Estruturas Sequenciais

As estruturas sequencias na computação são aquelas executas uma após a outra, ou seja, na sequencia, de cima para baixo, na ordem em que foram criadas em um tempo finito.

No Portugol Studio, primeiramente declare-se as constantes e variáveis, depois normalmente há a entrada de dados, em seguida é realizado o processamento desses dados e, no final apresenta a saída de dados. Esses passos são sequenciais por serem executados sempre nesta ordem todas as vezes, independente dos valores de entrada serem diferentes em cada execução do programa.

A Figura 1 demonstra um exemplo de estrutura sequencial no Portugal Studio. Nesta situação deseja-se calcular e exibir a média ponderada de duas notas, sabendo que os pesos são 4 e 6 respectivamente.

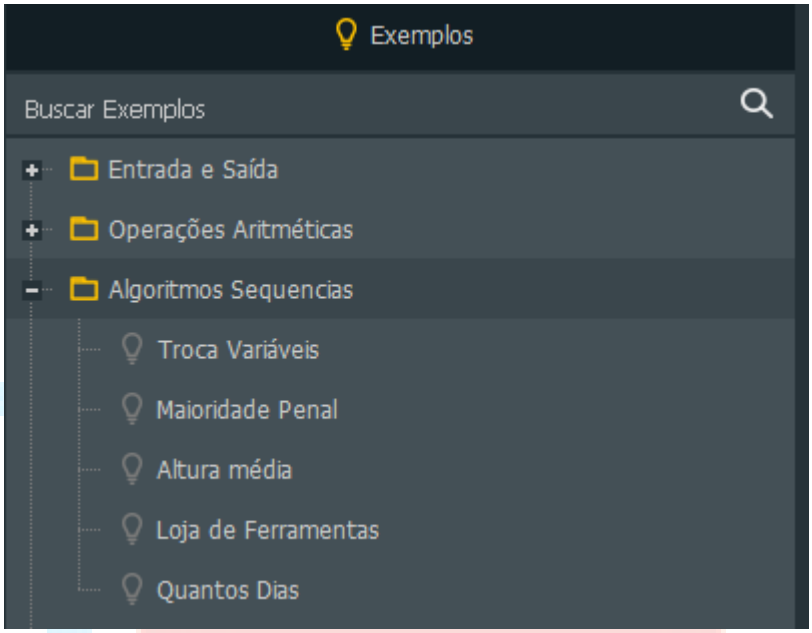
Figura 1: Exemplo de estrutura sequencial

```
programa
{
    funcao inicio()
    {
        // decalarações de constantes e variáveis
        const inteiro PESO1=4, PESO2=6

        real nota1, nota2, media

        // entrada de dados
        escreva("Digite a primeira nota: ")
        leia(nota1)
        escreva("\n")

        escreva("Digite a segunda nota: ")
        leia(nota2)
        escreva("\n")

        // cálculo da média (processamento)
        media=(nota1*PESO1+nota2*PESO2)/(PESO1+PESO2)

        // saída de dados
        escreva("Média ponderada: ",media)
    }
}
```

Fonte: Elaborado pela autora

O Portugol Studio ainda oferece vários exemplos de Algoritmos Sequenciais (Figura 2) para serem estudados para aprendizagem e modificados para se tornarem outros programas.

Figura 2: Algoritmos Sequenciais do Portugol Studio



Fonte: portugol-studio.jar, 2022

VOCÊ SABIA?

As pessoas também realizam estruturas sequencias todos os dias, pois quando acordam já têm uma rotina pela manhã que é sequencial. As rotinas podem ser diferentes para cada pessoa, mas são realizadas sequencialmente uma após a outra.

Estruturas de Controle

Muitas vezes, um programa de computador não segue somente um único caminho, ou seja, há situações que precisa seguir fluxos alternativos, de acordo com parâmetros diferentes em cada execução. Dessa forma, as estruturas de controle seguem as instruções de forma não linear.

Para alguns autores de algoritmos e/ou linguagens de programação,

consideram a estrutura linear como um tipo de estrutura de controle, pois independente se as instruções são executadas por um único caminho ou por caminhos diferentes, todas são controladas pelas regras da linguagem de programação em uso para execução do programa. Já outros autores, definem que estruturas de controle são basicamente de dois tipos:

- Estrutura de Seleção, também conhecida como Estrutura Condicional ou Desvio Condicional e Estrutura de Decisão
- Estrutura de Repetição ou Laços de Repetição

Estes últimos definem que se há um desvio na execução do programa, seja por um caminho ou outro, ou também por passar por um trecho várias vezes, o algoritmo deixar de ter uma estrutura sequencial para ter uma estrutura de controle.

Estruturas de Decisão Simples

Em computação também é necessário fazer escolhas, tomar decisões, como quando decide por uma rota ou outra para se chegar à algum lugar. Em lógica de programação, as estruturas de decisão, também chamadas de estruturas condicionais especificam as instruções que devem ser executadas de acordo com a condição analisada.

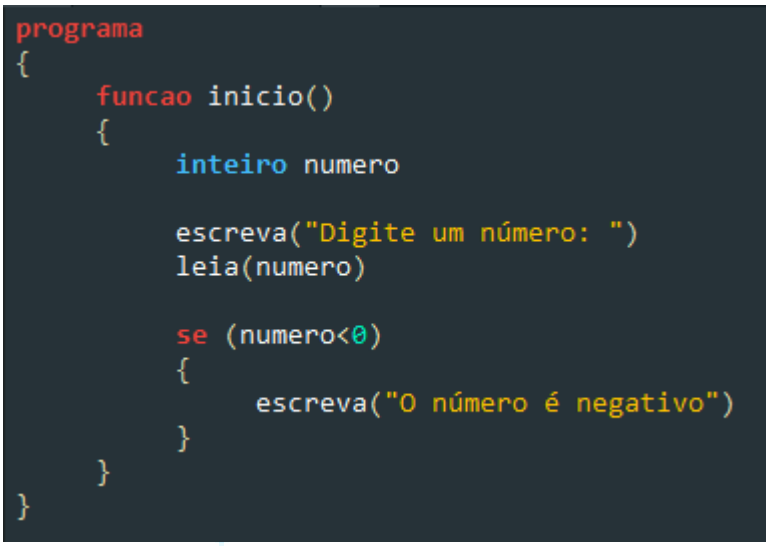
No Portugol Studio, as estruturas de decisão simples são aquelas que utiliza o comando "**se**" para verificar o teste lógico e prosseguir pelo caminho do resultado verdadeiro ou falso.

Uma das formas de utilizar esse comando é quando ele realiza somente as instruções seguintes quando o resultado da condição for verdadeira:

```
se (condição)
{
    instruções a serem executadas se verdadeiro
}
```

No exemplo da Figura 3, verifica-se se foi digitado um número negativo e exibe uma mensagem somente se essa condição for satisfeita.

Figura 3: Exemplo do comando "**se**"



```
programa
{
    funcao inicio()
    {
        inteiro numero

        escreva("Digite um número: ")
        leia(numero)

        se (numero<0)
        {
            escreva("O número é negativo")
        }
    }
}
```

Fonte: Elaborado pela autora

Como pode ser obervado na situação apresentada na Figura 3, caso o número digitado seja positivo, nenhum aviso é exibido para o usuário. Então, quando a condição for falsa e deseja-se executar outro conjunto de instruções, utiliza-se o comando "**se**" junto com o "**senão**":

```
se (condição)
{
    instruções a serem executadas se verdadeiro
}
senao
{
    instruções a serem executadas se falso
}
```

A Figura 4 demonstra o programa anterior modificado, informando se o número digitado pelo usuário é negativo ou positivo.

Figura 4: Exemplo do comando "**se-senao**"

```
programa
{
    funcao inicio()
    {
        inteiro numero

        escreva("Digite um número: ")
        leia(numero)

        se (numero<0)
        {
            escreva("O número é negativo")
        }
        senao
        {
            escreva("O número é positivo")
        }
    }
}
```

Fonte: Elaborado pela autora

Outra situação muito comum de acontecer é quando uma condição não é verdeira e precise verificar se outra condição é verdadeira. Para isso, no Portugol Studio, basta acrescentar o comando "**se**" imediatamente o comando "**senao**":

se (condição**)**
{
instruções a serem executadas se verdadeiro
}
senao se (condição**)**
{
instruções a serem executadas se falso para a condição anterior e se verdadeiro para esta condição
}

E ainda é possível acrescentar o comando "**senao**" para este "**senao se**", para no caso de não atender nenhuma das condições anteriores:

se (condição**)**

```
{
    instruções a serem executadas se verdadeiro
}
senao se (condição)
{
    instruções a serem executadas se falso para a
     condição anterior e se verdadeiro para esta condição
}
senao
{
    instruções a serem executadas se falso para a
     condição anterior e se falso para esta condição
}
```

A Figura 5 demonstra um exemplo no Portugol Stidio com várias condições para serem verificadas. Nesta situação, a partir da nota do atleta que é informada pela entrada de dados, verifica-se a sua classificação e exibe no Console do ambiente uma mensagem sobre o resultado, de acordo com uma tabela pré-definida a saber:

- nota menor que 5,0 está desclassificado;
- entre 5,0 e 7,0 classificado para a repescagem;
- entre 7,0 e 9,0 classificado para a semifinal;
- a partir de 9,0 classifificao para a final.

Figura 5: Exemplo do comando "**se-senao se**"

```
programa
{
    funcao inicio()
    {
        real nota

        escreva("Digite a nota obtida pelo atleta: ")
        leia(nota)

        se (nota>=9.0)
        {
            escreva("Excelente! Você está classificado para a Final.")
        }
        senao se (nota>=7.0)
        {
            escreva("Parabéns! Você está classificado para a Semifinal.")
        }
        senao se (nota>=5.0)
        {
            escreva("Muito bem! Você está classificado para Repescagem.")
        }
        senao
        {
            escreva("Que pena. Você não atingiu a nota mínima para continuar.")
        }
    }
}
```

Fonte: Elaborado pela autora

Como qualquer outra instrução, utilizar o comando "se" dentro de outro comando "**se**" é trivial na elaboração de programas para computadores.

A Figura 6 apresenta uma situação em que a idade é verifcada somente se o sexo for masculino. Assim, esta verificação (idade) é feita se a primeira condição for verdadeira. Caso contrário, com a entrada de qualquer outro caracter que não represente o sexo masculino, é apresentada uma mensagem de que está na equipe errada.

Ainda para melhorar o processamento desta situação, e leitura da idade poderia ser feita somente após a verificação do sexo (Figura 7), já que a idade interessa somente se esta condição for verdeira, ou seja, se o sexo for masculino.

Figura 6: Exemplo do comando "**se**" encadeado

```
programa
{
    funcao inicio()
    {
        caracter sexo
        inteiro idade

        escreva("Informe o sexo: ")
        leia(sexo)

        escreva("Informe a idade: ")
        leia(idade)

        se (sexo=='M' ou sexo=='m')
        {
            se (idade>=18)
            {
                escreva("Leberado para continuar")
            }
            senao
            {
                escreva("Ainda não pode participar")
            }
        }
        senao
        {
            escreva("Você não faz parte desta equipe")
        }

        escreva("\n")
    }
}
```

Fonte: Elaborado pela autora

Figura 7: Trecho do programa anterior com melhor processamento

```
        escreva("Informe o sexo: ")
        leia(sexo)

        se (sexo=='M' ou sexo=='m')
        {
            escreva("Informe a idade: ")
            leia(idade)
            se (idade>=18)
            {
                escreva("Leberado para continuar")
            }
            senao
```

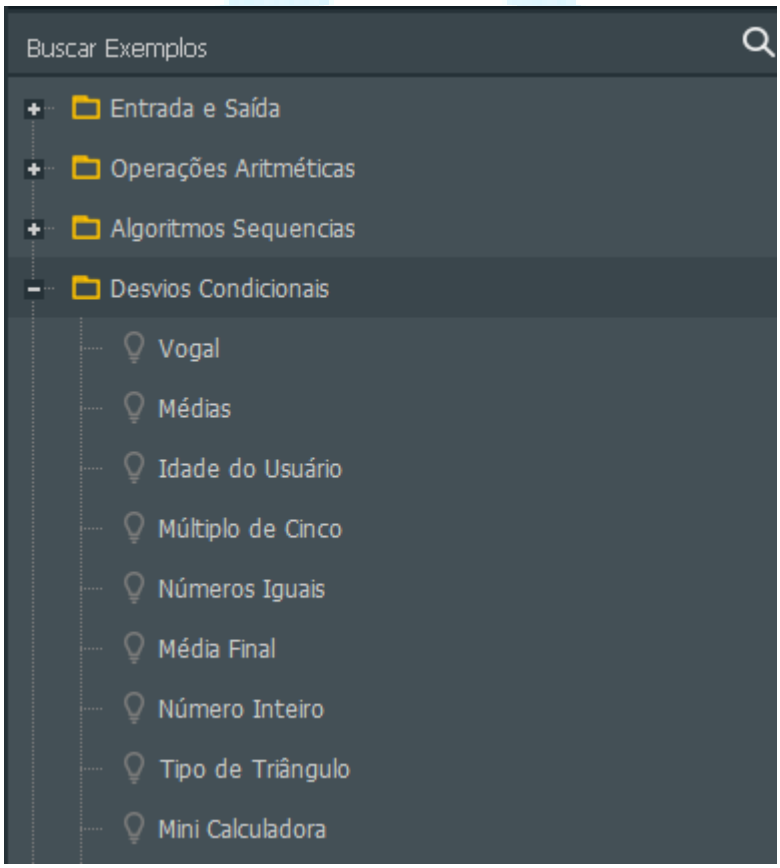
Fonte: Elaborado pela autora

VOCÊ SABIA?

O Portugol Studio diferencia caracteres minúsculos e maiúsculos. Por isso, no exemplo apresentado na Figura 6, foi necessário verificar ambos os casos para masculino (“M” e “m”).

O Portugol Studio também oferece vários exemplos sobre Desvios Condicionais (Figura 8) para serem estudados e modificados.

Figura 8: Desvios Condicionais do Portugol Studio (com “**se**”)



Fonte: portugol-studio.jar, 2022

Estruturas de Decisão Múltiplas

Quando há várias estruturas de decisão aninhadas são denominadas de estruras de decisão múltiplas.

Para esta situação, utiliza-se o comando "**escolha caso**", onde no "**escolha**" colocá-se a condição a ser testada e no "**caso**", o valor que pode ser resultado da condição. Também deve-se colocar o comando "**pare**" no final de cada "**caso**" para o Portugol Studio não continuar verificando os casos seguintes. E se ainda, o resultado não for encontrado, pode-se acrescentar o comando "**caso contrario**" (opcional) o qual executa as instruções se nenhum dos valores anteriores atender a condição.

```
escolha (variável)
{
    caso valor1
        instruções a serem executadas para o valor1
        pare
    caso valor2
        instruções a serem executadas para o valor2
        pare
    caso valorN
        instruções a serem executadas para o valorN
        pare
    caso contrario
        instruções a serem executadas para nenhum dos
        valores anteriores
}
```

Esse comando é muito utilizado na verificação das opçoes em um menu, conforme apresentado na Figura 9.

Figura 9: Exemplo do comando "**escolha caso**"

```
programa
{
    funcao inicio()
    {
        inteiro opcao

        escreva("1- Brasil \n")
        escreva("2- Argentina \n")
        escreva("3- Paraguai \n")
        escreva("4- Sair \n\n")

        escreva("Digite uma opção: ")
        leia(opcao)

        limpa()

        escolha (opcao)
        {
            caso 1:
                escreva ("Brasilia")
                pare
            caso 2:
                escreva ("Buenos Aires")
                pare
            caso 3:
                escreva ("Assunção")
                pare
            caso 4:
                escreva ("Programa encerrado.")
                pare
            caso contrario:
                escreva ("Opção Inválida.")
        }
        escreva("\n")
    }
}
```

Fonte: Elaborado pela autora

O comando "**se-senao se**" também pode ser utilizado para a criação de menu e substituir "**escolha caso**" (Figura 10). Porém, fica mais apresentável e elegante a organização do programa por meio deste último, reduzindo a complexidade do código.

Figura 10: Exemplo do comando "**se-senao se**" ao invés de "**escolha caso**"

```
programa
{
	funcao inicio()
	{
		inteiro opcao

		escreva("1- Brasil \n")
		escreva("2- Argentina \n")
		escreva("3- Paraguai \n")
		escreva("4- Sair \n\n")

		escreva("Digite uma opção: ")
		leia(opcao)

		limpa()

		se (opcao==1)
		{
			escreva ("Brasilia")
		}
		senao se (opcao==2)
		{
			escreva ("Buenos Aires")
		}
		senao se (opcao==3)
		{
			escreva ("Assunção")
		}
		senao se (opcao==4)
		{
			escreva ("Programa encerrado.")
		}
		senao
		{
			escreva ("Opção Inválida.")
		}

		escreva("\n")
	}
}
```

Fonte: Elaborado pela autora

Como observado na Figura 10, o código ainda fica organizado com o uso do "**se-senao se**", apesar de necessitar de mais chaves ({ e }). Mas a escrita do código ainda pode ficar pior se optar somente pelo comando "**se senao**" (Figura 11) , o qual precisa encadear um comando "**se**" dentro de outro para que instruções não sejam executadas sem necessidade.

Figura 11: Exemplo do comando "**se senao**" ao invés de "**escolha caso**"

```
programa
{
	funcao inicio()
	{
		inteiro opcao

		escreva("1- Brasil \n")
		escreva("2- Argentina \n")
		escreva("3- Paraguai \n")
		escreva("4- Sair \n\n")

		escreva("Digite uma opção: ")
		leia(opcao)

		limpa()

		se (opcao==1)
		{
			escreva ("Brasilia")
		}
		senao
		{
			se (opcao==2)
			{
				escreva ("Buenos Aires")
			}
			senao
			{
				se (opcao==3)
				{
					escreva ("Assunção")
				}
				senao
				{
					se (opcao==4)
					{
						escreva ("Programa encerrado.")
					}
					senao
					{
						escreva ("Opção Inválida.")
					}
				}
			}
		}

		escreva("\n")
	}
}
```

Fonte: Elaborado pela autora

O comando "**escolha caso**" apesar de ser similar ao "**se-senao se**", é utilizado para somente um tipo de desvio condicional, pois não é possível o uso de operadores lógicos, já que ele utiliza somente valores definidos como "**=**" em "**caso**" e "**!=**" em "**caso constrario**". Além disso, se não colocar "**pare**", o programa continua executando as instruções seguintes, aumentando o processamento.

VOCÊ SABIA?

O Portugol Studio possui um exemplo com o comando "**escolha caso**", que pode ser encontrado em Buscar Exemplos -> Desvios Condicionais -> Escolha-Caso.

Estruturas de Repetição com Variável de Controle

Quando necessita executar várias vezes o mesmo conjunto de instruções, utiliza-se estruturas de repetição para reduzir e simplificar o código de programação. Também conhecido como laço de repetição, é um tipo de estrutura que repete os comandos até a condição ser atendida.

A estrutura de repetição com váriável de controle, também chamada de Laço Para, define os valores inicial e final e o incremento ou decremento. Assim configura quantas vezes o conjunto de comandos da estrutura será executado. No Portugol Studio é o comando "**para**":

```
para (tipo variável=valor_inicial; condição; incremento/decremento)
{
    instruções a serem executadas enquanto a condição for verdadeira
}
```

VOCÊ SABIA?

No Portugol Studio há operadores próprios quando o incremento ou decremento é o valor 1 (um).

Símbolo	Operação	Exemplo
++	Incremento	var++ ⇔ var = var+1
--	Decremento	var-- ⇔ var = var-1

A Figura 12 apresenta um exemplo da estrutura de repetição com variável de controle em um programa que lê um certa quantidade (definida na constante) de notas e ao final apresente a média aritmética dessas notas. Ao definir o número de repetições na constante, facilita a alteração do programa para ser executado para qualquer quantidade de repetições desejada, pois basta apenas alterar o valor da constante.

Figura 12: Exemplo do comando "**para**"

```
programa
{
    funcao inicio()
    {
        const inteiro QTDE = 10

        real nota, soma=0.0, media

        para (inteiro i=1; i<=QTDE; i++)
        {
            escreva("Digite a "+i+"ª nota: ")
            leia(nota)
            soma += nota
        }

        media = soma / QTDE

        escreva("\n Média das notas: "+media)
    }
}
```

Fonte: Elaborado pela autora

VOCÊ SABIA?

O Portugol Studio possui operadores compostos de atribuição que simplificam o código nas expressões matemáticas, utilizando uma única variável na operação e recebimento do resultado.

Símbolo	Operação	Exemplo
+=	Adição	var += 1 ⇔ var = var+1
-=	Subtração	var -= 1 ⇔ var = var-1
*=	Multiplicação	var *= 1 ⇔ var = var*1
/=	Divisão	var /= 1 ⇔ var = var/1

Estruturas de Repetição Pré-Testado

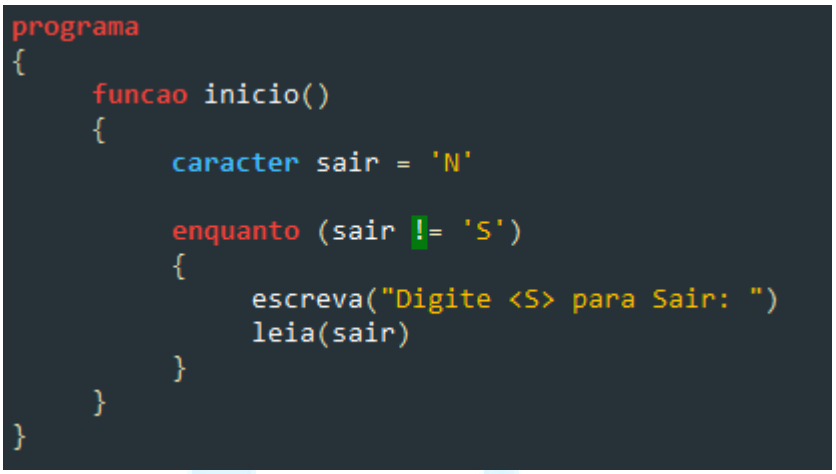
As estruturas de repetição pré-testado ou Laço Enquanto executa um conjunto de instruções enquanto a condição que é verificada, sempre no início do bloco, resultar verdadeiro. No Portugol Studio, utiliza-se o comando "**enquanto**":

```
enquanto (condição)
{
    instruções a serem executadas enquanto a condição for verdadeira
}
```

Nesta estrutura de repetição, diferente do comando "**para**", não se sabe de antemão o número de repetições que deverá ocorrer, pois é dentro do bloco que a condição deverá se tornar falsa para não ficar num eterno *loop forever*.

É muito utilizado na verificação de entrada de dados, como apresentado na Figura 13, onde o programa permanece em um ciclo enquanto o valor digitado não está dentro do esperado.

Figura 13: Exemplo do comando "**enquanto**"



```
programa
{
    funcao inicio()
    {
        caracter sair = 'N'

        enquanto (sair != 'S')
        {
            escreva("Digite <S> para Sair: ")
            leia(sair)
        }
    }
}
```

Fonte: Elaborado pela autora

Como a condição é verificada no início do conjunto de intruções que poderá se repetir, caso a expressão lógica resulte falso, todos os comandos dentro do bloco de repetição não serão executados, o que lhe dá o nome também de Loop Pré-testado.

Estruturas de Repetição Pós-Testado

A estrutura de repetição pós-testado ou Loop Pós-Testado, difere da entrutura anterior (enquanto) por executar pelo menos uma vez o conjuntos de instruções para repetição, já que a verificação da condição é no final do bloco.

No Portugol Studio Studio utliza-se o comando "faca enquanto":

faca
{
Instruções executadas pelo menos uma vez e enquanto a condição for verdadeira
} enquanto (condição**)**

Esse tipo de estrutura é muito útil da verificação dos dados lidos, solicitando novamente até que o valor informada esteja dentro do esperado, como pode ser observado no exemplo da Figura 14, o qual espera a

digitação de idade válida, ou seja idade maior que zero. Enquanto o valor digitado pelo usário for igual a zero ou negativo, o programa solicita novamente para digitar a idade.

Figura 14: Exemplo do comando "**faca enquanto**"

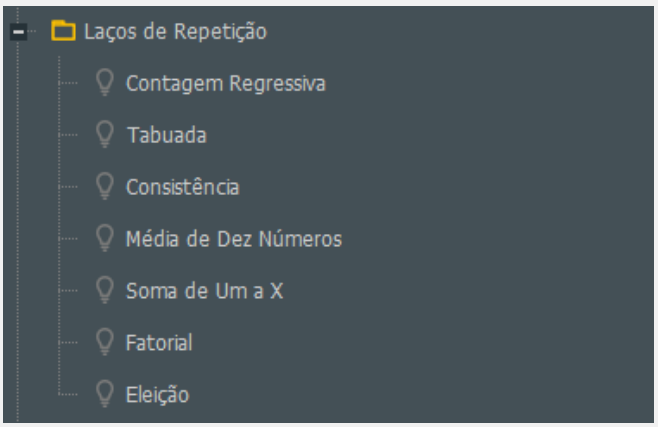
```
programa
{
    funcao inicio()
    {
        inteiro idade

        faca
        {
            escreva("Digite a idade: ")
            leia(idade)

        } enquanto (idade <= 0)
    }
}
```

Fonte: Elaborado pela autora

VOCÊ SABIA?

O Portugol Studio possui vários exemplos programas que utilizam as estruturas de repetição, encontrados em Exemplos -> Laços de Repetição (Figura 15).

Figura 15: Algoritmos de Laços de Repetição do Portugol Studio



Fonte: portugol-studio.jar, 2022

PARA PRATICAR

1. Ler um número de 1 até 12 representando um mês até que seja válido e exibir o nome desse mês.

2. Ler os coeficientes (a, b e c) para calcular as raízes da equação de 2º grau, conforme a fórmula de Bhaskara. O conteúdo de "a" deve ser diferente de zero. Exibir um dos resultados possíveis: "Não há raízes" ou "o valor de somente uma raiz" ou "os valores de duas raízes".

3. Ler vários números até que seja digitado zero. No final, exibir a soma de todos os números lidos.

4. Leia um número e exiba a sequência de Fibonacci (1, 1, 2, 3, 5, 8, 13, 21, 24, ...) até o número lido.

5. Leia um número e exiba uma mensagem identificando se o número lido é primo ou não.

REFERÊNCIAS

ALVES, Gustavo Furtado de Oliveira. **Estrutura de repetição ENQUANTO**. Disponível em: <https://dicasdeprogramacao.com.br/estrutura-de-repeticao-enquanto/> Acesso em 29 mar. 2022.

ALVES, Gustavo Furtado de Oliveira. **Estrutura de seleção múltipla ESCOLHA-CASO**. Disponível em: <https://dicasdeprogramacao.com.br/estrutura-de-selecao-multipla-escolha-caso/> Acesso em 27 mar. 2022.

BUENO, Gabriel. **Estrutura Sequencial**. Disponível em: <https://gabrielbueno072.github.io/rea-aed/aula_seq.html>. Acesso em 25 mar. 2022.

GASPAR, Wagner. **Estrutura de repetição FAÇA ENQUANTO em PORTUGO**. Wagner Gastar, 2021. Disponível em: <https://wagnergaspar.com/estrutura-de-repeticao-faca-enquanto-em-portugol/> Acesso em 30 mar. 2022.

Lógica de Programação. **O que é um Lação de Repetição?** Minutos de Leitura, 2021. Disponível em: <https://programadoresbrasil.com.br/2021/01/o-que-e-um-laco-de-repeticao/>. Acesso em 28 mar. 2022.

MACEDO, Hendrik. **Estruturas de Controle**. Disponível em: <https://cesad.ufs.br/ORBI/public/uploadCatalago/16455816022012Introducao_a_Ciencia_da_Computacao_Aula_4.pdf> Acesso em 26 mar. 2022.

PORTUGOL STUDIO. **Portugol Studio**. Disponível em: <http://lite.acad.univali.br/portugol/>. Acesso em 10 mar. 2022.

TERRA, Rafael. **O que são estruturas condicionais?** talentnetwork, 2021. Disponível em: <https://rockcontent.com/br/talent-blog/estruturas-condicionais-2/>. Acesso em 26 mar. 2022.