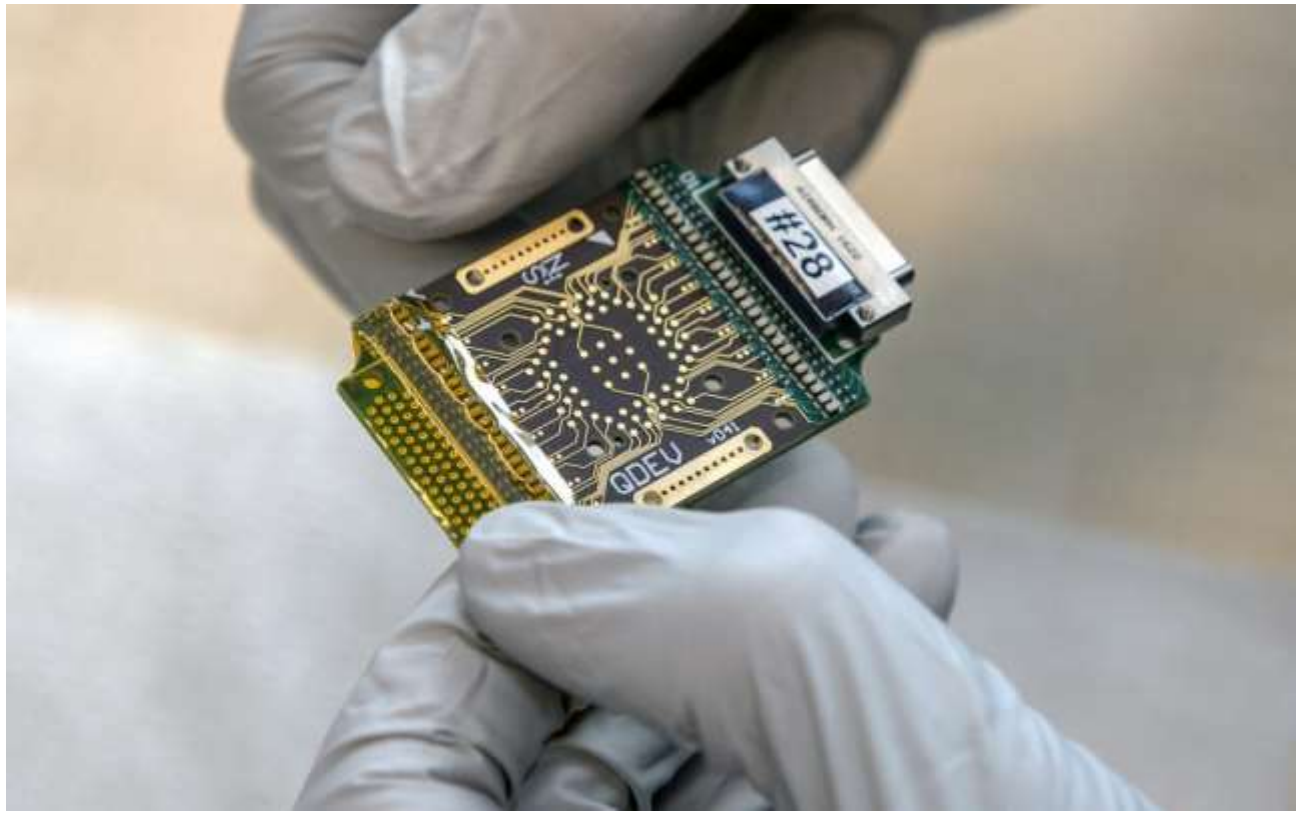
Chi ha costruito (o sta costruendo) un computer quantistico

- IBM => 20 qubit (50 qubit in costruzione)
- Google => 50 qubit (in costruzione)
- Intel => 49 qubit (annunciato a CES2018)
- NASA => 1097 qubit (avete capito bene)
- Microsoft => ?

(si tratta comunque in alcuni casi di tecnologie differenti e lo standard è ancora in fase di definizione)

Cos'è il Quantum Computing

Microsoft Topological Qubit (https://cloudblogs.microsoft.com/quantum/)



Si tratta di una tecnologia nata con lo scopo di avere una sorta di «qubit scalabili»

Cos'è il Quantum Computing

Microsoft Topological Qubit (https://cloudblogs.microsoft.com/quantum/)

- Teoria e simulazione (prototipazione)
- Fabbricazione (tecniche pioneristiche di costruzione)
- Miglioramento dei materiali (sviluppo)
- Misurazione (accuratezza e precisione)

Cos'è un Qubit

quantum bit => qubit

bit => 0 o 1

qubit => 0 o 1 o (0 e 1)

stato «fondamentale» (0)
stato «eccitato» (1)

quindi si tratta ancora di un sistema binario…
…o forse no…

Cos'è un Qubit

- il concetto di qubit si basa sulla meccanica quantistica
- può presentare una *«sovrapposizione coerente di stati»*
- quindi può essere nello stesso momento sia 0 che 1
- un qubit può essere rappresentato anche attraverso l'uso di un solo atomo

Cos'è un Qubit

bit vs qubit

registro a 3 bit =>

1)	valore 0	0|0|0
2)	valore 1	0|0|1
3)	valore 2	0|1|0
4)	valore 3	0|1|1
5)	valore 4	1|0|0
6)	valore 5	1|0|1
7)	valore 6	1|1|0
8)	valore 7	1|1|1

il registro a 3 bit può contenere uno ed uno solo degli 8 valori possibili

bit vs qubit

registro a 3 qubit =>

1)	valore 0	0|0|0
2)	valore 1	0|0|1
3)	valore 2	0|1|0
4)	valore 3	0|1|1
5)	valore 4	1|0|0
6)	valore 5	1|0|1
7)	valore 6	1|1|0
8)	valore 7	1|1|1

il registro a 3 qubit può contenere anche tutti gli 8 valori possibili contemporaneamente

Cos'è un Qubit

Fenomeni della meccanica quantistica:

- sovrapposizione coerente di stati (*superposition*)
- interferenza quantistica
- correlazione quantistica (*entanglement*)
- principio di indeterminazione di Heisenberg

Cos'è un Qubit

Superposition

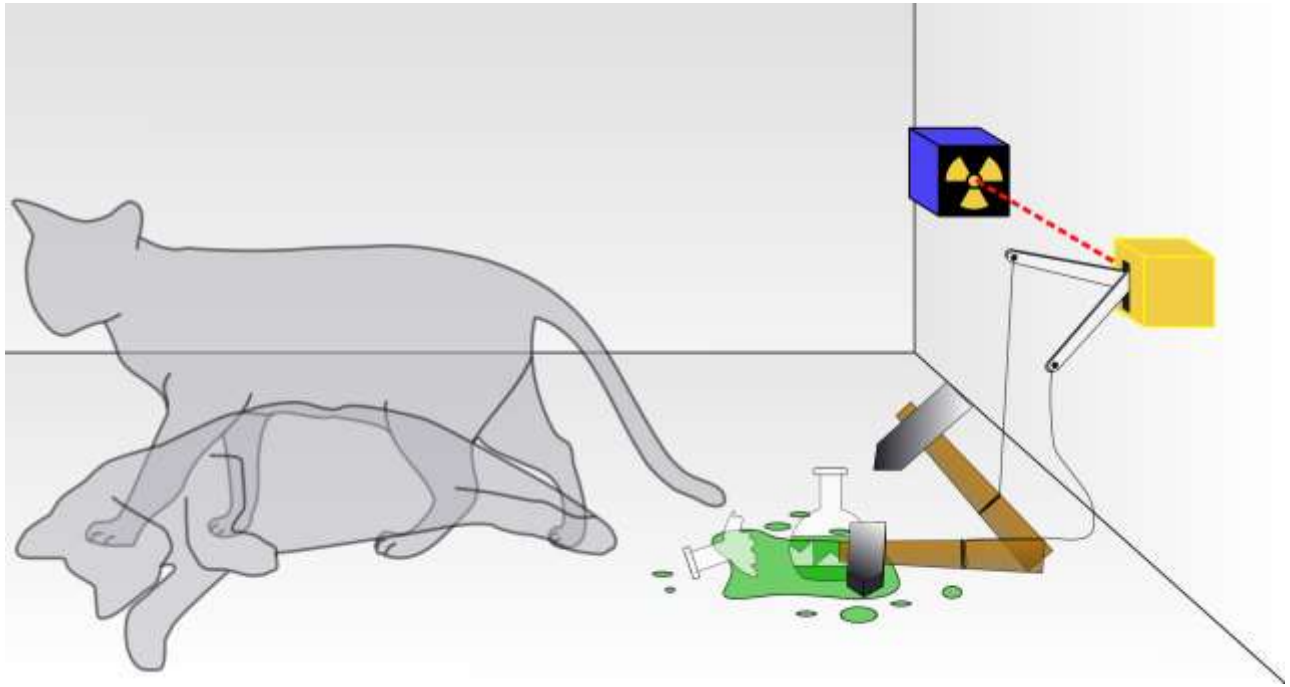
- non è facile «immaginare» questa situazione
- si tratta di una condizione «estranea» alla nostra capacità cognitiva
- si può trovare «contemporaneamente» negli stati quantistici |0> e |1> «con pesi diversi»
- non si tratta di un ulteriore «stato 2»

$$|\Psi> = \alpha|0> + \beta|1>$$

(dove α e β sono coefficienti complessi)

Paradosso del gatto di Schrödinger

- «trasferire» la sovrapposizione di stati ad un oggetto macroscopico può potenzialmente portare ad un paradosso
- il gatto si trova rinchiuso in una scatola dove vi è una condizione «parimenti probabile» che possa essere avvelenato
- *La funzione Ψ dell'intero sistema porta ad affermare che in essa il gatto vivo e il gatto morto non sono degli stati puri, ma miscelati con uguale peso*



Cos'è un Qubit

Entanglement

- fenomeno quantistico senza corrispondente «classico»
- due qubit o più qubit possono stabilire una relazione «entangled» (record cinese: 18 qubit «entangled»)
- la misurazione di un qubit entangled influenza la misurazione del qubit con cui si relaziona
- la relazione è (teoricamente) stabilita senza limiti di distanza
- quindi i qubit «entangled» si influenzano a vicenda

Cos'è un Qubit

Superposition, Entanglement…

«Cit.»

«If you think you understand quantum mechanics, you don't understand quantum mechanics.»

Richard Feynman

Premio Nobel per la fisica nel 1965
per l'elaborazione dell'elettrodinamica quantistica.

Microsoft Quantum Development Kit

Da dove partire?

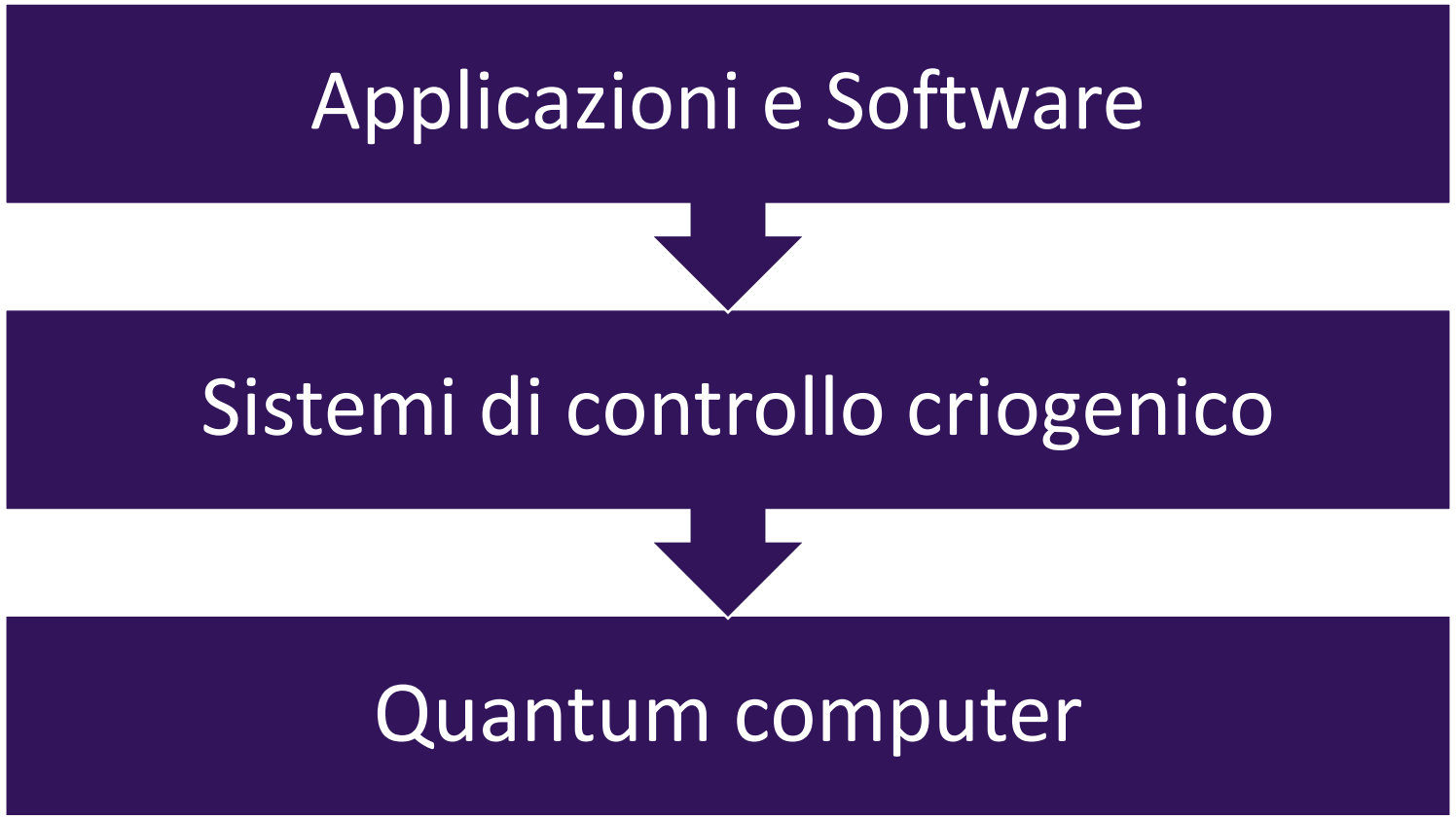
microsoft.com/quantum

docs.microsoft.com/quantum (documentation)

github.com/Microsoft/Quantum (QDK samples and libraries)

- Windows 10
  - Visual Studio 2017
- Windows + Linux + macOS
  - Visual Studio Code
  - Command Line

The «Big Picture»: un sistema completo e scalabile



Applicazioni e Software

Q# Code

Quantum Algotithm Libraries

Classical Host Program

Classical Runtime

Cryo-Control

Local Simulator

Azure Simulator

Quantum Computer

Installazione in Visual Studio 2017

Visual Studio Marketplace
https://marketplace.visualstudio.com/items?itemName=quantum.DevKit

Installazione in Visual Studio Code e/o Command Line

- installare il **.NET Core SDK 2.0** o versione successive
  https://www.microsoft.com/net/download

- installare i **template aggiornati** per la creazione di applicazioni o librerie Q#
  utilizzando da PowerShell o Bash il comando:

```
dotnet new -i "Microsoft.Quantum.ProjectTemplates::0.2.1809.701-preview"
```

- installare (o meno **Visual Studio Code**)
  https://code.visualstudio.com/

- clonare il *Microsoft Quantum Development Kit* da *GitHub*:

```
git clone https://github.com/Microsoft/Quantum.git
```

Cosa possiamo fare con il QDK?

- Creare esperimenti quantistici
- Far girare su computer «classici» operazioni quantistiche
- Agire come un «coprocessore» dedicato
- Il codice Q# gira «internamente» al codice C#

Livelli di utilizzo della computazione quantistica:

- La computazione classica usa la computazione quantistica internamente per le elaborazioni
- La computazione quantistica avviene su computer quantistici
- La computazione quantistica usa la computazione classica internamente per le elaborazioni

Specifiche linguaggio Q#: i tipi

Tipo	Descrizione
`Int`	intero a 64 bit
`Double`	doppia precisione
`Bool`	booleano
`Qubit`	qubit
`String`	stringa
`Pauli` `(+ PauliI, PauliX, PauliY, PauliZ)`	matrice di Pauli
`Result`	risultato della misurazione, valore `One` o `Zero`
`String`	stringa
`Range`	sequenza di interi

Specifiche linguaggio Q#: gli array

Tipo	Descrizione
`'T[]`	Array di oggetti di tipo `'T`
`'T[][]`	Array di array di oggetti di tipo `'T`

- In Q# gli array non sono supportate matrici «rettangolari»

Specifiche linguaggio Q#: le tuple

Tipo	Descrizione
`('T1,'T2,'T3,…)`	Tupla di oggetti di tipo `'T1,'T2,'T3,…`
`()`	Tupla «vuota»

- In Q# si possono creare array di tuple, ma anche tuple di array
- Sono ammesse anche tuple di tuple
- Una volta creata una tupla, il suo contenuto non può essere più variato (*«immutabilità»*)

Specifiche linguaggio Q#: *singleton tuple equivalence*

- Una tupla con un singolo elemento viene chiamata *singleton*
- Il linguaggio Q# tratta la tupla come il valore corrispondente del suo tipo, ad esempio:
  - `(5) == 5`
  - `(2)+1 == 3`
  - `(8)+(1) == 9`
- Questa caratteristica viene detta *singleton tuple equivalence*
- Può essere usata nelle assegnazioni e nelle espressioni

Specifiche linguaggio Q#: *user-defined types*

- Si possono derivare nuovi tipi derivati da tipi esistenti o da tuple:
  - `newtype TypeA = (Int, TypeB);`
  - `newtype TypeB = (Double, TypeC);`
  - `newtype TypeC = (TypeA, Range);`
- La «mutabilità» di un tipo derivato dipende da quella del tipo base
- Non si possono creare strutture ricorsive

Specifiche linguaggio Q#: alcune porte logiche

Porta	Descrizione
`H`	applica la trasformazione di Hadamard ad un singolo qubit
`I`	esegue l'operazione di identity su un singolo qubit (no-op)
`M`	misura di un singolo qubit (utilizzando la matrice PauliZ)
`R`	applica una rotazione su un certo asse di Pauli
`X`	inverte lo stato di un qubit (not)
`CNOT`	controlled NOT (xor)

Specifiche linguaggio Q#: le *«callables»*

- Operations
  - sono subroutine quantistiche
  - possono contenere operazioni quantistiche
  - sono paragonabili ai metodi «statici» in C#
- Functions
  - sono subroutine classiche
  - non possono contenere operazioni quantistiche
  - possono essere utilizzate all'interno di algoritmi quantistici
  - non possono richiamare le Operations
  - non possono istanziare qubit, ma possono riceverli come parametro

Specifiche linguaggio Q#: le *«callables»*

- Le Operations e le Functions devono avere sempre un solo input e un solo output
- Possono usare come input e/o output le tuple, anche vuote
  - Operations `('Tinput => 'Tresult)`
  - Functions `('Tinput -> 'Tresult)`

Specifiche linguaggio Q#: *Functors*

- Si tratta di factory che definiscono una nuova Operation a partire da una esistente
- In Q# esistono due *Functor* standard:
  - `Adjoint`
  - `Controlled`
- Applicando un `Adjoint` all'operazione `Y`, si ottiene una nuova operazione `(Adjoint Y)`
- Un esempio: `(Qubit => () : Adjoint, Controlled)`

Specifiche linguaggio Q#: *Functors*

- Adjoint
  - può creare una nuova operation che corrisponde alla trasposizione complessa della operation base
- Controlled
  - può creare una nuova operation che consente l'applicazione della operation base solo se i qubit di controllo sono in uno specifico stato

Specifiche linguaggio Q#: le librerie standard

- Prelude
  - operazioni e funzioni definite in base al sistema di destinazione
  - sistemi diversi possono avere implementazioni specifiche
  - utilizza il codice nativo .NET
- Canon
  - operazioni e funzioni definite sulla base del Prelude
  - le implementazioni sono le stesse indipendentemente dalle macchine di destinazione

Quantum Simulator

- Esegue localmente una quantità ridotta di Qubit
- Esegue in cloud una versione con più Qubit
- Richiede grandi quantità di memoria
  - 32 qubits => 32GB di RAM
  - 33 qubits => 64GB di RAM
  - 34 qubits => 128GB di RAM
  - …
  - 40 qubits =>
- Simile al simulatore LIQ$Ui|\rangle$ sviluppato da *Microsoft Research* http://stationq.github.io/Liquid/

Demo
Q#

Link utili

- microsoft.com/quantum
- docs.microsoft.com/quantum
- cloudblogs.microsoft.com/quantum
- github.com/Microsoft/Quantum
- github.com/Microsoft/QuantumKatas
- tio.run/#qs-core

«Cit.»

«I computer sono incredibilmente veloci, accurati e stupidi.

Gli uomini sono incredibilmente lenti, inaccurati e intelligenti.

L'insieme dei due costituisce una forza incalcolabile.»

Albert Einstein

Grazie

...per non aver sparato sul pianista :-D