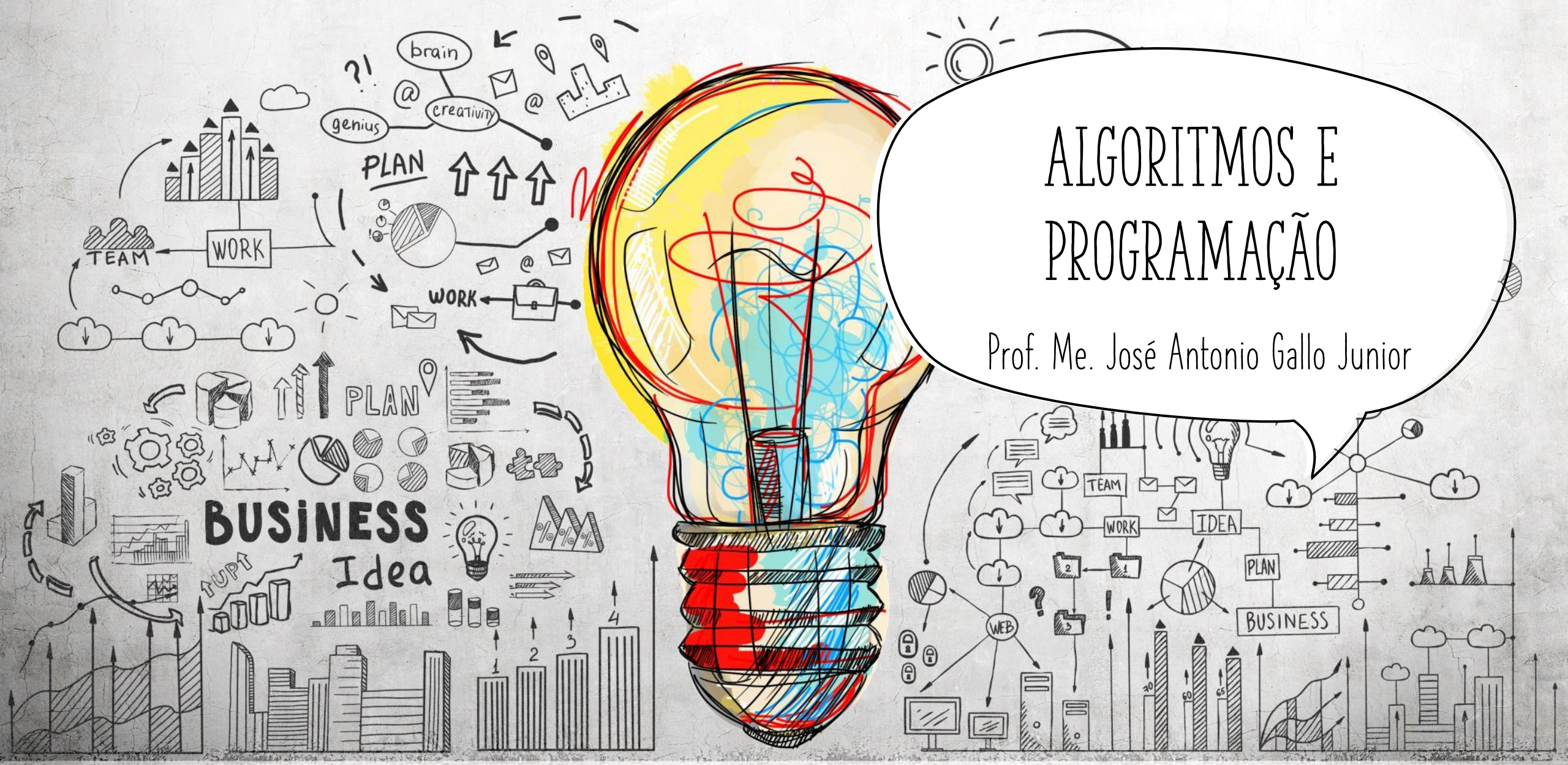

ALGORITMOS E PROGRAMAÇÃO

Prof. Me. José Antonio Gallo Junior

ESTRUTURAS DE CONTROLE

Em alguns casos, precisamos que o programa **tome decisões** ou **repita ações** com base nos dados recebidos.

Sem estruturas de controle, o programa executa os comandos na ordem em que aparecem, de cima para baixo.

As estruturas de controle permitem **mudar a ordem de execução** ou **repetir instruções**, tornando possível resolver problemas mais complexos, o que por sua vez, torna o programa mais inteligente e flexível.

ESTRUTURAS DE CONTROLE - DESVIOS CONDICIONAIS

Assim como na vida tomamos **decisões**, nos algoritmos também.

Um **desvio condicional** executa um bloco de código **apenas se** uma condição for **verdadeira**.

Exemplos:

- Um aluno só é aprovado se a média das notas for maior que 7.
- Um time só pode jogar em um campeonato se estiver classificado.

ESTRUTURAS DE CONTROLE - DESVIOS CONDICIONAIS

Principais tipos:

- se
- se-senao
- se-senao-se
- escolha-caso

DESVIOS CONDICIONAIS - SE

Usado quando queremos que algo aconteça apenas se uma **condição** for **verdadeira**.

Uma **condição** é um teste que pode dar **verdadeiro** ou **falso**.

Se for verdadeiro, o código dentro do **SE** é executado.

Se for falso, o programa pula esse bloco e continua normalmente.

Este desvio condicional é considerado **simples** e conhecido como o comando **SE**.

DESVIOS CONDICIONAIS - SE

Sintaxe:

```
se (condicao)
{
    //Instruções a serem executadas se o desvio for verdadeiro
}
```

DESVIOS CONDICIONAIS - SE

```
inteiro num
escreva("Digite um número: ")
leia(num)
se (num == 0) {
        escreva("O número digitado é 0")
}
```

DESVIOS CONDICIONAIS - SE-SENAO

Usado quando precisamos executar um código se a **condição** for **verdadeira** e outro se for **falsa**.

Sua sintaxe é simples, basta no termino do comando se ao lado do fechamento de chaves, colocar o comando senao e entre chaves colocar as instruções a serem executadas caso o comando se for falso

DESVIOS CONDICIONAIS - SE-SENAO

```
logico condicao = verdadeiro
se (condicao)
{
    //Instruções a serem executadas se o desvio for verdadeiro
}
senao
{
    //Instruções a serem executadas se o desvio for falso
}
```

DESVIOS CONDICIONAIS - SE

```
inteiro num
escreva("Digite um número: ")
leia(num)
se (num == 0)
{
        escreva("Impossível Dividir")
}
senao
{
        escreva("20 dividido por ", num, " = ", 20/num)
}
```

DESVIOS CONDICIONAIS - SE-SENÃO-SE

Usado quando temos mais de duas possibilidades.

O programa testa a primeira condição:

- Se for verdadeira, executa o bloco e para.
- Se for falsa, testa a próxima condição (senão se).
- Se nenhuma condição for verdadeira, executa o bloco final (senão).

A sua sintaxe é parecida com a do **senao**, mas usando o comando **se** imediatamente após escrever o comando **senao**.

DESVIOS CONDICIONAIS - SE-SENÃO-SE

```
se (condicao)
{
    //Instruções a serem executadas se o desvio for verdadeiro
}
senao se (condicao2)
{
    //Instruções a serem executadas se o desvio anterior for falso e este desvio verdadeiro
}
senao
{
    //Instruções a serem executadas se o desvio anterior for falso
}
```

DESVIOS CONDICIONAIS - SE-SENÃO-SE

```
se (nota >= 7)
{
    escreva("Aluno aprovado")
}
senao se (nota >= 4)
{
    escreva("Aluno em recuperação")
}
senao
{
    escreva("Aluno reprovado")
}
```

DESVIOS CONDICIONAIS - ESCOLHA-CASO

Usado para verificar o valor de uma variável e executar um código diferente para cada valor.

É mais organizado que usar vários **SE-SENÃO-SE**.

Funciona apenas com valores exatos (não aceita **operadores lógicos**).

Cada opção é um **caso**, e normalmente termina com o comando **pare** para evitar que outros casos sejam executados.

DESVIOS CONDICIONAIS - ESCOLHA-CASO

```
escolha (numero)
{
    caso 1:
        //Instruções caso o número for igual a 1
    pare
    caso 2:
        //Instruções caso o número for igual a 2
    pare
    caso contrario:
      //Instruções para caso não reconhecido
}
```

VAMOS PRATICAR!?!