DOI: https://doi.org/10.26642/ten-2024-1(93)-240-245
УДК 004.43

Д.О. Сапожник, аспірант
Державний університет «Житомирська політехніка»

Аналіз можливостей мови програмування Q# шляхом реалізації програми для генерації випадкових чисел

(Представлено: Морозов А.В., к.т.н., доц.)

У статті проведено аналіз можливостей мови програмування Q# шляхом реалізації програми для генерації випадкового числа. Звертається увага на основні інструменти квантового програмування, проводиться аналіз напрямків інвестицій у квантові технології. Проводиться аналіз публікацій у сфері квантового програмування та наголошується увага на основних особливостей квантових бітів. Основна увага приділяється використанню гібридної парадигми програмування, яка поєднує класичні та квантові обчислення. Описано структуру та особливості квантової програми на Q#, зокрема операції GenerateRandomBit та GenerateRandomNumberInRange, які забезпечують генерацію випадкових бітів та чисел завдяки квантовій суперпозиції та вимірюванню. Висвітлено переваги Q# у поєднанні з класичними мовами програмування для ефективного використання квантових алгоритмів у криптографії та статистичному моделюванні. У висновках наголошено на перспективності мови програмування Q# як інструмента для розробки квантових програм у гібридному середовищі, що дозволяє використовувати потенціал квантових обчислень разом із класичними підходами.

Ключові слова: *мова програмування Q#; квантові обчислення; генерація випадкових чисел; кубіти; квантова суперпозиція; гібридна парадигма програмування; квантові алгоритми.*

Актуальність дослідження можливостей мови програмування Q# є важливим у контексті поточного розвитку квантових технологій, зокрема квантової криптографії, квантових алгоритмів. Десятки компаній та країн інвестують у квантові технології все більше ресурсів, серед них можна виокремити: Microsoft з сервісом Azure Quantum та мовою програмування Q# [1], IBM з власними квантовими комп'ютерами та бібліотекою qiskit для роботи з ними [2], Google також спеціалізується на дослідженнях в області квантового програмування [3]. Країни Європи, Азії, США та інші на одному рівні з суперкомп'ютерами мають квантові комп'ютери також. Квантові технології розвиваються, потужність та стійкість квантових комп'ютерів теж зростає з року в рік, це можна порівняти з минулим, двадцятим століттям, коли класичні комп'ютери тільки зароджувалися та розвивалися. Про активний розвиток квантових технологій свідчить і значне зростання інвестицій у цю сферу: у 2022 році інвестиції сягнули рекордних 2,5 мільяра доларів, за 2024 рік вже було проінвестовано 1,1 мільярда, діаграму можна побачити на рисунку 1 [4]. Таким чином гостро постає потреба у дослідженнях в області квантового програмування.

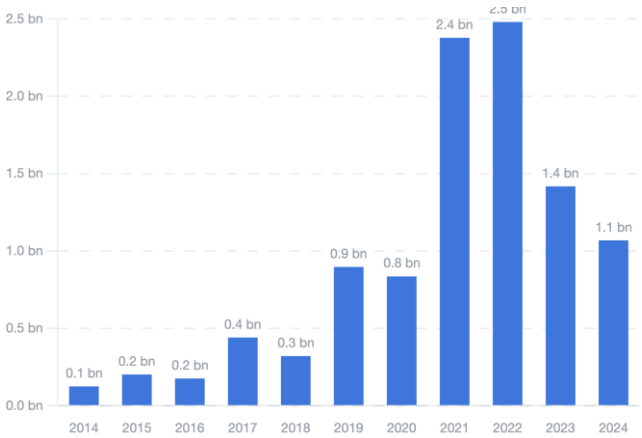

Рис. 1. Обсяги інвестицій у квантові технології [4]

Мета статті полягає у дослідженні особливостей квантового програмування, синтаксису та логічних структур мови програмування Q#, оцінки потенціалу квантових обчислень у контексті реалізації алгоритмів для генерації випадкових чисел.

Викладення основного матеріалу. Квантове програмування, зокрема мова Q#, є інноваційною галуззю, яка відкриває нові можливості у світі обчислень. Кубіти як основа квантових обчислень дозволяють вирішувати складні завдання завдяки властивостям суперпозиції, заплутаності та інтерференції. Щоб краще зрозуміти потенціал квантових обчислень і програмування на Q#, важливо розглянути відповідну літературу.

У навчальному посібнику «Вступ до квантового обчислення та квантової інформації» (Крохмальський В.Д., 2021) детально розглядаються теоретичні основи квантових обчислень та квантової інформації, що є базисом для розуміння квантового програмування. Автор описує базові принципи квантової механіки, які формують основу квантових обчислювальних систем. Суперпозиція, заплутаність та інтерференція є ключовими поняттями, які дозволяють квантовим комп'ютерам перевершувати класичні у вирішенні певних завдань[5].

Кубіти, як зазначено у роботі Крохмальського, відрізняються від класичних бітів тим, що вони здатні перебувати одночасно у станах $|0\rangle$ і $|1\rangle$ завдяки суперпозиції. Автор описує математичну модель кубіта у вигляді лінійної комбінації станів: $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, де α та β є комплексними амплітудами. Це дозволяє кубітам обробляти значно більше інформації, ніж класичним бітам. Заплутаність кубітів дозволяє квантовим системам демонструвати кореляцію між станами незалежно від відстані між ними, що, в поєднанні з інтерференцією, є ключовими факторами для швидшого розв'язання певного роду задач [5].

Поняття квантових бітів. Квантове програмування – це галузь, яка зосереджується на розробці алгоритмів та програм, що використовують принципи квантової механіки для виконання обчислень, які можуть бути значно швидшими або ефективнішими в деяких задачах порівняно з класичними обчисленнями. Завдяки своїм унікальним властивостям, квантові комп'ютери відкривають нові можливості в таких областях, як криптографія, оптимізація, хімічний моделінг і машинне навчання [6].

Основою квантових обчислень є кубіти, або квантові біти. На відміну від класичних бітів, які можуть перебувати у стані 0 або 1, кубіт може існувати в суперпозиції обох станів одночасно. Ця властивість значно збільшує обчислювальну потужність квантових комп'ютерів для певних алгоритмів. Рисунок 2 відображає представлення класичного біта – може мати тільки два значення 1 та 0, квантового біта (кубіта) – представлений у вигляді системи блоха (може перебувати у всіх значеннях одночасно).

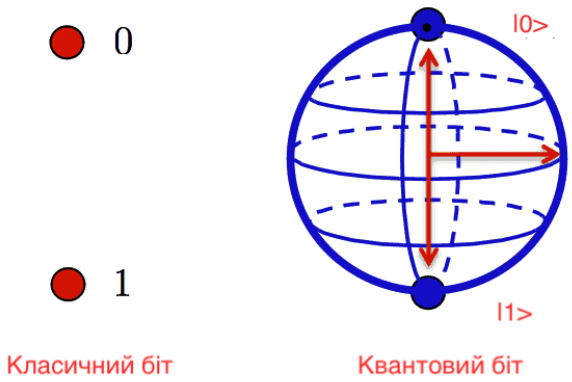

Рис. 2. Представлення класичного біта та кубіта

До основних властивостей квантових бітів можна зарахувати:

1. *Суперпозицію.* Кубіт може знаходитись одночасно в станах $|0\rangle$ та $|1\rangle$, а також в будь-якій лінійній комбінації цих станів. Це описується як $\alpha|0\rangle + \beta|1\rangle$, де α та β – комплексні числа, що визначають ймовірність вимірювання кубіта в кожному зі станів;
2. *Заплутаність*. Кубіти можуть бути заплутані між собою, стан одного кубіта може залежати від стану іншого, незалежно від відстані між ними. Ця властивість дозволяє квантовим алгоритмам виконувати складні обчислення, де результати одних обчислень миттєво впливають на інші, що значно зменшує час виконання програми;
3. *Інтерференцію.* Кубіти можуть також проявляти квантову інтерференцію, яка дозволяє підсилювати правильні відповіді та пригнічувати неправильні у процесі квантового обчислення.

Ці характеристики роблять кубіти надзвичайно потужними для розв'язання певних типів задач, таких як факторизація великих чисел, пошук у базах даних та симуляції квантових систем, які можуть бути недосяжними для класичних комп'ютерів з практичної думки через велику витрату часу або ресурсів [7].

Основи синтаксису Q#. Мова програмування Q# є ключовим компонентом квантової обчислювальної платформи Azure Quantum. Однією з її найцікавіших характеристик є гібридна парадигма програмування,

яка поєднує класичні та квантові обчислення для максимально ефективного розв'язання різних задач. Ця парадигма передбачає розділення логіки програми на класичні та квантові частини, які взаємодіють через інтерфейс між класичним контролером (виконуваним на класичному комп'ютері) та квантовим обчислювальним пристроєм (рис. 3).

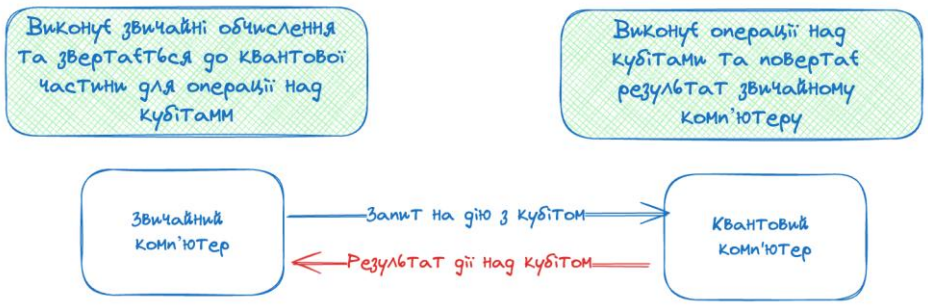

Рис. 3. Схема гібридної взаємодії Q#

У квантових обчисленнях основна частина обчислювальної роботи виконується квантовим процесором або симулятором, тоді як класична програма керує цим процесом та обробляє результати. Комбінуючи теоретичні засади квантових обчислень з практичними можливостями мови Q#, можна зрозуміти потенціал квантового програмування для генерації випадкових чисел та інших завдань. Важливим є те, що Q# не лише надає синтаксичні можливості для розробки квантових алгоритмів, але й дозволяє симулювати виконання програм на квантових комп'ютерах за допомогою квантових симуляторів.

Основи синтаксису мови Q# базуються на концепціях, що є звичними для розробників класичних мов програмування, але також мають унікальні особливості, характерні для квантових обчислень. Програми на Q# організовані у вигляді файлів із розширенням *qs*, кожен з яких містить один або кілька просторів імен. Простори імен слугують для структурування коду та дозволяють уникати конфліктів між іменами різних елементів. Ключовими будівельними блоками програм є операції та функції. Операції використовуються для опису квантових дій над кубітами, враховуючи такі стандартні квантові операції, як вимірювання, застосування квантових вентилів тощо. Функції в Q# відповідають за виконання суто класичних обчислень і не можуть змінювати стан кубітів.

Типи даних у Q# охоплюють як звичні типи, такі як цілі числа, дійсні числа та булеві значення, так і специфічні для квантових обчислень типи, – наприклад, кубіти та результати вимірювань. Кубіти є основним об'єктом у квантових програмах і використовуються в операціях, які виконують квантові обчислення. Операції та функції оголошуються за допомогою ключових слів *operation* і *function*, після яких слідує опис параметрів повернення та їх типів.

Окрему увагу варто приділити операціям керування кубітами – квантовим вентилям. Найпопулярнішим є вентиль Адамара, у квантових схемах та у мові програмування Q# його позначають як **H**. Вентиль Адамара дозволяє перевести кубіт у такий стан, що при вимірюванні вірогідність отримання значення 1 та 0 рівні, ця операція над кубітами лежить в основі всіх квантових схем та алгоритмів. Стандартна бібліотека Q# надає широкий спектр базових квантових операцій, що дозволяють змінювати стан кубітів. Програмісти можуть також створювати власні квантові операції, які складаються з інших операцій або вентилів. Окрім цього, синтаксис мови дозволяє використовувати квантові операції в умовних структурах, надаючи можливість впроваджувати адаптивні квантові алгоритми.

Окрім квантових операцій та функцій, Q# підтримує концепцію callable-файлів, які містять тільки операції та функції для модульної організації коду. Унікальними для Q# є також директиви *open* та *namespace*, які дозволяють імпортувати квантові операції та функції з інших просторів імен, забезпечуючи повторне використання коду [8].

Програма для генерації випадкових чисел (рис. 4)**.** У цій програмі на мові Q# реалізована генерація випадкових чисел за допомогою квантових операцій. Програма розпочинається з визначення простору імен *Random*, де міститься вся логіка квантової програми. Ключові бібліотеки, що відкриваються за допомогою директив *open*, такі:

1. *Microsoft.Quantum.Convert:* забезпечує функції для конвертації результатів квантових вимірювань;
2. *Microsoft.Quantum.Intrinsic:* містить базові квантові операції, такі як квантові вентилі та вимірювання;
3. *Microsoft.Quantum.Math:* надає математичні функції для обчислень.

```
namespace Random {
    // Імпорт та відкриття бібліотек
    open Microsoft.Quantum.Convert;
    open Microsoft.Quantum.Intrinsic;
    open Microsoft.Quantum.Math;

    // Точка входу в програму
    @EntryPoint()
    Run | Histogram | Estimate | Debug | Circuit
    operation Main() : Int {
        // Задається максимальне значення числа
        let max = 100;
        Message($"Sampling a random number between 0 and {max}: ");

        // Викликається функція генерації
        return GenerateRandomNumberInRange(max);
    }

    // Створення функції генерації числа
    operation GenerateRandomNumberInRange(max : Int) : Int {
        mutable bits = [];
        let nBits = BitSizeI(max);
        // Генерація бітів
        for idxBit in 1..nBits {
            set bits += [GenerateRandomBit()];
        }
        let sample = ResultArrayAsInt(bits);

        // Якщо число більше максимуму – запустити генерацію повторно.
        return sample > max ? GenerateRandomNumberInRange(max) | sample;
    }

    // Створення функції генерації біта
    operation GenerateRandomBit() : Result {
        use q = Qubit();
        H(q);

        let result = M(q);

        Reset(q);
        return result;
    }
}
```

Рис. 4. Програма для генерації випадкових чисел мовою програмування Q# [9]

Визначається операція *Main*, яка позначена атрибутом *@EntryPoint()*, що робить її точкою входу для виконання програми. Можна виокремити такі пункти операції:

- операція повертає випадкове число у межах від 0 до max;
- задається константа max = *100*, яка визначає верхню межу випадкового числа;
- використовується квантова операція *GenerateRandomNumberInRange* для генерації випадкового числа в заданому діапазоні;
- повідомлення передається на консоль за допомогою функції *Message*, а результат повертається.

Операція *GenerateRandomNumberInRange(max: Int) : Int* відповідає за генерацію випадкового числа в діапазоні від 0 до max. Можна вирізнити такі пункти операції:

- створюється змінна *bits* як порожній список;
- обчислюється кількість бітів, необхідна для представлення числа max, за допомогою функції *BitSizeI* з бібліотеки *Microsoft.Quantum.Math*;
- в циклі *for idxBit in 1..nBits* генерується список бітів, викликаючи квантову операцію *GenerateRandomBit*;
- функція *ResultArrayAsInt* конвертує список бітів у ціле число;
- якщо отримане число більше за max, операція рекурсивно викликає себе знову для отримання нового значення. Інакше повертається отримане значення.

Операція *GenerateRandomBit() : Result* генерує один випадковий біт за допомогою квантового кубіта. Можна виокремити такі пункти операції:

- використовується оператор *use q = Qubit()* для виділення кубіта;
- застосовується квантовий вентиль Адамара (*H(q)*) для переведення кубіта в суперпозицію станів |0⟩ та |1⟩;
- проводиться вимірювання кубіта (*let result = M(q)*) для отримання випадкового результату *Zero* або *One*;
- кубіт повертається до початкового стану за допомогою операції *Reset(q)*;
- виміряний результат повертається з операції.

Базуючись на коді програми, можна вирізнити структуру програми:

1. *Простір імен та відкриття бібліотек*. Містить основні операції програми та імпортує бібліотеки для виконання квантових операцій;
2. *Операція Main()*. Основна точка входу програми, що викликає функцію генерації випадкових чисел та повертає результат виконання програми (рис. 5);
3. *Операція GenerateRandomNumberInRange()*. Визначає логіку генерації випадкового числа в заданому діапазоні за допомогою квантових бітів;
4. *Операція GenerateRandomBit()*. Використовується для генерування одного випадкового біта з квантовим ефектом суперпозиції.

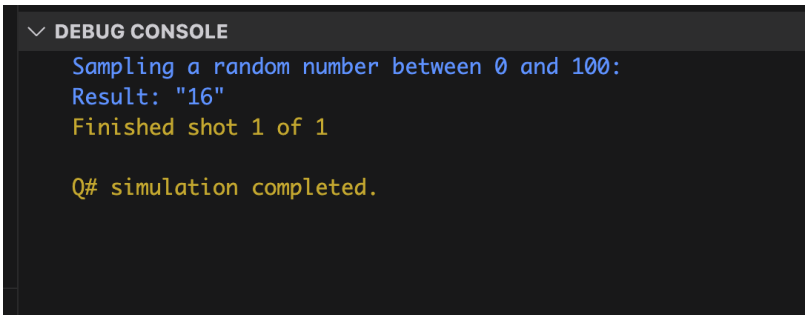

Рис. 5. Результат виконання програми

Висновки. У цій статті було проведено всебічний аналіз квантового програмування, зокрема синтаксичних та логічних структур мови програмування Q#, а також оцінено потенціал квантових обчислень у реалізації алгоритмів для генерації випадкових чисел. Встановлено, що мова Q# забезпечує інтуїтивний синтаксис та ефективну інтеграцію з класичними мовами програмування, що дозволяє реалізувати квантові алгоритми високої складності. Зокрема, використання квантової операції GenerateRandomBit, яка базується на принципі суперпозиції кубітів, дозволяє отримати випадковість високої якості.

Результати дослідження демонструють, що Q# є потужним інструментом для розробки гібридних квантово-класичних алгоритмів, здатних ефективно вирішувати завдання генерації випадкових чисел. Крім того, інтеграція квантових операцій з класичними функціями забезпечує гнучкість та підвищену ефективність алгоритмів. Використання хмарних сервісів, таких як Azure Quantum, відкриває перспективи для масштабування квантових обчислень та їх тестування на реальних квантових процесорах.

Подальші дослідження у цьому напрямі мають бути спрямовані на вдосконалення методів генерації випадкових чисел та розробку більш складних квантових алгоритмів, що використовують переваги гібридних обчислень. Таким чином, мова Q# виявилася перспективним інструментом для розробки квантових програм, які можуть значно підвищити ефективність розв’язання обчислювальних задач.

Список використаної літератури:

1. Azure Quantum [Electronic resourse]. – Access mode : https://quantum.microsoft.com/.
2. IBM Quantum [Electronic resourse]. – Access mode : https://www.ibm.com/quantum.
3. Google Quantum AI [Electronic resourse]. – Access mode : https://quantumai.google/.
4. The Quantum Insider [Electronic resourse]. – Access mode : https://app.thequantuminsider.com/.
5. *Крохмальський В.Д.* Вступ до квантового обчислення та квантової інформації / *В.Д. Крохмальський*. – ЛНУ імені Івана Франка, 2021 [Електронний ресурс]. – Режим доступу : http://ktf.lnu.edu.ua/books/Krokhmalskii-VKO.pdf.
6. Квантове програмування [Електронний ресурс]. – Режим доступу : https://uk.wikipedia.org/wiki/%D0%9A%D0%B2%D0%B0%D0%BD%D1%82%D0%BE%D0%B2%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F.

7. Кубіт [Електронний ресурс]. – Режим доступу : https://uk.wikipedia.org/wiki/%D0%9A%D1%83%D0%B1%D1%96%D1%82.
8. Документація Q# [Електронний ресурс]. – Режим доступу : https://microsoft.github.io/qsharp/.
9. Tutorial: Implement a quantum random number generator in Q# [Electronic resourse]. – Access mode : https://learn.microsoft.com/en-us/azure/quantum/tutorial-qdk-quantum-random-number-generator.

References:

1. Azure Quantum, [Online], available at: https://quantum.microsoft.com/
2. IBM Quantum, [Online], available at: https://www.ibm.com/quantum
3. Google Quantum AI, [Online], available at: https://quantumai.google/
4. The Quantum Insider, [Online], available at: https://app.thequantuminsider.com/
5. Krokhmalskyi, V.D. (2021), *Vstup do kvantovoho obchyslennia ta kvantovoi informatsii*, LNU imeni Ivana Franka, [Online], available at: http://ktf.lnu.edu.ua/books/Krokhmalskii-VKO.pdf
6. «Kvantove prohramuvannia», [Online], available at: https://uk.wikipedia.org/wiki/%D0%9A%D0%B2%D0%B0%D0%BD%D1%82%D0%BE%D0%B2%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F
7. «Kubit», [Online], available at: https://uk.wikipedia.org/wiki/%D0%9A%D1%83%D0%B1%D1%96%D1%82
8. *Dokumentatsiia Q#*, [Online], available at: https://microsoft.github.io/qsharp/
9. «Tutorial: Implement a quantum random number generator in Q#», [Online], available at: https://learn.microsoft.com/en-us/azure/quantum/tutorial-qdk-quantum-random-number-generator

Сапожник Дмитро Олегович – аспірант кафедри програмної інженерії Державного університету «Житомирська політехніка».

E-mail: asp_sdo@student.ztu.edu.ua.

Sapozhnyk D.O.

Analysis of Q# Programming Language Capabilities through Implementation of a Random Number Generation Program

This paper analyzes the capabilities of the Q# programming language through the implementation of a random number generation program. Attention is drawn to the main tools of quantum programming, and an analysis of investment trends in quantum technologies is provided. The article reviews publications in the field of quantum programming and emphasizes the key features of quantum bits. The main focus is on using the hybrid programming paradigm, which combines classical and quantum computing. The structure and features of the quantum program in Q# are described, specifically the GenerateRandomBit and GenerateRandomNumberInRange operations, which ensure the generation of random bits and numbers through quantum superposition and measurement. The advantages of Q# in combination with classical programming languages for effective use of quantum algorithms in cryptography and statistical modeling are highlighted. The conclusion emphasizes the prospects of the Q# programming language as a tool for developing quantum programs in a hybrid environment, enabling the potential of quantum computing to be leveraged together with classical approaches.

Keywords: Q# programming language; quantum computing; random number generation; qubits; quantum superposition; hybrid programming paradigm; quantum algorithms.

Стаття надійшла до редакції 12.04.2024.