Pesquisa “quântica” em vetor

Mário Leite

...

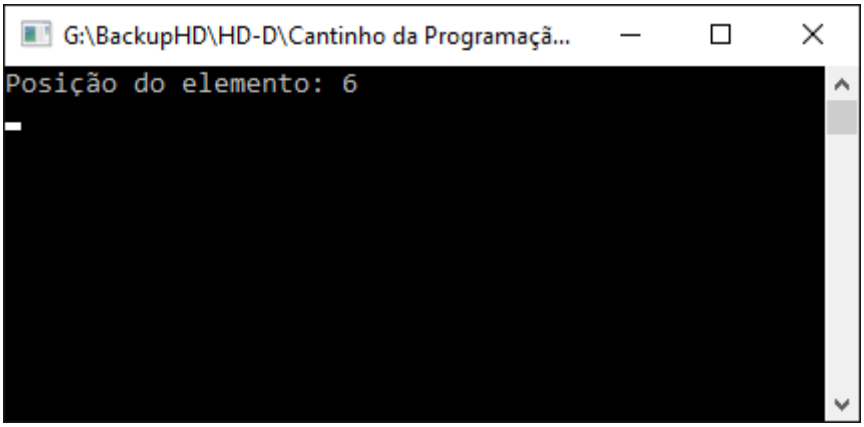
A onda, agora, é comentar termos quânticos: “computador quântico”, “entrelaçamento”, *qubits*, etc, aplicados à programação. Eu disse “onda”, porque na verdade, de tempos em tempos aparece uma nova “onda” na programação, deixando os candidatos a programador meio confusos, que começam a procurar uma “nova” linguagem para aprender aplicar esses conceitos e se esquecem do BÁSICO em programação: LÓGICA. Mas, infelizmente, muitos programadores - até os mais experientes - acham que a Lógica não é muito necessária: basta empregar o” paradigma correto” (as aspas são intencionais, mesmo) e o problema estará resolvido! Eu, embora seja um velho programador (mas, não um programador velho) ainda acho que na Programação (não confundir com Codificação) a Lógica é o fator mais importante, pois o objetivo é resolver o problema do USUÁRIO, e procurar na *codificação* a solução mais moderna para o problema é uma contradição com o termo PROGRAMAÇÃO!

Observe o código abaixo em C# (modo *console*): ele implementa um programa para procurar um determinado elemento numa lista de inteiros, contidos num vetor. Note que esse código se envereda por caminho de solução “quântica” muito complexa e cheia de desvios estranhos, tornando a codificação muito “pesada”, confusa, e exigindo um certo “malabarismo” para entender; funciona!? Sim, funciona; MAS a um custo-benefício pífio, apenas para usar o termo *chic* “quântico” na implementação do “Algoritmo de Glove”r!

Agora, observe a segunda solução em *pseudocódigo* bem simples e inteligível: sem empregar sofisticações desnecessárias. É bem direto, que até um programador iniciante entende sem precisar conhecer nenhum paradigma em linguagem real! E se alguém estiver preocupado com problemas mais “modernos” como: reconhecimento facial, localização por GPS, coordenadas de mísseis hipersônicos, solução para o ” Gato de Schrödinger”, Geometria Não Euclidiana, Inteligência Artificial, e outros “*balangandans*”, que não se iluda: TUDO PASSA PELA LÓGICA; muito antes de seguir aquele *paradigma fantástico* visto naquele vídeo sobre programação no You Tube!

Em tempo: Eu não estou querendo dizer que a Codificação não seja importante: É SIM; mas quero afirmar que codificar é apenas (e tão somente) a representação da programação seguindo as regras de uma linguagem real; mas, não é a solução do problema. A solução do problema é dada, muito antes: na programação do algoritmo que implementa essa solução! Em tom de brincadeira, eu diria que “*Codificação é um avatar da Programação*”.

```
/*
PesqQuant.cs
Pesquisa um determinado elemento num vetor de inteiros simulando conceitos quânticos
bem básicos.
-------------------------------------------------------------------------------
data: 21/05/2023
-------------------------------------------------------------------------------
*/

using System;
namespace PesqQuant
{
    class Program
    {
        static void Main(string[] args)
        {
            int[] vetNum = new int[] { 1, 1, 2, 3, 5, 8, 13, 21 };
            int ind = PesqQuant(vetNum, 8);  //pesquisa elemento de valor 8
            Console.WriteLine("Posição do elemento: " + (ind+1);
            Console.ReadKey();
        }
        //-----------------------------------------------------------------------
        static int PesqQuant(int[] vetNum, int elemento)
        {
            / Obtém o número de qubits necessários para representar "vetNum"
            int numQubits = (int)Math.Ceiling(Math.Log(vetNum.Length, 2));

            //Cria um circuito quântico com um dado número de qubits
            var circuito = new AlgorGlover(numQubits, vetNum);

            // Executa o circuito quântico
            ResultPesq resultado = circuito.Run(elemento);

            //Retorna o índice encontrado pelo algoritmo
            return resultado.Indice;
        }
    }
    //-----------------------------------------------------------------------
    class AlgorGlover
    {
        private int numQubits;
        private int[] vetNum;

        public AlgorGlover(int numQubits, int[] vetNum) {
            this.numQubits = numQubits;
            this.vetNum = vetNum;
        }
        //-----------------------------------------------------------------------
        public ResultPesq Run(int elemento)
        {
            //Pode implementar aqui o "Algoritmo de Grover"

            // Aqui está um exemplo simplificado retornando o índice do elemento
            for (int i = 0; i < vetNum.Length; i++) {
                if (vetNum[i] == elemento) {
                    return new ResultPesq() { Indice = i };
                }
            }
            // Se o elemento não for encontrado, retorna -1
            return new ResultPesq() { Indice = -1 }; }
        }
    }
    //-----------------------------------------------------------------------
    class ResultPesq {
        public int Indice { get; set; }
    }
} //Fim da aplicação ---------------------------------------------------------
```



Resultado da pesquisa do elemento de valor 8

```
Programa "PesquisaElemento"
{Pesquisa um elemento em um vetor lido.
Pesquisa um determinado elemento num vetor de inteiros usando
conceitos quânticos básicos
-------------------------------------------------------------------
Autor: Mário Leite
data: 21/05/2023
}
-------------------------------------------------------------------
Declare VetNum: array[100] de inteiro
   j, n, X, EleNum, EleVet: inteiro
   Num: real
   Achou: lógico
Início
   MAXELE=100  //Estabelece um limite para o tamanho do vetor
   Repita
      Escreva("Quantos elementos terá o vetor [min:2 - max:", MAXELE, "]?: ")
      Leia(n)
   AtéQue((n>=2) e (n<= MAXELE))
   Para j De 1 Até n Faça
      Escreva("Digite um número inteiro: ")
      Leia(Num)
      VetNum[j] ← Int(Num)
   FimPara
   EscrevaLn("")  //apenas salta uma linha
   {Acessa um determinado elemento do vetor}
   EscrevaLn("")
   Escreva("Digite o número (ordem) do elemento desejado: ")
   Leia(EleNum)
   Achou ← Falso
   Para j De 1 Até n Faça
      Se(j=EleNum) Então
         X ← j
         Achou ← Verdade
         Abandone  //achou: abandona o loop
      FimSe
   FimPara
   EscrevaLn("")
   EscrevaLn("")
   Se(Achou) Então
      EleVet ← VetNum[X]
      EscrevaLn("O elemento", X, " do vetor é:", EleVet)
   Senão
      EscrevaLn("Não existe elemento no vetor com índice",EleNum")
   FimSe
FimPrograma
```