

## Declaração Otimizada De Variáveis De Controle de Loops

Mário Leite

...

Existem três tipos básicos de estruturas de repetição, também conhecidas como *loops* (laços): dois tipos lógicos e um numérico; todos os outros tipos que aparecem como recursos extras nas linguagens de programação são derivados destes três. “Loop lógico com teste no início”, “Loop lógico com teste no final” e “Loop numérico”: em pseudocódigo: “Enquanto..Faça”, “Faça..AtéQue” e “Para..FimPara”, respectivamente. E como definem estruturas de controle, algumas linguagens permitem que as variáveis que os controlam sejam declaradas localmente a elas; isto é, o escopo destas variáveis passa a ser local ao *loop*, sendo visíveis (accessíveis) somente dentro destas estruturas. Este recurso é uma grande vantagem para essas linguagens, pois, sendo as variáveis endereços da memória RAM ficam disponíveis logo após o término do *loop*, tornando o processamento mais eficiente. Deste modo, declarando as variáveis de controle dentro de seus respectivos *loops*, elas são imediatamente descartadas da memória após execução do bloco de instruções dessas estruturas, enquanto que as outras (fora do *loop*) permanecem ocupando espaço na memória até o término da rotina. Observe as declarações das variáveis de controle de *loops*: **j** e **k**, dentro de estruturas **for** no programa “PrMostraPrimos” (codificado em **C#** Console), cujo objetivo é listar os números primos na faixa definida pelo usuário. A saída deste programa é mostrada na **figura 1**.

---

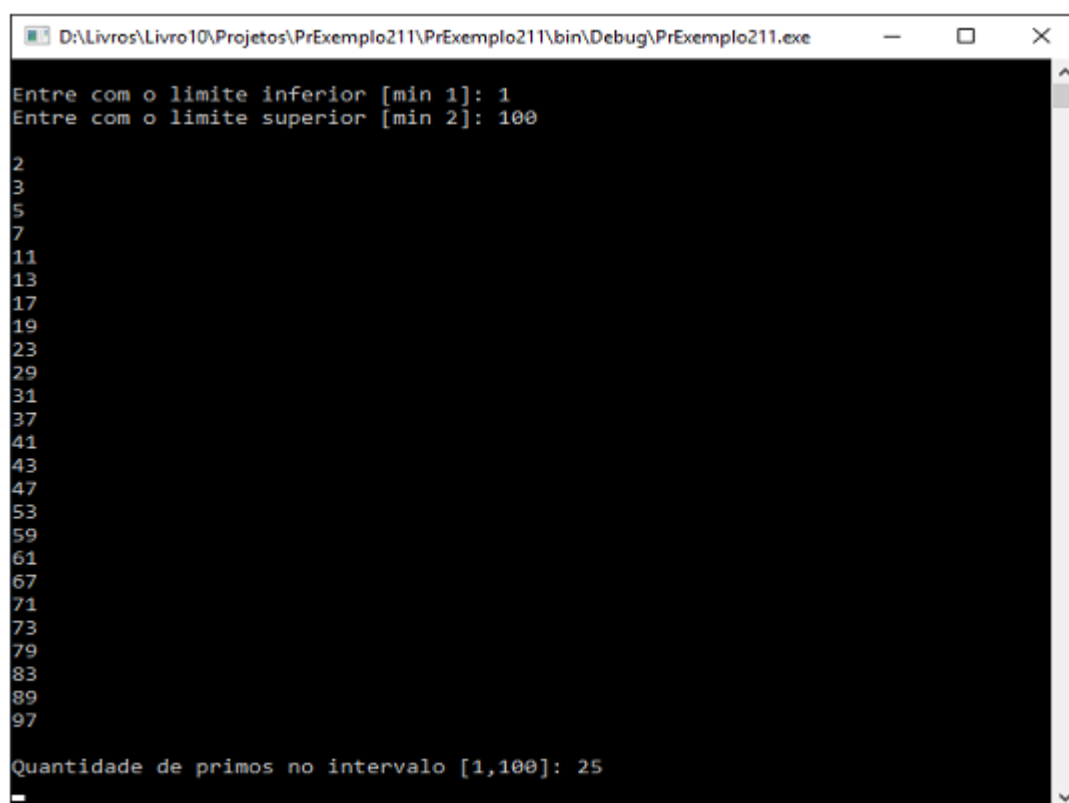
Para adquirir o *pdf/e-book* deste livro ou o *pdf* de outros livros sobre programação, entre em contato pelo *e-mail*: **marleite@gmail com**

---

```

namespace PrMostraPrimos
{
    class Program
    {
        static void Main(string[] args)
        {
            //Mostra os números primos em uma faixa definida pelo usuário
            //Em C#/Console
            //Autor: Mário Leite
            //-----
            int cont;
            uint R, num, limInf, limSup;
            bool cond, q;
            //Inicializações convenientes
            limInf = 1;
            limSup = 0;
            cond = (limInf>limSup) || ((limInf<1) || (limSup < 2));
            while(cond)
            {
                Console.WriteLine("");
                Console.Write("Entre com o limite inferior [min 1]: ");
                limInf = uint.Parse(Console.ReadLine());
                Console.Write("Entre com o limite superior [min 2]: ");
                limSup = uint.Parse(Console.ReadLine());
                cond = (limInf>limSup) || ((limInf<1) || (limSup < 2));
            }
            Console.WriteLine(""); //salta linha
            cont = 0;
            for (uint j=limInf; j<=limSup; j++)
            {
                num = j;
                q = true;
                for (uint k=2; k<=(num-1); k++)
                {
                    R = (num % k);
                    if (R == 0)
                    {
                        q = false;
                    }
                }
                if ((q) && (num != 1))
                {
                    Console.WriteLine(num);
                    cont++; //acumula a quantidade de primos
                }
            }
            Console.WriteLine("");
            Console.WriteLine("Quantidade de primos no intervalo" +
                             " [{0},{1}]: {2} ",limInf, limSup,cont);
            Console.ReadKey(); //aguarda pressionar uma tecla
        }
    }
} //fim do programa

```



```
D:\Livros\Livro10\Projetos\PrExemplo211\PrExemplo211\bin\Debug\PrExemplo211.exe
Entre com o limite inferior [min 1]: 1
Entre com o limite superior [min 2]: 100

2
3
5
7
11
13
17
19
23
29
31
37
41
43
47
53
59
61
67
71
73
79
83
89
97

Quantidade de primos no intervalo [1,100]: 25
```

**Figura 1 - Saída do programa**