

# Manual do Código - Aplicação de Cálculo de Sequências Lógicas

## 1. Introdução

Este manual tem como objetivo explicar o funcionamento do código desenvolvido para uma Aplicação de Cálculo de Sequências Lógicas. O programa permite ao usuário realizar diversos cálculos matemáticos, como o cálculo de números triangulares, a geração de sequências de números primos e o cálculo do fatorial. Este documento oferece uma descrição detalhada dos processos de entrada de dados, processamento das operações e exibição dos resultados, proporcionando uma visão abrangente de como o programa é estruturado e executado.

## 2. Ferramentas Utilizadas

O programa foi desenvolvido com a ferramenta VisualG, que permite a criação e execução de algoritmos utilizando a linguagem Portugol. O Portugol é uma linguagem de programação didática, projetada para ensinar lógica de programação de forma clara e acessível. O VisualG é amplamente utilizado em cursos introdutórios de programação, ajudando os usuários a entenderem os conceitos fundamentais da área, antes de avançarem para linguagens mais complexas e específicas.

## 3. Declaração de Variáveis

O programa utiliza diversas variáveis do tipo real e inteiro para o funcionamento do menu e das operações. As variáveis são organizadas conforme o tipo de operação matemática ou como variáveis universais para o programa como um todo.

Variáveis Universais:

- **i**: Índice ou contador utilizado nas estruturas de repetição.
- **resultado**: Armazena o resultado final das operações e é utilizado para exibir o resultado ao usuário.

## 4. Estrutura do Código

O código é estruturado em duas partes principais:

1. **Menu Principal**: Exibe as opções de operações matemáticas para o usuário e permite que ele escolha qual operação deseja executar.
2. **Aba de Operações**: Realiza os cálculos das operações selecionadas e exibe os resultados ao usuário.

O menu está contido em uma estrutura de repetição **repita**, que mantém o menu visível até que o usuário escolha sair. As operações são divididas em funções **escolha**, que são executadas quando a condição de cada operação é atendida.

## 5. Menu

Variável utilizada na estrutura do menu:

- **op**: Variável do tipo real que controla o menu. Ela determina quando o menu será exibido, qual operação será executada ou se o programa será encerrado.

Exibição do Menu:

O programa exibe o menu interativo logo no início da execução, utilizando a função **escreval**. Inicialmente, a variável **op** recebe o valor 1, o que faz com que o menu apareça na tela dentro de uma estrutura de repetição **enquanto**.

```
var
  op, termo, comeco : Real
  fatorial, num1, resultado, i, quantidade_numeros_primos, numero_atual, contador

Início
  op <- 1

  Enquanto (op = 1) faça //Estrutura de repetição para o menu aparecer várias

    // Exibe o menu inicial
    escreval("+-----+")
    escreval("| » MENU - SEQUÊNCIAS MATEMÁTICAS « |")
    escreval("+-----+")
    escreval("| 1 | Números Triangulares |")
    escreval("| 2 | Sequência de Números Primos |")
    escreval("| 3 | Sequência Fatorial |")
    escreval("| 4 | Sequência de Cubos |")
    escreval("| 5 | Sequência Alternada |")
    escreval("| 6 | Sequência Quadrados Perfeitos |")
    escreval("| 0 | Sair |")
    escreval("+-----+")
    escreval("| By DREAM TEAM • Est. 2025 |")
    escreval("+-----+")

    Leia(op)

    Enquanto (op > 6) ou (op < 0) faça //Limitar a escolha das opções
      Escreval("De 0 a 3 apenas")
      Leia(op)
    Fimenquanto

    se (op = 0) então
      fimse
    limpatela
```

O menu exibe ao usuário todas as operações disponíveis que podem ser executadas pelo programa. Com base nessas opções, o usuário pode escolher a operação desejada, selecionando um número de 1 a 6, ou optar por encerrar o programa ao escolher a opção 0.

## 6. Escolhendo a Operação

Após a exibição do menu na tela, o programa solicita ao usuário que digite o número correspondente à operação que deseja executar. Em seguida, utilizando a função **leia**, o código captura a opção fornecida pelo usuário e armazena na variável **op**. Através da estrutura **escolha**, o código verifica se a opção escolhida é válida e determina qual operação será executada.

```
    escolha op
    caso 1

    caso 2

    caso 3
```

Se a opção for válida, o programa limpa a tela, removendo o menu, e começa a executar a operação matemática escolhida pelo usuário. O código também garante que apenas opções válidas sejam aceitas, utilizando a estrutura de repetição **enquanto** para restringir as entradas do usuário às opções disponíveis no menu.

```
Leia(op)

Enquanto (op > 6) ou (op < 0) faca //Limitar a escolha das opções
    Escreval("De 0 a 6 apenas")
    Leia(op)
Fimenquanto

se (op = 0) entao |
fimse
limpatela
```

Se o valor atribuído pelo usuário à variável **op** for maior que 6 ou menor que 0, o programa exibirá uma mensagem informando que o valor é inválido e solicitará que o usuário insira um valor válido. Esse processo continuará até que um valor válido seja fornecido. Caso o usuário digite o valor 0 (última opção do menu), o programa será encerrado e a tela será limpa.

```
30      se (op = 0) entao
31      fimse
32      limpatela
```

Por fim, após a conclusão da operação escolhida, o programa pergunta ao usuário se deseja encerrar o programa. Caso o usuário digite 1, o programa será encerrado. Se o valor digitado for 0, o menu será exibido novamente, permitindo que o usuário escolha outra operação e continue utilizando o programa.

```

repita
    Escreval("Deseja continuar no programa? (0 para CONTINUAR, 1 para SAIR)")
    leia(op)
ate (op = 0) ou (op = 1) // Garante que só aceita 0 ou 1

// Lógica invertida para controle do loop principal
se op = 0 entao // Se digitou 0 (quer CONTINUAR)
    op <- 1 // Mantém op=1 para continuar no loop
senao // Se digitou 1 (quer SAIR)
    op <- 0 // Muda op para 0 para sair do loop
    Escreval("=====")
    Escreval(" DREAM TEAM AGRADECE, ATÉ MAIS! ")
    Escreval("=====")
fimse

Fimenquanto
fimalgoritmo

```

## 7. Operações Matemáticas

Cada operação está contida em uma estrutura condicional **escolha**, que será executada quando a opção escolhida pelo usuário corresponder ao número da operação desejada. Assim que a operação é iniciada, o menu é limpo da tela e o número da operação escolhida é exibido, indicando ao usuário que a execução da operação está em andamento. Ao finalizar a operação, ou seja, quando o resultado for exibido na tela, o código exibe uma mensagem informando que a operação foi concluída.

Ao Início da Operação:

```

36      escreval("=====")
37      escreval("  NUMEROS TRIÂNGULARES  ")
38      escreval("=====")

```

Ao Término da Operação:

```

46      escreval("=====")
47      escreval("  FIM DE NUMEROS TRIÂNGULARES  ")
48      escreval("=====")
49      Fimse

```

## 8. Números Triangulares

Definição:

Um número triangular é um número que pode ser representado por pontos dispostos em forma de triângulo equilátero. Esses números seguem uma sequência crescente e têm a propriedade de formar triângulos quando representados graficamente.

Sequência:

A sequência de números triangulares começa com 1, 3, 6, 10, 15, 21,

Fórmula:

O n-ésimo número triangular ( $T_n$ ) pode ser calculado pela fórmula:

$$T_n = n(n+1) / 2$$

Onde  $n$  representa a posição do número na sequência. Por exemplo, o 1º número triangular é  $T_1 = 1$ , o 2º número triangular é  $T_2 = 3$ , e assim por diante.

Variáveis Utilizadas na Operação:

- **numfi**: Armazena a quantidade de sequências fornecidas pelo usuário.
- **resultado**: Armazena o resultado do cálculo da quantidade de triângulos possíveis.

Entrada de Dados:

O programa solicita que o usuário informe um valor numérico representando a quantidade de sequências desejadas.

```
39     escreval ("Informe o valor de sequências: ")//solicita que o usuário informe a quantidade de sequências que deseja
40     leia (num1)//faz a leitura da variável
```

Processamento:

O programa utiliza um laço de repetição para identificar números triangulares, calculando cada número da sequência com base na fórmula.

```
42     resultado <- num1*(num1 +1)\ 2)//a variável resultado recebe a operação lógica
```

Saída de Dados:

O programa exibe a quantidade de números triangulares gerados até que o usuário atinja a quantidade desejada.

```
44     escreval ("A quantidade de triângulos possíveis será de: ", resultado , " triângulos")
45
46         escreval("=====")
47         escreval("      FIM DE NUMEROS TRIÂNGULARES  ")
48         escreval("=====")
49     Fimse
```

## 9. Sequência de Números Primos

Definição:

A sequência de números primos é composta por números naturais maiores que 1 que não podem ser divididos exatamente por nenhum outro número além de 1 e ele mesmo.

Sequência:

A sequência de números primos começa com 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, ...

Fórmula:

Não existe uma fórmula simples para gerar números primos, mas eles podem ser identificados por meio de um processo de verificação. Um número  $n$  é considerado primo se não for divisível por nenhum número natural entre 2 e  $\sqrt{n}$ .

Variáveis Utilizadas na Operação:

- **quantidade\_numeros\_primos:** Armazena a quantidade de números primos que o usuário deseja gerar.
- **numero\_atual:** Número que está sendo verificado para determinar se é primo.
- **contador\_primos:** Conta quantos números primos já foram encontrados.
- **eh\_primo:** Variável auxiliar usada para verificar se o número atual é primo ou não.

Entrada de Dados:

O programa solicita ao usuário a quantidade de números primos que deseja gerar.

```
58         escreva("Digite a quantidade de números primos que deseja gerar: ")
59         leia(quantidade_numeros_primos)
```

Processamento:

O código usa um laço de repetição para verificar cada número natural, a partir de 2, para saber se ele é primo. O número **numero\_atual** é testado para divisibilidade por números entre 2 e o próprio número. Se for divisível por algum número além de 1 e ele mesmo, o número não é primo e o processo continua com o próximo número. Se o número for primo, ele é adicionado à lista de números primos.

```
61         numero_atual <- 2 // Primeiro número a ser testado
62         contador_primos <- 0 // Conta quantos primos já foram encontrados
63
64         // Loop para encontrar os números primos
65         enquanto contador_primos < quantidade_numeros_primos faca
66             eh_primo <- 1 // Assume que o número é primo
67
68             // Verifica se o número atual é primo
69             para i de 2 ate numero_atual - 1 faca
70                 se (numero_atual mod i = 0) entao
71                     eh_primo <- 0 // O número não é primo
72             fimse
73         fimpara
```

Saída de Dados:

Os números primos são exibidos até que a quantidade desejada seja atingida. O código imprime os números primos encontrados na sequência solicitada.

```

75         // Se for primo, exibe e incrementa o contador
76         se eh_primo = 1 entao //eh_primo 1 para verdadeiro, 0 para falso
77             escreval(numero_atual, " ")
78             contador_primos <- contador_primos + 1
79         fimse
80
81         numero_atual <- numero_atual + 1 // Testa o próximo número
82     fimenquanto
83     escreval("=====")
84     escreval("      FIM DA SEQUÊNCIA DE PRIMOS   ")
85     escreval("=====")
86
87     Fimse

```

## 10. Sequência Fatorial

Definição:

O fatorial de um número natural  $n$  é o produto de todos os inteiros positivos menores ou iguais a  $n$ . Ele é denotado por  $n!$ . O fatorial é amplamente utilizado em combinações e permutações, além de outras áreas da matemática.

Sequência:

A sequência de fatoriais começa com  $1!$ ,  $2!$ ,  $3!$ ,  $4!$ ,  $5!$ , ...

- $1! = 1$
- $2! = 2 \times 1 = 2$
- $3! = 3 \times 2 \times 1 = 6$
- $4! = 4 \times 3 \times 2 \times 1 = 24$
- $5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$

E assim por diante.

Fórmula:

A fórmula do fatorial de  $n$  é:

$$n! = n \times (n-1) \times (n-2) \times \dots \times 1$$

Variáveis Utilizadas na Operação:

- **fatorial:** Número fornecido pelo usuário para o cálculo do fatorial.
- **resultado:** Armazena o valor do fatorial, que é o resultado final do cálculo.

- i: Índice utilizado no laço de repetição para realizar a multiplicação dos números inteiros de 1 até **fatorial**.

Entrada de Dados:

O programa solicita que o usuário insira um número inteiro positivo (menor que 13) para o cálculo do fatorial.

```

99         escreval ("Por favor, insira um número inteiro, não negativo. (E menor que 13):")
100        leia (fatorial)

```

Processamento:

O cálculo do fatorial é realizado por meio de um laço de repetição, onde o número **fatorial** é multiplicado pelos números decrescentes até chegar a 1. O laço vai multiplicando e armazenando o resultado parcial na variável **resultado**.

```

101        // Estrutura de repetição enquanto utilizada para evitar números não compatíveis com o programa
102        enquanto (fatorial > 12) ou (fatorial < 0) faça
103            escreval("Este programa não suporta esse número. Por favor, tente novamente.")
104            leia (fatorial)
105        fimenquanto
106        // Segunda estrutura de repetição, utilizada para calcular o fatorial do número utilizado
107        enquanto fatorial >= 1 faça
108            resultado <- (resultado * fatorial)
109            fatorial <- (fatorial - 1)
110        fimenquanto

```

Saída de Dados:

Após a conclusão do cálculo, o resultado final do fatorial é exibido ao usuário. Se o número inserido for 5, por exemplo, o programa exibirá:

O fatorial de 5 é: 120.

```

111        escreval (resultado)
112
113        // Fim do algoritmo
114        escreval("=====")
115        escreval("      FIM DA SEQUÊNCIA FATORIAL    ")
116        escreval("=====")
117        Fimse

```

## 11.Sequência de Cubos

Definição:

- Calcula uma sequência de números naturais elevados ao cubo ( $n^3$ ).

Sequência:

- 1, 8, 27, 64, 125, 216, ...



Fórmula:

- termo =  $i * i * i$  ou  $i^3$ .

Variáveis Utilizadas na Operação:

- numcubo: Quantidade de termos desejada.
- x: Contador para a posição atual.
- resultadocubo: Armazena o valor do cubo calculado.

Entrada de Dados:

O programa solicita que o usuário informe um valor numérico representando a quantidade de sequências desejadas.

```
escreval("Digite o número de cubos desejável:")
leia(numcubo)
```

Processamento:

O laço para calcula o cubo de cada número de 1 até numcubo.

Fórmula  $x * x * x$ : Multiplica o contador x por ele mesmo três vezes (equivalente a  $x^3$ ).

```
para x de 1 ate numcubo faca
    resultadocubo <- x*x*x
    escreval(resultadocubo)
fimpara
```

Saída de Dados:

A sequência é exibida em linha

```
    escreval(resultadocubo)
fimpara

escreval("=====")
escreval("      FIM DA SEQUÊNCIA DE CUBOS      ")
escreval("=====")
```

## 12. Sequência Alternada

Definição:

Esta operação gera uma sequência numérica que alterna entre valores positivos e negativos, com base em um número inicial fornecido pelo usuário.

Sequência:

- Exemplo com número inicial = 2: 2, -4, 6, -8, 10, -12, ...

Fórmula:

- Cada termo da sequência é calculado por: termo =  $(-1)^{(i + 1)} * (\text{comeco} * i)$

Onde:

- i: Posição atual na sequência.
- comeco: Número inicial fornecido pelo usuário.

Variáveis Utilizadas na Operação:

- quantidade\_termos: Armazena quantos termos o usuário deseja gerar (inteiro positivo).
- comeco: Número inicial da sequência (pode ser positivo ou negativo).
- i: Contador para a posição atual na sequência.

Entrada de Dados:

O programa solicita que o usuário informe um valor numérico representando a quantidade de termos da sequências desejadas.

Após isto o programa solicita o número por onde o usuário deseja iniciar a sequencia.

Podendo ser este número positivo ou negativo

```
escreval("Quantos termos da sequência alternada de pares você deseja? ")
leia(n)
// usuario informa valor icinial da sequencia. sendo negativo ou positivo
escreval("Qual o numero inicial da sequencia? ")
leia(comeco)
```

Processamento:

Um laço para é usado para gerar cada termo da sequência, de 1 até quantidade\_termos.

Fórmula termo =  $(-1)^{(i + 1)} * (\text{comeco} * i)$ :

$(-1)^{(i + 1)}$ : Alterna o sinal do termo. Se i for ímpar, o resultado é +1 (termo positivo); se par, -1 (termo negativo).

$(\text{comeco} * i)$ : Multiplica o número inicial pela posição atual (i), garantindo que o valor absoluto cresça linearmente.

Se comeco = 2 e i = 3, o termo será  $(-1)^4 * (2 * 3) = +6$ .

```
para i de 1 ate n faca
    termo <-  $(-1)^{(i + 1)} * (\text{comeco} * i)$ 
    escreva(termo, " ") // impressao da sequencia
fimpara
```

Saída de Dados:

Os termos são exibidos em linha (ex: 2 -4 6 -8 10).

```

        escreva(termo, " ") // impressao da sequencia
fimpara
escreval ("" )
escreval ("=====")
escreval ("          FIM DA SEQUÊNCIA DE ALTERNADA          ")
escreval ("=====")

```

### 13. Sequência de Quadrados Perfeitos

Definição:

- Gera uma sequência de números naturais elevados ao quadrado ( $n^2$ ), em ordem crescente.

Sequência:

- 1, 4, 9, 16, 25, 36, ...

Fórmula:

- termo =  $i * i$  Onde  $i$  é o número natural da posição atual.

Variáveis Utilizadas na Operação:

- quantidade\_quadrados: Quantidade de termos desejada pelo usuário.
- i: Contador para a posição atual.

Entrada de Dados:

O programa solicita que o usuário informe um valor numérico representando a quantidade de sequências desejadas.

```

escreval ("Digite a quantidade de numeros que deseja que apareça")
leia (i)

```

Processamento:

O laço para itera de 1 até o valor informado pelo usuário.

Cálculo  $i * i$ : Eleva o contador  $i$  ao quadrado

Cada resultado é exibido imediatamente com escreva().

```

para n de 1 ate i faca
    nQ <- n * n
    escreval (nQ)
fimpara

```

Saída de Dados:

Os quadrados perfeitos são listados em sequência

```
    escreval (nQ)
fimpara
escreval("=====")
escreval("      FIM DA QUADRADOS PERFEITOS      ")
escreval("=====")
```

#### 14. Sequência Geométrica

Definição:

- Gera uma progressão geométrica de acordo com as especificações do usuário, como número inicial, razão (q) e o número de termos que existem na sequência.

Sequência:

- A depender do número inicial da sequência, razão e quantidade de termos inserida

Variáveis Utilizadas na Operação:

- N1 - Número que dará início na sequência
- Q – Lê a razão da sequência
- I - Contador para a posição atual
- Sequencia – Gera a quantidade de sequência na qual o programa irá exibir

Entrada de Dados:

O programa solicita que o usuário informe os valores das variáveis n1, q e sequencia, os valores digitados pelo usuário são atribuídos para sua variáveis desejadas.

```
escreva ("digite o primeiro numero da sequência: ")
leia (n1)

//Usuario informara o valor de q da progressão sequencial
escreva ("digite a razão: ")
leia (q)

//Usuário informa a quantidade de elementos da sequência (
escreva ("quantos números da sequência deseja: ")
leia(sequencia)
```

Processamento:

Na função para será impresso na tela o n1, mostrando o início da sequência. Após o numero ser impresso a variável recebera o produto de  $n1 * q$ , alterando valor de n1 com o próximo número da sequência

```

escreval ("os", sequencia, " primeiros numeros serão: ")
//a funcao para ira imprimir os numeros da sequencia e calc
para i de 1 ate sequencia faca
    escreval (n1) //imprimir todos os numeros da sequencia na
    n1 <- n1*q //calculo do proximo numero da sequencia
fimpara

```

Saída de Dados:

Com i igual a sequencia, a função para é encerrada. E com a sequência já criada e impressa na tela para o usuário com a quantidade de termo desejada por ele, o programa mostra na tela a mensagem

```

escreval("=====")
escreval("          FIM DE SEQUÊNCIA GEOMÉTRICA          ")
escreval("=====")

```

## 15. Sequência de Tribonacci

Definição:

- Este algoritmo gera e exibe a sequência de Tribonacci até o termo n escolhido pelo usuário, onde cada termo a partir do quarto é a soma dos três termos anteriores.

Sequência:

- 0,0,1....

Fórmula:

- termo =  $i * i$  Onde i é o número natural da posição atual.

Variáveis Utilizadas na Operação:

- n (inteiro): número do termo máximo até onde a sequência será calculada.
- i (inteiro): variável de controle para o laço de repetição
- t0, t1, t2 (inteiros): armazenam os três últimos termos da sequência para cálculo do próximo termo.
- t3 (inteiro): próximo termo calculado como soma dos três anteriores.

Entrada de Dados:

Solicita ao usuário um número n entre 0 e 37 (validação da entrada).

```

escreval("Por favor, digite um número entre 0 e 37 para exibir a sequência")
leia(n)

```

Processamento:

Inicializa os três primeiros termos da sequência:  $t_0 = 0$ ,  $t_1 = 0$ ,  $t_2 = 1$ .

Exibe os termos da sequência até o termo  $n$ :

Se  $n = 0$ , exibe apenas  $t_0$ .

Se  $n \geq 1$ , exibe  $t_0$  e  $t_1$ .

Se  $n \geq 2$ , exibe  $t_2$ .

Para  $i$  de 3 até  $n$ , calcula o próximo termo  $t_3 = t_0 + t_1 + t_2$ , exibe  $t_3$  e atualiza os valores:  $t_0 \leftarrow t_1$ ,  $t_1 \leftarrow t_2$ ,  $t_2 \leftarrow t_3$ .

```
ate (n >= 0) e (n <= 37)

// Inicialização dos termos
t0 <- 0
t1 <- 0
t2 <- 1

// Exibição dos termos
se n = 0 entao
  escreval(t0)
senao
  escreval(t0)
  escreval(t1)

  se n >= 2 entao
    escreval(t2)

    para i de 3 ate n faca
      t3 <- t0 + t1 + t2
      escreval(t3)
      t0 <- t1
      t1 <- t2
      t2 <- t3
```

Saída de Dados:

A sequência de Tribonacci é listada na tela e é exibido uma imagem de finalização

```
escreval("=====")
escreval("      FIM DE SEQUENCIA DE TRIBONACCI      ")
escreval("=====")
```

## 16. Sequência de Fibonacci

Definição:

- Este algoritmo de Fibonacci é uma sequência numérica infinita em que cada termo a partir do terceiro é a soma dos dois termos anteriores

Sequência:

- 0,0,1....

Fórmula:

- termo recebe a entrada da variável "n", e atribuir o valor 1 as variáveis "a" e "b". Logo após será criado um laço de repetição com o contador "i" crescendo até o valor de "n", onde será escrito o valor da variável "a" para exibir a sequência de Fibonacci

Variáveis Utilizadas na Operação:

- n (inteiro): número do termo máximo até onde a sequência será calculada.
- i (inteiro): variável de controle para o laço de repetição
- a (inteiro): variável de soma
- b (inteiro): variável de soma
- Temp (inteiro): armazenamento de soma a+b

Entrada de Dados:

Solicita ao usuário a quantidade de sequências desejadas:

```
Escreval("Digite a quantidade de sequências desejada")
leia(n)
```

Processamento:

Recebe a entrada da variável "n", e atribui o valor 1 as variáveis "a" e "b".

Logo após será criado um laço de repetição com o contador "i" crescendo até o valor de "n", onde será escrito o valor da variável "a" para exibir a sequência de Fibonacci.

variável "temp" recebe valor de a + b com objetivo de armazenar a soma temporariamente.

"a" recebe "b" para armazenar o número inicial enquanto "b" recebe "temp" para próxima soma ser feita com o valor já acumulado.

```
a <- 1
b <- 1

para i de 1 ate n faca
    escreval(a, " ")
    temp <- a + b
    a <- b
    b <- temp
```

Saída de Dados:

A sequência de Fibonacci é listada na tela e é exibido uma imagem de finalização

```
escreval("=====")
escreval("      FIM DE SEQUENCIA DE FIBONACCI      ")
escreval("=====")
```