BEM-VINDOS

Residência de Software 2022/1

Disciplina: Lógica de Programação

Características Observadas.

- Dedicação
- Autonomia
- Participação
- Responsabilidade
- Iniciativa
- Cooperação
- Tomada de decisões
- Realização de tarefas variadas e complexas
- Identificação e resolução de problemas
- Adaptação às mudanças e ao trabalho em equipe
- Desenvolvimento de conhecimentos técnicostransferíveis.
- Empatia

Conteúdo Programático

- Algoritmo (Portugol / JavaScript)
    - Lógica booleana (E, OU, NOT)
    - Árvore de decisão
    - Estruturas de Laço
    - Conceito de recursividade
    - Estrutura de dados (Vetor, Matriz, Fila, Pilha...).
- Conceito de variável e constante
- Git

Algoritmo

- Sequência finita de passos que levam à execução de uma tarefa
- Algo muito comum no nosso dia a dia, sendo de TI ou não !

Ex: Trocar uma lâmpada, encontrar um tesouro a partir de pistas, fazer uma receita Culinária.

Algoritmo

- Algoritmo — Trocar a lâmpada

Passo 1— Pegar uma lâmpada nova;

Passo 2 — Pegar a escada;

Passo 3 — Posicionar a escada embaixo da lâmpada queimada;

Passo 4 — Subir na escada com a lâmpada nova;

Passo 5 — Retirar a lâmpada queimada;

Passo 6 — Colocar a lâmpada nova;

Passo 7 — Descer da escada;

- Passo 8 — Ligar o interruptor;
- Passo 9 — Guardar a escada;
- Passo 10 — Jogar a lâmpada velha no lixo..

Algoritmo

- **Receita de um bolo.**

Vocês devem escrever em um arquivo de texto as etapas de uma receita de bolo.

Dica : Lembre-se que uma receita geralmente é separada em ingredientes e modo de preparo

Algoritmo

• Como ficou a receita?

Início da Receita

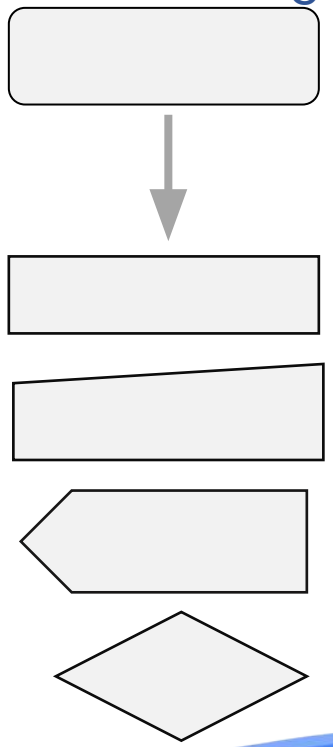
Ingredientes e Estruturas:

•Farinha = 3 **xícaras**
•Fermento = 1 **colher de sopa**
•Ovo = 3 unidades
•Manteiga_Massa = ½ **tablete**
•Manteiga_Untar = 1 colher de sopa (para untar)
•Laranja = 2 **xícaras**
•Açúcar = 3 **xícaras**
•Batedeira
•Forma_assar
•Forma_preparo

•Passo 0 : Define um tempo de preaquecimento
    - 15 minutos
•Passo 1 : Untar a forma
    - Manteiga
•Passo 2 : Bater na batedeira
    - Ovos, manteiga e açúcar
•Passo 3 : Juntar na batedeira partes secas
    - Farinha, fermento
•Passo 4 : Juntar ingredientes
    - Passo 2, Passo 3 e laranja (batedeira)
•Passo 5 : Por ingr. na forma e colocar no forno
    - Deixar no forno por 40 minutos
    - Forno = 180 ºC
    - Retirar do forno e furar massa com palito
        - Repete
            - Se massa estiver mole
                - Deixa mais 5 minutos
            - Senão
                - Termina o modo de preparo
•FIM DA RECEITA

Fluxograma - Outra forma de representação

É uma forma universal de representação, pois se utiliza de figuras geométricas para ilustrar passos a serem seguidos para a resolução de problemas.



Indica o início ou fim do algoritmo

Indica o sentido do fluxo de dados

Indica cálculos e atribuições de valores

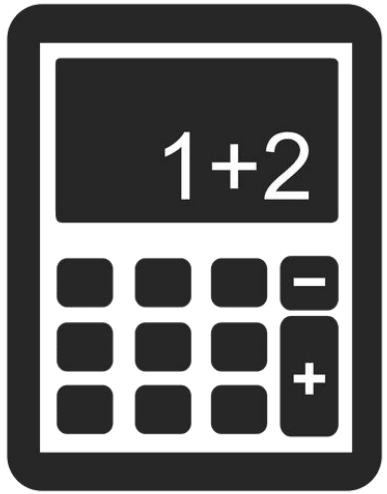
Representa a entrada de dados

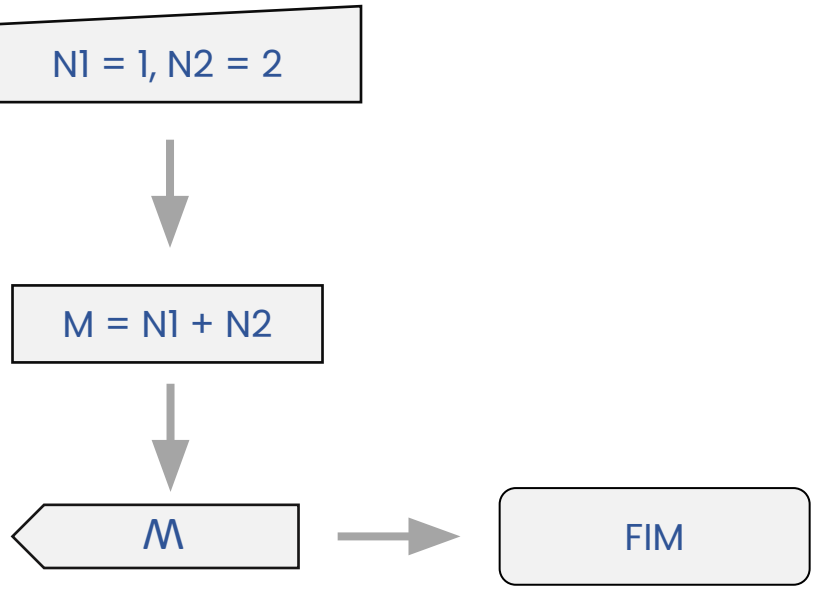
Representa a saída de dados

Tomada de decisão

Fluxograma

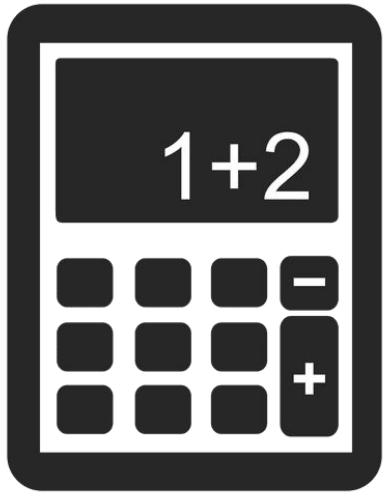
Exemplo : Soma de 2 números

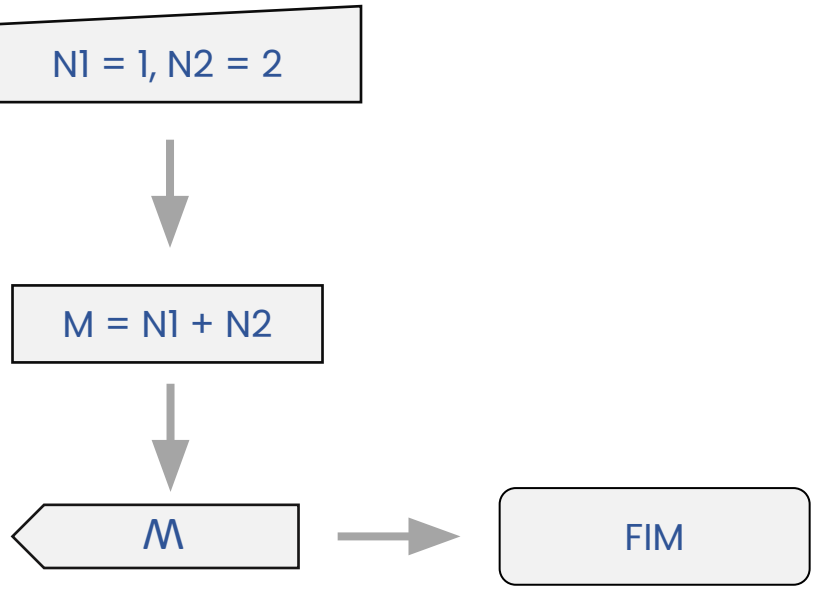




Fluxograma

Exemplo : Soma de 2 números





O que é um programa?

Programa

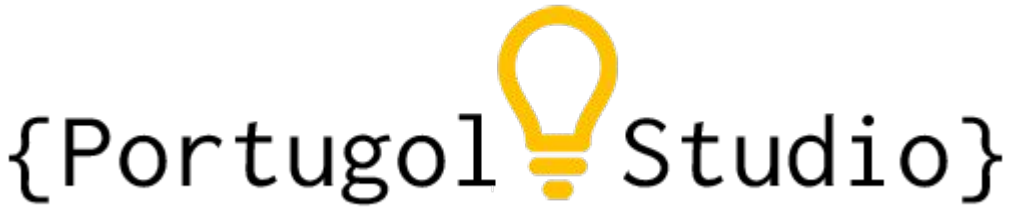
O que é um programa?

Algoritmo escrito em uma **linguagem** de programação

Linguagem

E qual linguagem usaremos neste curso?

Nesta disciplina o foco é entender os princípios da programação. Assim, utilizaremos a ferramenta Portugol Studio, que possui uma linguagem própria que aproxima a linguagem de programação ao português!



Portugol Studio – Estrutura do código.

Estrutura inicial de um código em Portugol

```
programa
{
	/* Declaração de variáveis, estruturas e outras funções */
	funcao inicio ()
	{
		/*Execução da função início*/
	}
}
```

Portugol Studio – Estrutura do código.

Estrutura inicial de um código em Portugol

```
programa
{
    /* Declaração de variáveis, estruturas e outras funções */
    Açucar meu_açucar = 3 xicaras
    Forno nosso_forno
    ...

    função passo_0(tipo forno) {
            // o que a gente faz aqui nessa parte : Preaquecer o forno
        }

    funcao inicio ()
    {
            passo_0(nosso_forno)
            ... //define outros passos
    }
}
```

Recapitulando

- Já aprendemos:
  - Valores
    - Como nos organizamos como turma
    - Valores em trabalho em equipe e desenvolvimento de software
  - Conteúdo
    - O que é um algoritmo
    - O que é um programa
    - Qual ferramenta utilizaremos
    - Representações de algoritmos
      - Pseudocódigo
      - Fluxograma

Próximos Passos

O que iremos aprender :

- Operações de entrada e saída
- O que são variáveis e constantes
- Desvios condicionais (se e senão)
- Operadores Lógicos (E, OU ...)
- Laços de repetição (enquanto, para ... até)
- Estruturas de dados (Vetores, Matrizes, Filas e Pilhas)
- Subrotinas (Funções)
  - Recursividade
  - Bibliotecas

Nosso primeiro programa: Olá mundo!

- Execute no Portugol Studio o código : Olá Mundo

  - O que esse código faz?
  - Quais dificuldades vocês tiveram em entender este trecho de código?

<HELLO WORLD />

Identificando-se : Qual é o seu nome?

- Execute os seguintes programas - ‘Numero Digitado’ e ‘Seu Nome’

  - O que esses códigos fazem?
  - A partir deles, escreva um novo programa que recebe seu nome e escreve o seu nome na tela

Outros Programas

Ficou parecido com a solução abaixo?

```
programa
{
    funcao inicio ()
    {
        cadeia nome   /*cadeia se refere ao tipo da variável que é uma cadeia de caracteres*/
        escreva("Digite seu nome: ")
        leia(nome)
        escreva("Seu nome é : ", nome ,“ \n”)
    }
}
```

Operações de entrada e saída

Por que entrada e saída?

Quando escrevemos :

```
cadeia nome
leia(nome)
```

leia é uma operação de **entrada** que permite que o que escrevemos no teclado seja **lido e armazenado** na variável **“nome”**. Logo estamos **entrando** com uma informação no programada durante sua execução.

Operações de entrada e saída

Quando escrevemos :

```
cadeia nome = “André”
escreva(“Meu nome é: “, nome)
```

escreva é uma operação de **saída** que permite que a informação escrita entre seus parênteses **“()”** seja apresentado na tela do computador, logo como é uma informação de apresentação, entendemos como uma informação de **saída**.

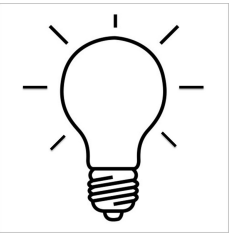
Variáveis e constantes

Voltando ao programa anterior...

```
programa
{
    funcao inicio ()
    {
        cadeia nome   /*cadeia se refere ao tipo da variável que é uma cadeia de caracteres*/

        escreva("Digite seu nome: ")
        leia(nome)
        escreva("Seu nome é : ", nome ,“ \n”)
    }
}
```

Ficou faltando entendermos o que são as **variáveis!**



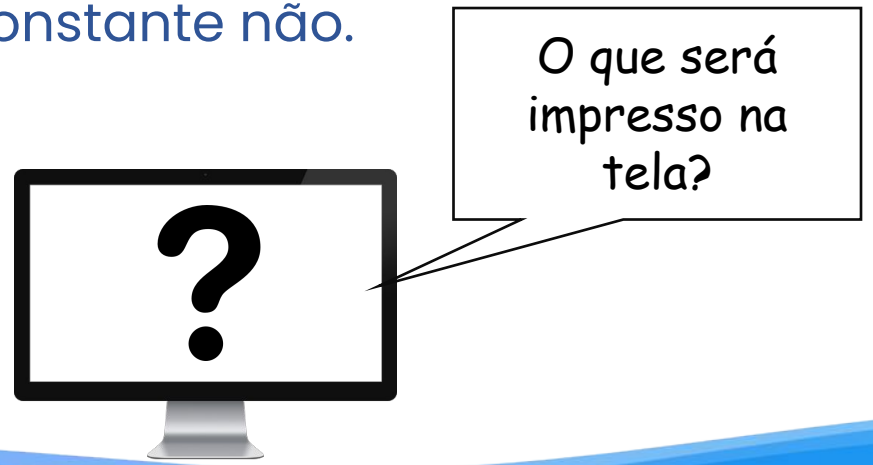
Variáveis e constantes

- Variáveis e constantes representam uma posição na memória, onde pode ser armazenado um único dado (valor) .

- Possuem tipo, nome e um valor.

- A diferença entre variáveis e constantes é que enquanto o valor da variável pode mudar durante a execução do programa o valor da constante não.

```
{
cadeia nome = “André”
nome = “Luiz”
escreva (nome)
}
```



Variáveis e constantes

- Em algumas linguagens (incluindo Portugol) as variáveis podem ser tipadas, ou seja, aceitam apenas valores referentes ao seu tipo, representado antes do nome da variável:

Tipos de variáveis na linguagem do Portugol Studio.

- inteiro : Número inteiros -> 1 ; 2 ; 3.
- real : Números de ponto flutuante -> 1.1 ; 3.14 ; 10.3
- cadeia : Cadeia de caracteres -> “Adoro estudar programação”
- caracter : Apenas um caractere -> ‘A’, ‘1’
- logico : Caractere booleano : verdadeiro, falso

Variáveis e constantes

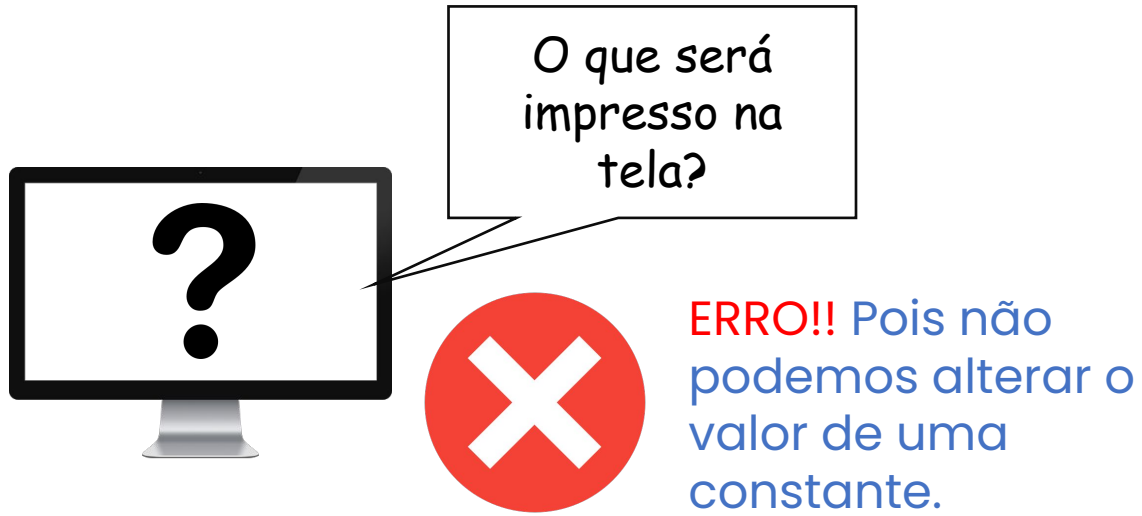
- Finalmente, para declarar uma constante basta colocar o indicador const antes da declaração da constante

```
{
const cadeia nome = “André”
nome = “Luiz”
escreva (nome)
}
```

Variáveis e constantes

- Finalmente, para declarar uma constante basta colocar o indicador const antes da declaração da constante

```
{
const cadeia nome = “André”
nome = “Luiz”
escreva (nome)
}
```



ERRO!! Pois não podemos alterar o valor de uma constante.

Recapitulando

- Já aprendemos:
  - Valores
    - Como nos organizamos como turma
    - Valores em trabalho em equipe e desenvolvimento de software
  - Conteúdo
    - O que é um algoritmo
    - O que é um programa
    - Qual ferramenta utilizaremos
    - Representações de algoritmos
    - Operações de entrada e saída
    - O que são variáveis e constantes

Próximos Passos

- Desvios condicionais (se e senão);
- Operadores lógicos (E, OU ...);
- Laços de repetição (enquanto);
- Estruturas de dados (Vetores, Matrizes, Filas e Pilhas);
- Subrotinas (Funções);
    - Recursividade;
    - Bibliotecas.

Desvios Condicionais

Lembram do exemplo de escrever e imprimir?

```
programa
{
    funcao inicio ()
    {
        cadeia nome
        escreva("Digite seu nome: ")
        leia(nome)
        escreva("Seu nome é : ", nome ,"\n")
    }
}
```

Vamos transformar este programa para ser capaz de **validar** o nome do usuário e conceder acesso ao sistema!
Apenas o usuário cadastrado poderá entrar no sistema.

Desvios Condicionais

Para resolver este problema podemos usar os condicionais **se** e **senao** (**if** e **else** em inglês).

```
programa
{
    funcao inicio () {
        cadeia nome
        escreva("Digite seu nome de usuário: ")
        leia(nome)
        se(nome == “André”) {
            escreva(“Bem vindo ", nome ,“ \n”)
            }
            senao {
                escreva(“Acesso negado!!! \n”)
            }
    }
}
```

Desvios Condicionais

Como vimos, podemos utilizar as cláusulas se e senão para direcionar a execução de nosso código. A estrutura consiste em basicamente :

```
se (condição) {
// Execute uma parte de código
}
senao {
    // Execute outra parte de código
}
```

Desvios Condicionais

Será que apenas o nome de usuário é suficiente?

Para validarmos corretamente um usuário precisamos também verificarmos se sua senha está correta. Assim precisamos validar o nome de usuário E sua senha.

```
programa
{
	funcao inicio () {
		cadeia nome
		escreva("Digite seu nome de usuário: ")
		leia(nome)
		se(nome == "Raul" e senha == "MinhaSenha") {   /*Note o operador lógico E para verificar o usuário E senha*/
			escreva("Bem vindo ", nome, "\n")
			}
			senao {
			escreva("Acesso negado!!! \n")
			}
	}
}
```

Operadores Lógicos

Podemos usar os operadores lógicos E , OU e NÃO (!) para melhorar ainda mais nossas condições.

Entendendo melhor os resultados dos operadores lógicos:

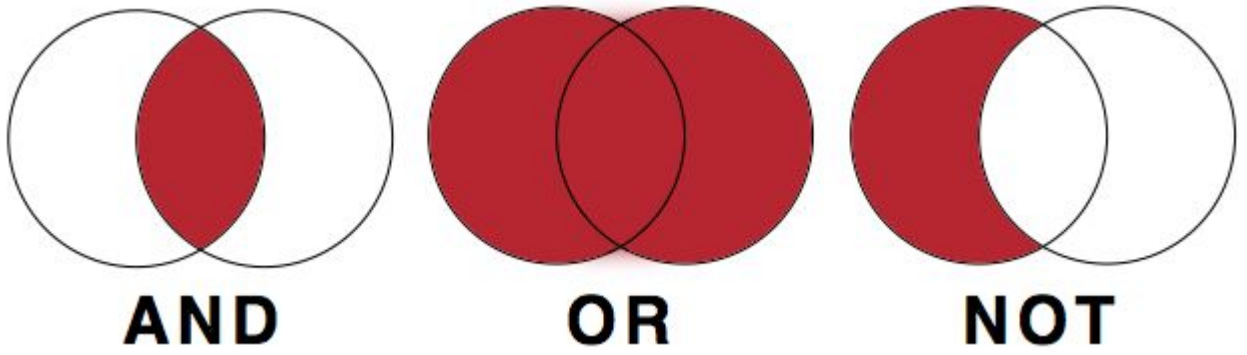
Verdadeiro E Verdadeiro = Verdadeiro
Verdadeiro E Falso = Falso
Falso E Falso = Falso
Verdadeiro OU Falso = Verdadeiro
Falso OU Falso = Falso
!Verdadeiro = Falso
!Falso = Verdadeiro
== (igual)
!= (diferente, ou seja não igual)

Operadores Lógicos

Na maioria das linguagens os operadores E , OU, e NÃO são representados por &&, || e ! , respectivamente.

Então :

E == && == AND
OU == || == OR
NAO == ! == NOT



Operadores Lógicos

A negação (!) pode ser utilizada na comparação de igual para negar uma igualdade

Exemplo:

1 == 1 (um igual a 1)

1 !=2 (um não igual a 2 || um diferente de 2)

Outro exemplo :

```
se(nome != “Fulano”) {  /**/
        escreva(“Você não é o Fulano \n”)
    }
    senao {
        escreva(“Olá Fulano!!! \n”)
    }
```

Recapitulando

- Já aprendemos:
  - Valores
    - Como nos organizamos como turma
    - Valores em trabalho em equipe e desenvolvimento de software
  - Conteúdo
    - O que é um algoritmo
    - O que é um programa
    - Qual ferramenta utilizaremos
    - Representações de algoritmos
    - Operações de entrada e saída
    - O que são variáveis e constantes
    - Desvios condicionais (se e senão);
    - Operadores lógicos (E, OU ...).

Próximos Passos

- Laços de repetição (enquanto);
- Estruturas de dados (Vetores, Matrizes, Filas e Pilhas);
- Subrotinas (Funções);
  - Recursividade;
  - Bibliotecas.