M06

Fundamentos de Programação

Prof. Romário Nzenguele da Silva

Licenciatura em Informática

2024 - 2025

Sobre o Documento

Material de apoio **gratuito**, desenvolvido exclusivamente para os estudantes do **1º Ano da Licenciatura em Informática**;

O conteúdo, sempre que necessário, será atualizado para refletir as mudanças e melhorias no decorrer do curso. Cada atualização será devidamente versionada, permitindo o fácil acompanhamento das revisões;

Este recurso é um **resumo das aulas**, essencial para apoiar no aprendizado da disciplina.

O **repositório da disciplina** (contendo exercícios, conteúdos detalhados, programa...) será reencaminhado por email;

Sugestões de melhoria podem ser apresentadas pelo email que se encontra no rodapé.

Conteúdo

6. Procedimentos (funções)

6.1. Sem retorno

6.2. Com retorno

6.3. Escopo de variáveis

Objectivos

- ✓ Compreender o conceito e a importância dos procedimentos;
- ✓ Conhecer os diferentes escopos de variáveis;
- ✓ Diferenciar e implementar procedimentos.

Conteúdo

7. Introdução à estruturas de dados

7.1. Vector

7.2. Matriz

Objectivos

- ✓ Compreender o conceito e o funcionamento de vectores/matrizes.

6. Procedimentos (funções)

- Aprende a reutilizar e organizar o seu código em blocos.

Suporte: Portugol (Portugol Studio)

6. Procedimentos (Funções)

❑ **Blocos de código que executam instruções específicas.**

- São definidos uma vez e podem ser chamados sempre que necessário, ou seja, permitem a reutilização do código.

6.1. Funções sem retorno

❑ **Sintaxe:**

```
//sem parâmetros
funcao <nome>(){
        <instruções>
}
```

```
//com parâmetros
funcao <nome>(<parâmetros>){
        <instruções>
}
```

6. Procedimentos (Funções)

6.2. Funções com retorno

- **Sintaxe:**

```
//sem parâmetros
funcao  <tipo_dados>  <nome>(){
        <instruções>
        return <valor>
}
```

```
//com parâmetros
funcao <nome>(<parâmetros>){
        <instruções>
}
```

- **Chamada (utilização) de funções**

```
//sintaxe
<nome>() //sem parâmetros
<nome>(<argumentos>) //com parâmetros
```

Funções sem retorno

```
programa
{
    // Função sem argumentos
    funcao imprimirMensagem()
    {
        escreva("Hello world!\n")
    }

    // Função com argumentos
    funcao somarEImprimir(inteiro a, inteiro b)
    {
        inteiro resultado = a + b
        escreva("Soma: ", resultado, "\n")
    }

    // Função principal
    funcao inicio()
    {
        // Executando as funções
        imprimirMensagem()
        somarEImprimir(5, 7)
    }
}
```

Funções com retorno

```
programa
{
    // Função sem argumentos que retorna uma cadeia
     funcao cadeia imprimirMensagem()
    {
        retorne "Hello world!"
    }

    // Função com argumentos que retorna um inteiro
    funcao inteiro somar(inteiro a, inteiro b)
    {
        retorne a + b
    }

    // Função principal (ponto de entrada do programa)
    funcao inicio()
    {
        // Executando as funções
        cadeia msg = imprimirMensagem()
        inteiro soma = somar(5, 7)

        escreva("\n", msg, "\n")
        escreva("\nSoma = ", soma, "\n")
    }
}
```

Nota: o retorno pode ser atribuído a uma variável do mesmo tipo.

6. Procedimentos (Funções)

6.3 Escopo de Variáveis

O escopo de uma variável define onde ela é acessível.

- ✓ **Locais**: são acessíveis apenas dentro do método onde foram declaradas;
- ✓ **Globais**: são acessíveis em qualquer parte do programa.

```
programa
{
    // Variável global
    inteiro soma = 0 // Esta variável é acessível em qualquer parte do programa

    funcao calcularSoma()
    {
        // Variáveis Locais: não podem ser acessadas fora desta função

        inteiro a = 5 // variável local
        inteiro b = 10 // variável local

        // Acessando a variável global soma
        soma = a + b
        escreva("a + b = ", soma, "\n")
    }

    funcao inicio()
    {
        calcularSoma()

        // Acessando a variável global soma
        escreva("O dobro da soma = ", 2 * soma, "\n")
    }
}
```

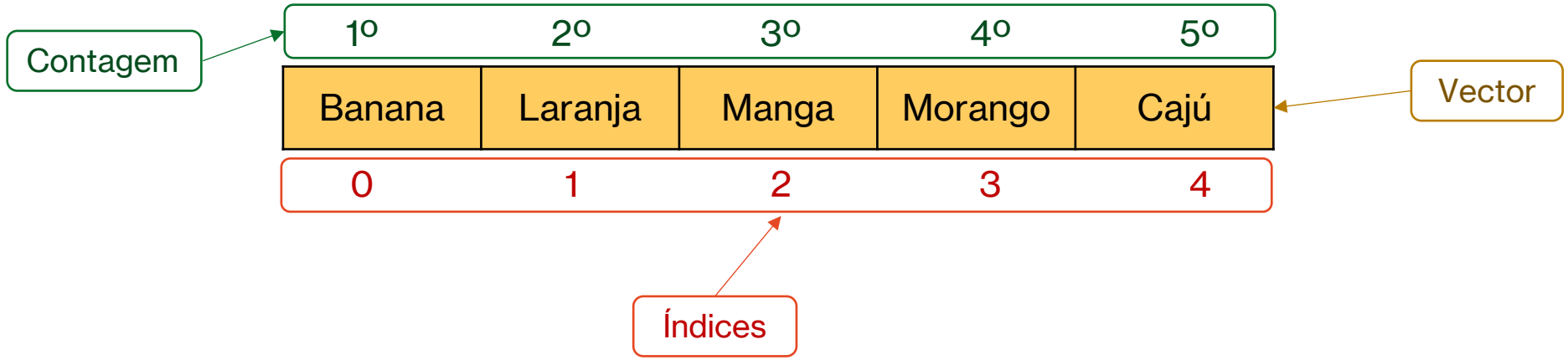
7. Introdução à estrutura de dados

Suporte: Portugol (Portugol Studio)

7. Introdução à estrutura de dados

7.1 Vetores

- Um vetor é uma coleção de elementos do mesmo tipo;
- É acessado por um índice. Os índices começam em 0.



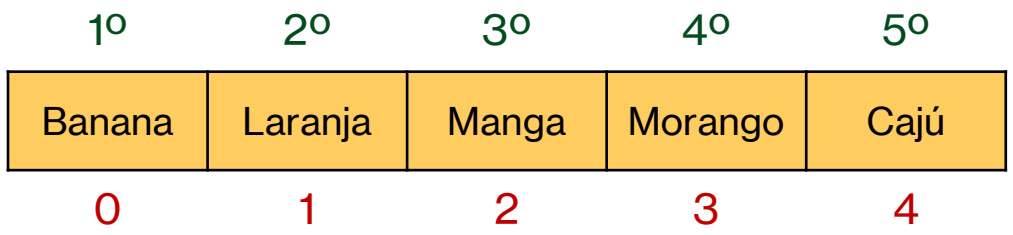
7. Introdução à estrutura de dados

7.1 Vetores

- **Sintaxe:**

```
//declaração
<tipo_dados>  <nome_vector>[ ] = {<valores>}

//acesso
 <nome_vector> [<índice>]
```




```
programa
{

    funcao inicio()
    {
        // Primeira forma - declaração e inicialização em uma linha
        cadeia frutas[] = {"Banana", "Laranja", "Manga", "Morango", "Caju"}

        // Segunda forma - declaração e inicialização separadas
        cadeia frutas2[5]
        frutas2[0] = "Banana"
        frutas2[1] = "Laranja"
        frutas2[2] = "Manga"
        frutas2[3] = "Morango"
        frutas2[4] = "Caju"

        // Exibindo a terceira fruta
        escreva(frutas[2])

    }
}
```

7. Introdução à estrutura de dados

7.1 Vetores

- **Percorrendo um vector:**

  Para percorrer um vector, precisamos de utilizar uma estrutura de repetição.

```
programa
{
    funcao inicio()
    {
        // Declaração e inicialização do vetor de frutas
        cadeia frutas[] = {"Banana", "Laranja", "Manga", "Morango", "Caju"}

        // Percorrendo o vetor e exibindo cada fruta
        para (inteiro i = 0; i <= 4; i++)
        {
            escreva(frutas[i], "\n")
        }
    }
}
```

7. Introdução à estrutura de dados

1.7.2 Matrizes

- Coleção de elementos apresentados em forma de linhas (i) e colunas (j);
- É um vector de vectores;
- O acesso é feito pelos índices da coluna e linha. Os índices começam em 0.

0	1	2	
Jogador	**País**	**Clube**	0
Vini jr.	Brasil	Real Madrid	1
Lamine Yamal	Espanha	Barcelona	2
Mike Maignan	França	AC Milan	3
Romário Faria	Brasil	Aposentado	4
Toni Kroos	Alemanha	Aposentado	5

j = índice-colunas

i = índice-linhas

7. Introdução à estrutura de dados

7.2 Matrizes

- **Sintaxe:**

//declaração
<tipo_dados> <nome_matriz> [ ] [ ]={{<valores>}}

//acesso
<nome_matriz >[<indice_linha>] [<indice_coluna>]

```
programa
{

	funcao inicio()
	{
		cadeia jogadores[][] = {
			{"Jogador", "País", "Clube"},
			{"Vini Jr.", "Brasil", "Real Madrid"},
			{"Lamine Yamal", "Espanha", "Barcelona"},
			{"Mike Maignan", "França", "AC Milan"},
			{"Romário Faria", "Brasil", "Aposentado"},
			{"Toni Kroos", "Alemanha", "Aposentado"}
		}

		//Exibindo 'Franca' => i = 3, j=1
		escreva(jogadores[3][1])
	}
}
```

7. Introdução à estrutura de dados

1.7.2 Matrizes

Percorrendo uma matriz:

Para percorrer uma matriz, utilizamos 2 estruturas de repetição. Uma para iterar as linhas (i) e a outra para iterar as colunas (j).

```
programa
{

	funcao inicio()
	{
		//Declaracao e inicializacao da matriz
		cadeia jogadores[][] = {
			{"Jogador", "País", "Clube"},
			{"Vini Jr.", "Brasil", "Real Madrid"},
			{"Lamine Yamal", "Espanha", "Barcelona"},
			{"Mike Maignan", "França", "AC Milan"},
			{"Romário Faria", "Brasil", "Aposentado"},
			{"Toni Kroos", "Alemanha", "Aposentado"}
		}

		// Percorrendo a matriz
		para (inteiro i = 0; i <= 5; i++){
			para (inteiro j = 0; j<=2; j++){
				escreva(jogadores[i][j], "  \t| ")
			}
			escreva("\n")
		}
	}
}
```

ANEXOS

Caderno de exercícios

Lista de exercícios para consolidar o aprendizado.

Consultar no repositório da turma.

...

Sequência -> M07