

**Pratique,
aprenda,
conquiste.**

 **Proz**
Viva sua profissão!

Disciplina | Técnicas de Programação e Algoritmos

Técnico em Desenv. de Sistemas



**Aqui
começa
a sua
jornada**

Vamos nessa?



Disciplina | Técnicas de Programação e Algoritmos
Técnico em Desenv. de Sistemas



SUMÁRIO

TÉCNICAS DE PROGRAMAÇÃO E ALGORITMOS.....	4
INTRODUÇÃO.....	4
TEMA 01.....	5
Introdução a Lógica de Programação.....	5
TEMA 2.....	18
Fluxogramas e Condicionais.....	18
TEMA 03.....	45
Valores aleatórios e Laços de Repetição.....	45
TEMA 04.....	56
Menus e validações de dados.....	56
Tema 05.....	65
Vetores.....	65
TEMA 06.....	75
Matrizes.....	75
TEMA 07.....	85
Procedimento.....	85
TEMA 08.....	95
Técnicas e dicas de programação.....	95
TEMA 09.....	100
Conceitos básicos de Python.....	100
TEMA 10.....	124
Listas, tuplas e dicionários.....	124
TEMA 11.....	137
Condicionais.....	137
TEMA 12.....	142
Laços de repetição.....	142
TEMA 13:.....	149
Validação de dados.....	149
TEMA 14.....	152
Funções e Módulos.....	152

TÉCNICAS DE PROGRAMAÇÃO E ALGORITMOS



INTRODUÇÃO

A programação é uma competência crucial no mundo atual, com aplicações abrangentes em diversos campos. É uma linguagem universal que permite transmitir instruções precisas para um computador, criando uma infinidade de soluções práticas. Esta apostila se propõe a desmistificar o pensamento computacional e a linguagem de programação, fornecendo as ferramentas para entender e aplicar esses conceitos de maneira eficaz.

No início, apresentaremos uma visão geral do que é a programação, seguida por uma introdução ao funcionamento dos computadores. Isso permitirá uma compreensão ampla e profunda de como os programas interagem com o hardware e o sistema operacional. Ao longo do texto, serão abordadas as estruturas de controle de fluxo, tais como loops e condicionais, além de conceitos fundamentais como tipos de dados, variáveis e funções.

Para facilitar a assimilação do conteúdo teórico, cada capítulo contém exemplos práticos e exercícios. Estes são essenciais para consolidar o conhecimento adquirido e para desenvolver habilidades práticas. Recomenda-se que cada exercício seja concluído para garantir a máxima compreensão do conteúdo.

Vale ressaltar que a aprendizagem de programação não está restrita à memorização de sintaxe ou regras rígidas - é, acima de tudo, sobre aprender a pensar de uma nova maneira. Nesse sentido, a abordagem adotada aqui visa não apenas ensinar a escrever código, mas também desenvolver o pensamento lógico e a capacidade de resolver problemas.

Portanto, convidamos todos a embarcar nesta jornada de aprendizagem, a explorar cada tópico, a resolver cada exercício e a desenvolver habilidades valiosas para o mundo atual. Esta jornada certamente será desafiadora, mas também será recompensadora e estimulante.

TEMA 01

Introdução a Lógica de Programação

Habilidades:

- Elaborar algoritmos.

História e Principais Conceitos



Vídeo: O que é Programação de Computadores: Fonte: <https://youtu.be/qyfFw6oWuTk>

Disponível em <<https://tinyurl.com/36wxaphva>>. Acesso em: 10 jul. 2023.

Uma linguagem de programação é um conjunto de instruções que dizem a um computador o que fazer. É como se fosse uma forma sofisticada e precisa de dizer ao computador quais tarefas realizar. Com essa linguagem, os programadores podem definir dados importantes, como transmiti-los, como receber informações, como processá-las e qual resultado esperar.

A linguagem de programação usa uma sequência lógica para funcionar. O desafio dos programadores é criar uma forma de escrever o código que seja fácil de entender, intuitiva e organizada. Nas décadas de 1940, quando os primeiros computadores surgiram, as pessoas

precisavam criar linguagens especiais para se comunicar com eles, porque naquela época não existia memória suficiente. Essas primeiras linguagens eram complexas e chamadas de "linguagens de máquina".

Assembly

Inicialmente, a programação era feita em uma linguagem chamada assembly, e embora fosse possível programar computadores grandes, era muito complicado. Outras linguagens fáceis de usar como ALGOL, FORTRAN e COBOL surgiram na década de 1950 e ainda estão em uso hoje. Paralelamente, também foram desenvolvidos o Lips e o Proglog, importantes para pesquisas na área de inteligência artificial. Depois vieram a linguagem Simula 67, que introduziu o conceito de classes na década de 1970, e Smalltalk, a primeira linguagem totalmente orientada a objetos que aprofundou esse conceito. Mais tarde, a linguagem popularizou as classes C++.

O Conceito de Linguagem de Programação

Como vimos, a linguagem de programação teve uma evolução gradativa, acompanhando a evolução dos computadores. A partir dos processadores e da expansão da memória, novas linguagens de programação foram surgindo, e desta forma, a linguagem contribuiu para a expansão da evolução humana, transformando o raciocínio lógico em algo palpável e produtivo, e desta forma, tornando a informática algo fundamental para a sociedade moderna.

Principais Linguagens de programação



Existem milhares de linguagens de programação, dentre delas, apresentaremos as principais linguagens de programação do mercado:

Linguagens Antigas	Linguagens mais Modernas	Linguagens para Web
ALGOL	C#	JavaScript
Assembly	C++	Java
COBOL	Delphi	PHP

FORTRAN	MATLAB	Perl
Object Pascal	Objective-C	Python
PL/SQL	Ruby	Visual Basic .NET
Portugol	Swift	
Visual Basic		

Fonte: Elaborado pelo autor (2023). Fonte Vídeo: <https://youtu.be/z1XTcKKRbKM>

Níveis de Programação

Podemos classificar a linguagem de programação em 2 níveis:

Linguagem de Programação de baixo nível

- O primeiro é a linguagem de baixo nível, esta linguagem trabalha no nível da arquitetura do hardware, nível de programação trabalha diretamente com os registradores e processadores e chegam a utilizar editores de base hexadecimal. Exemplos Linguagem de Montagem (Assembly)

Linguagem de Programação de alto nível

- A linguagem de alto nível é uma linguagem mais estruturada, que não trabalha no nível da arquitetura do computador, este formato trabalha com códigos fontes, convertendo o código para código de máquina utilizando os interpretadores e compiladores. Este tipo de linguagem se assemelha com a linguagem humana. Exemplos: C,Java,Python,JavaScript,PHP.

Interpretadores e Compiladores

Todo código de programação pode ser traduzido em código de máquina por um compilador ou interpretador por um interpretador. Essa conversão transforma o código-fonte em código de máquina. Um compilador basicamente traduz um programa em código de máquina. Quando um programa é compilado, ele não está mais na forma "editada", mas sim em uma que já existe no sistema onde o programa está inserido.

A cada fase ou atualização, o programa deve realizar uma nova compilação. Cada linguagem requer um tipo correspondente de interpretador e compilador. Você pode criar vários programas dessa maneira.

Classificação das Linguagens de Programação

Existem diferentes tipos de linguagens de programação, cada uma com características específicas. A ACM (Association for Computing Machinery) classifica as linguagens de programação em vários tipos, buscando padronizar seu uso e suas aplicações, são elas:

- Linguagens aplicativas, ou de aplicação: Usadas para criar programas e aplicativos.
- Linguagens concorrentes, distribuídas e paralelas: Lidam com a execução simultânea de tarefas em diferentes computadores.
- Linguagens de altíssimo nível: Mais próximas da linguagem humana e fáceis de entender.
- Linguagens de aplicação especializada: Desenvolvidas para resolver problemas específicos em áreas específicas.
- Linguagens de fluxo de dados: Organizam o processamento e a sequência de dados em programas.
- Linguagens de microprogramação: Controlam diretamente os circuitos eletrônicos de um computador.
- Linguagens de montagem e de macro: Permitem programar em nível mais baixo, próximo à linguagem de máquina.
- Linguagens de projeto: Usadas para descrever e planejar estruturas de software.
- Linguagens extensíveis: Podem ser expandidas com novas funcionalidades.
- Linguagens não determinísticas: Resultados diferentes em diferentes execuções.
- Linguagens não procedurais: Baseadas em regras e restrições lógicas.
- Linguagens orientadas a objeto: Organizam programas em objetos interativos.

Conceito de Programação

Apesar de todas as linguagens de programação em teoria alcançar o mesmo objetivo lógico, existem conceitos para cada tipo de programação, estruturamos as linguagens em categorias maiores e gerais para organizar seus conceitos, são elas:

- Programação Estruturada
- Programação Linear
- Programação Modular
- Programação Orientada a Eventos
- Programação Orientada a Objeto

Programação Estruturada

A programação estruturada foi a precursora da estruturação da linguagem de programação, ela divide em três partes:

Diagrama – Programação Estruturada



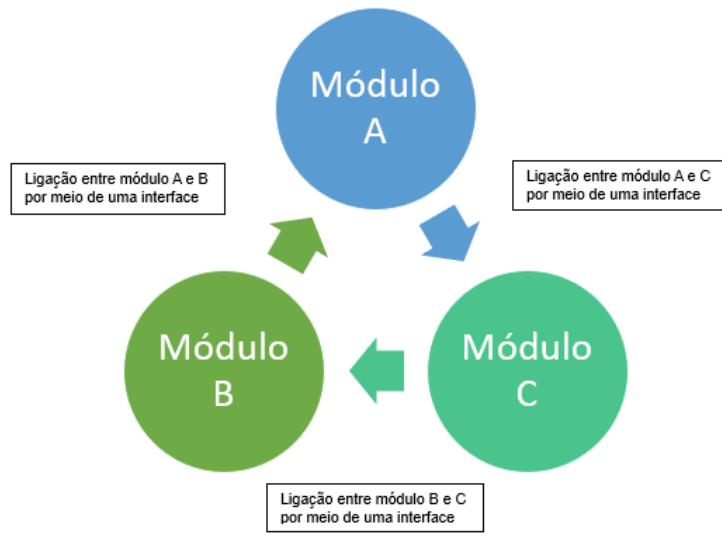
Fonte: Elaborado pelo autor (2023).

Este conceito orientava os programadores a usarem códigos simples, usando funções e subrotinas em sua criação.

Programação modular

A programação modular apresenta o conceito de estruturar um programa por módulos relacionados e interdependentes através de uma interface em comum. Este tipo de linguagem rapidamente substituiu a linguagem estruturada, pois ela apresenta algo mais robusto e confiável.

Diagrama – Programação Estruturada

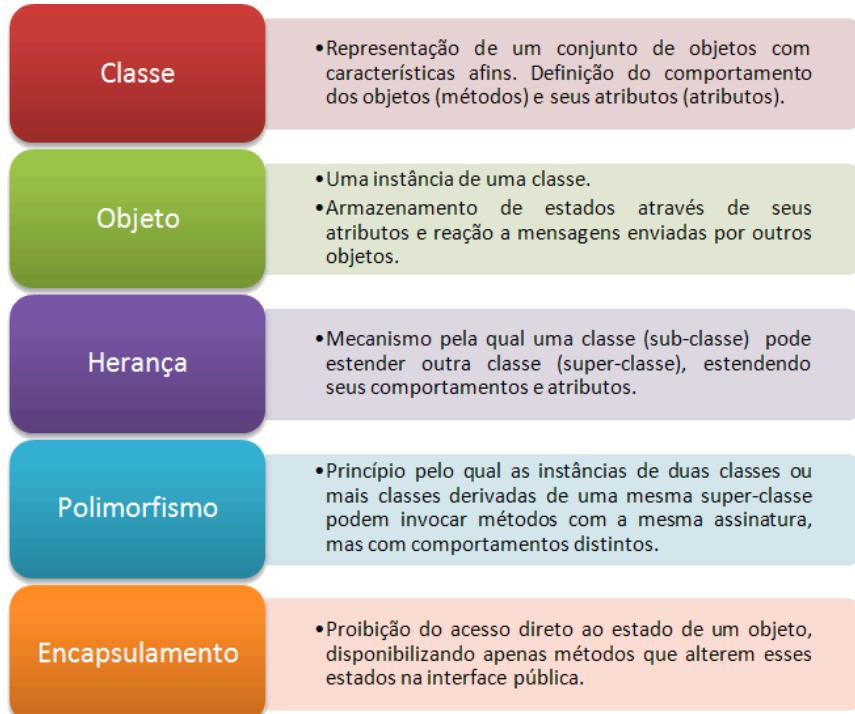


Fonte: Elaborado pelo autor (2023).

Programação Orientada a Objeto

A programação orientada a objeto é uma abordagem que permite estruturar o código em objetos, que possuem características (atributos) e ações (métodos). Isso facilita a reutilização de código e a criação de interações entre os objetos. Essa forma de programação é amplamente utilizada na criação de inteligência artificial e é baseada em cinco princípios fundamentais.

Diagrama 3 – Programação Orientada a Objeto/ Fonte PHPH



Fonte: *Elaborado pelo autor (2023)*.

Programação Linear

A programação linear é focada em otimizações e soluções de problemas de programação, este tipo de programação utiliza uma estrutura matemática para buscar resoluções. Historicamente a programação linear apresenta resoluções como dualidade, decomposição, e a importância da convexidade e suas generalizações.

Programação Orientada a Eventos

Diferente de programas normais que seguem um fluxo de controle padronizado, os controles de fluxo de programas orientados a eventos são guiados por ações externas, chamadas eventos. Sua aplicação é grande no desenvolvimento de sistemas de interface com o usuário.

Conceito de Lógica

A lógica é um conceito que tem se expandido desde os princípios da humanidade, desde a Grécia antiga se tem a lógica uma ferramenta em favor do exercício do pensamento e da linguagem. Trazendo para a realidade da informática, a lógica é um conjunto ordenado de instruções visando alcançar um objetivo, sistematizando todas as suas etapas em sequência, a fim de organizar um pensamento, e desta forma, alcançar a solução de uma problemática.

Linguagens Web

As linguagens de programação web são usadas para criar páginas e sistemas que funcionam na internet. Elas permitem criar a estrutura que permite a interação dos usuários com bancos de dados. As principais linguagens para desenvolvimento web são:

- HTML: para criar a estrutura e o conteúdo das páginas.
- CSS: para estilizar as páginas, definindo cores, fontes e layout.
- JavaScript: para adicionar interatividade às páginas, como animações e validação de formulários.

Neste material, vamos começar com problemas simples que envolvem raciocínio lógico e que têm soluções flexíveis, para ajudar você a entender os passos básicos para resolver problemas. É importante lembrar que o desenvolvimento de software segue um processo bem definido e controlado. Esta apostila se concentra em ensinar conceitos básicos que ajudam a especificar corretamente o software e a desenvolvê-lo.



Fonte do vídeo: https://youtube.com/playlist?list=PLHz_AreHm4dmSi0MHol_aoNYCSGFqvfXV

Introdução a Lógica de Programação

A lógica é uma forma de pensamento que nos ajuda a resolver problemas de maneira eficiente. Tem raízes na Matemática e na Filosofia, e remonta à Grécia antiga, com Aristóteles sendo um dos principais estudiosos. Ele desenvolveu a lógica formal e material, que se concentra na estrutura do raciocínio e nas operações do pensamento.

Existem muitas linguagens de programação hoje em dia, como C, C++, Java, JavaScript, Python, PHP, entre outras. No início da era dos computadores, os programadores precisavam escrever em linguagem de máquina, que usava números binários. Porém, isso era muito difícil, então foi criada uma linguagem chamada Assembly, que era mais próxima da linguagem humana. O Assembly traduzia as instruções binárias em palavras ou abreviações conhecidas como mnemônicos.

Segue abaixo um exemplo de programa em Assembly que exibe a frase "Hello World" na tela:

```
section .data
    helloMsg db 'Hello World', 0

section .text
    global _start

_start:
    ; Escreve a mensagem na saída padrão
    mov eax, 4
    mov ebx, 1
    mov ecx, helloMsg
    mov edx, 11
    int 0x80

    ; Termina o programa
    mov eax, 1
    xor ebx, ebx
    int 0x80
```

Fonte: Elaborado pelo autor (2023).

Com o tempo, foram desenvolvidas linguagens de programação mais próximas da linguagem humana, chamadas de linguagens de alto nível. Alguns exemplos são Pascal, BASIC e C.

Com o passar do tempo foi criada uma pseudo linguagem utilizada para ensinar lógica de programação em países de língua portuguesa, o Portugol. Existem interpretadores como o VISUALG 3.0, Portugol Studio e G-Portugol que ajudam a executar programas escritos nessa pseudolínguagem. São a partir deles que iniciamos o processo para programar o código fonte.

O código-fonte é o texto escrito em uma linguagem de programação e pode ser lido por pessoas, mesmo sem conhecimento em programação. Esse texto segue as regras da linguagem, como a sintaxe dos comandos e funções e conseguimos criar os programas executáveis.

Os Programas executáveis são sequências de instruções em linguagem de máquina, compreensíveis pelo computador, mas difíceis para seres humanos entenderem. Esses programas podem ser criados através de interpretação ou compilação. Na interpretação, o código-fonte é convertido em linguagem de máquina durante a execução do programa, enquanto na compilação, o código-fonte é convertido completamente em linguagem de máquina antes da execução. As instruções são escritas por meio de algoritmos.

Um algoritmo é um conjunto de instruções, regras ou comandos utilizados para resolver problemas. A lógica é usada para encontrar soluções para esses problemas.

Introdução ao VISUALG

O Portugol é um pseudocódigo, é uma forma de expressar o pensamento lógico, principalmente para quem não fala inglês. O VISUALG como mencionado anteriormente é um interpretador que utilizaremos para executar a pseudo linguagem chamada de Portugol. Para utilizá-lo basta baixá-lo da internet, não precisa fazer nenhuma instalação.

Abaixo segue a tela inicial do VISUALG, note que todo novo arquivo já começa previamente estruturado.

The screenshot shows the VISUALG 3.0.7.0 software interface. The title bar reads "VISUALG 3.0.7.0 * Interpretador e Editor de Algoritmos * última atualização: 03 de Outubro de 2015 * CEDUPHI - Centro de Educação Profissional". The menu bar includes Arquivo, Editar, Run (executar), Exportar para, Manutenção, and Help (Ajuda). Below the menu is a toolbar with various icons. The main area is titled "Área dos algoritmos (Edição do código fonte) -> Nome do arquivo: [semnome]". It contains the following pseudocode:

```
1 Algoritmo "semnome"
2 // Disciplina : [Linguagem e Lógica de Programação]
3 // Professor : Nome do(a) professor(a)
4 // Descrição : Aqui você descreve o que o programa faz! (função)
5 // Autor(a) : Nome do(a) aluno(a)
6 // Data atual : dd/mm/aaaa
7 Var
8 // Seção de Declarações das variáveis
9
10
11 Inicio
12 // Seção de Comandos, procedimento, funções, operadores, etc...
13
14
15 Fimalgoritmo
```

At the bottom of the code area, there is a status bar with the text "Permite CRIAR, ALTERAR, EXCLUIR, CONSULTAR e VISUALIZAR o código fonte escrito em V".

Fonte: Elaborado pelo autor (2023).

Na tela acima, temos algumas instruções importantes:

- O campo "semnome" deve ser preenchido com o nome do algoritmo que estamos criando.
- É necessário preencher os outros campos (Disciplina, Professor, Descrição, Autor, Data atual) para documentar corretamente o programa.
 - Para adicionar comentários ao código, usamos "//". Os comentários aparecem em verde e servem para explicar partes do programa.
 - É bom sempre adicionar comentários para facilitar a leitura do código no futuro.
 - A parte onde está escrito "Var" é onde declaramos as variáveis do programa.
 - Entre as palavras "Inicio" e "Fimalgoritmo", escrevemos a lógica do programa. Essas palavras indicam o início e o fim do algoritmo no VisuALG.

Exemplo 1: Utilizando o comando “escreva”

- Passo 1: Conforme o exemplo abaixo, na linha 1 após a tag Algoritmo escreva “Saudação”
- Passo 2: Utilize o comando escreva (“Olá Mundo!”) após a tag Início, conforme a linha 13.

```

1 Algoritmo "Saudacao"
2 // Disciplina : [Linguagem e Lógica de Programação]
3 // Professor : Nome do professor(a)
4 // Descrição : Aqui você descreve o que o programa faz! (função)
5 // Autor(a) : Nome do(a) aluno(a)
6 // Data atual : dd/mm/aaaa
7 Var
8 // Seção de Declarações das variáveis
9
10 Inicio
11 // Seção de Comandos, procedimento, funções, operadores, etc...
12
13 escreva ("Olá Mundo!")
14
15 Fimalgoritmo

```

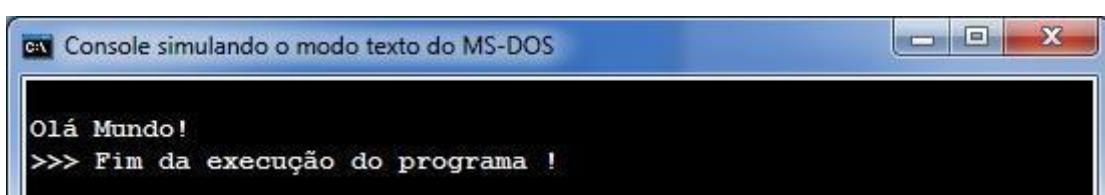
Fonte: Elaborado pelo autor (2023).

- Passo 3: Para executar o algoritmo vá em RUN => Rodar o Algoritmo ou pressione a tecla F9 do teclado.



Fonte: Elaborado pelo autor (2023).

Abaixo segue a tela de execução:



Fonte: Elaborado pelo autor (2023).

Exemplo 2: Utilizando o comando “escreval”

```
1 Algoritmo "Saudacao"
2 // Disciplina : [Linguagem e Lógica de Programação]
3 // Professor : Nome do professor(a)
4 // Descrição : Aqui você descreve o que o programa faz! (função)
5 // Autor(a) : Nome do(a) aluno(a)
6 // Data atual : dd/mm/aaaa
7 Var
8 // Seção de Declarações das variáveis
9
10 Inicio
11 // Seção de Comandos, procedimento, funções, operadores, etc...
12
13 escreval ("Olá Mundo!")
14
15 Fimalgoritmo
16
```

Fonte: Elaborado pelo autor (2023).

Ao substituir o comando “escreva” por “escreval”, note abaixo que após escrever na tela irá pular uma linha.



O estudo de algoritmos e lógica de programação é fundamental para criar programas de computador. Ele envolve a etapa de planejamento, onde definimos as instruções necessárias para resolver um problema, antes mesmo de escolher a linguagem de programação.

A lógica é uma forma eficiente de resolver problemas, tanto na programação quanto em situações cotidianas. É importante entender isso para desenvolver programas e lidar com desafios diários.



- 1.** O que são linguagens de programação?
- 2.** O que significa dizer que uma linguagem é de baixo nível?
- 3.** O que significa dizer que uma linguagem é de alto nível?
- 4.** Pesquise e cite duas linguagens que são de baixo nível. Pesquise e cite três linguagens de alto nível.

5. O que é o Portugol?
6. O que são programas executáveis?
7. Qual a diferença entre programas interpretadores e compiladores?
8. Qual é a tecla de atalho utilizada para executar um algoritmo no VISUALG?
9. Defina código-fonte.
10. Desenvolva no VISUALG um algoritmo que irá mostrar um ditado popular para o usuário

Exemplo:

O ditado popular do dia é "Quem com ferro fere, com ferro será ferido."

11. Faça uma pesquisa de alguma receita de comida na internet e faça um algoritmo no VISUALG que exiba esta receita para o usuário.

Exemplo:

Pudim de Leite Condensado

INGREDIENTES:

- o 1 lata de leite condensado
- o 1 lata de leite (utilize a lata do leite condensado como medida)
- o 3 ovos
- o 1 xícara (chá) de açúcar (para a calda)

MODO DE PREPARO:

- o Prepare a calda de caramelo derretendo o açúcar em fogo baixo e espalhando-o na forma de pudim.
- o No liquidificador, bata o leite condensado, o leite e os ovos até obter uma mistura homogênea.
- o Despeje a mistura do pudim na forma caramelizada.
- o Asse em banho-maria a 180°C por cerca de 1 hora, cobrindo a forma com papel alumínio.

- Faça o teste do palito para verificar se o pudim está pronto.
 - Deixe o pudim esfriar e leve-o à geladeira até ficar bem gelado.
 - Desenforme o pudim, passando uma faca nas bordas, e sirva.

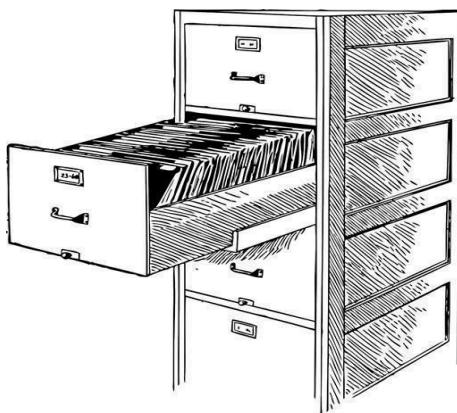
TEMA 2

Fluxogramas e Condicionais

Habilidades:

- Elaborar algoritmos.
- Codificar programas utilizando técnica de programação estruturada.

Variáveis



Fonte: Elaborado pelo autor (2023).

Fonte vídeos: <https://youtu.be/oyU94AHLyAM>

As variáveis são como espaços de memória do computador onde podemos guardar valores, como números, letras ou verdadeiro/falso. Esses valores são armazenados enquanto o programa está sendo executado e são apagados quando o programa é fechado. Podemos comparar a um armário onde arquivos modificam informações o tempo todo. São pelas variáveis que os programas fazem verificações e cálculos de acordo com o que é passado.

Existem dois tipos de variáveis: globais e locais. As variáveis globais podem ser acessadas em qualquer parte do programa, enquanto as variáveis locais só podem ser usadas em partes específicas do programa, como funções ou procedimentos. A escolha entre usar variáveis globais ou locais depende de como queremos utilizar essas variáveis.

Existem diferentes tipos de variáveis que podemos usar em programas:

Inteiros: São usados para armazenar números inteiros, ou seja, números sem casas decimais.

Exemplo:

```
idade = 20
```

Fonte: Elaborado pelo autor (2023).

Números reais: São usados para armazenar números com casas decimais, como números com ponto flutuante.

Exemplo:

```
altura = 1.75
```

Fonte: Elaborado pelo autor (2023).

Lógicas (ou Booleanas): São usados para armazenar valores verdadeiros ou falsos. Vale lembrar que aqui são somente válidos “verdadeiro” ou “falso”.

Exemplo:

```
temperatura_alta = verdadeiro
```

Fonte: Elaborado pelo autor (2023).

Caracteres: São usados para armazenar letras, palavras ou até mesmo números que serão tratados como texto.

Exemplo:

```
nome = "João"
```

Fonte: Elaborado pelo autor (2023).

Criando variáveis do tipo inteiro

Para definir uma variável do tipo inteiro fazemos o seguinte, no VisualG existe uma área para declaração de variáveis, toda variável criada nesta parte será uma variável global.

No exemplo a área para declaração de variáveis está localizada a partir da linha 5.

Para criar uma variável do tipo inteiro, basta colocar o nome_da_variável: inteiro, assim como feito na linha 6.

Na linha 12 atribuímos o valor “10” para a variável “a”.

Para atribuir valores para alguma variável utilizamos os símbolos <- (veja na linha 10), dizemos neste caso que a variável “a” recebeu o valor “10”.

E por último mostramos o valor que está armazenado na variável “a” através do comando escreva (veja na linha 15)

```

1 Algoritmo "Variável do tipo inteiro"
2 // Disciplina : [Linguagem e Lógica de Programação]
3
4 Var
5 // Seção de Declarações das variáveis
6 a:inteiro
7
8 Inicio
9 // Seção de Comandos, procedimento, funções, operadores, etc...
10
11 //atribuindo o valor 10 para a variável "a"
12 a <- 10
13
14 //escrevendo na tela
15 escreva (a)
16
17 Fimalgoritmo

```



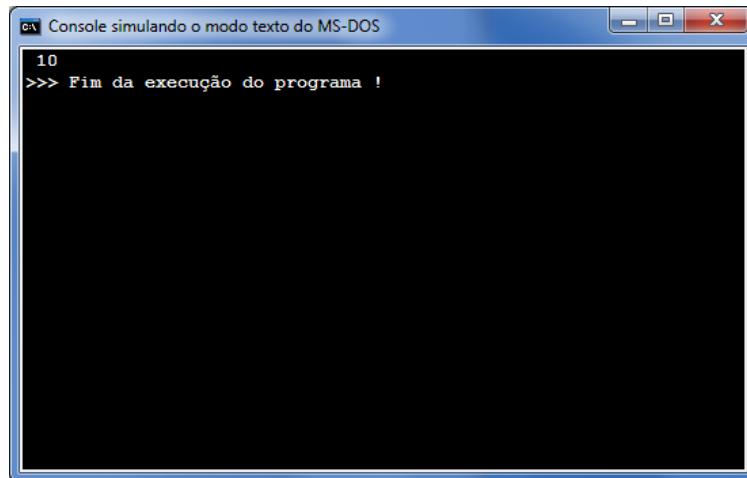
Fonte: Elaborado pelo autor (2023).

Pressionando a tecla “F9” ou indo em Run (executar) => Rodar o Algoritmo



Fonte: Elaborado pelo autor (2023).

Abaixo segue a execução do nosso algoritmo.



Fonte: Elaborado pelo autor (2023).

No exemplo abaixo declaramos duas variáveis (a e b) na mesma linha como do tipo inteiro (Veja a linha 6).

Note que nas linhas 12 e 13 atribuímos os valores para as variáveis.

Na linha 15 mandamos escrever na tela a soma dos dois valores armazenados nas variáveis.

Fonte: próprio autor

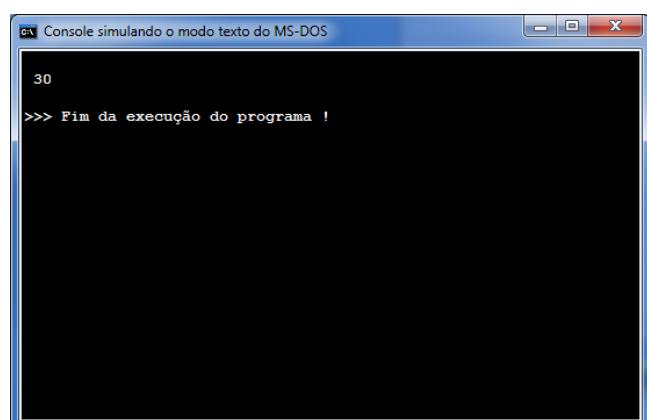
```
1 Algoritmo "Variável do tipo inteiro"
2 // Disciplina : [Linguagem e Lógica de Programação]
3
4 Var
5 // Secção de Declarações das variáveis
6 a,b:inteiro ←
7
8 Inicio
9 // Seção de Comandos, procedimento, funções, operadores, etc...
10
11 //atribuindo valores para as variáveis "a" e "b"
12 a <- 10 ←
13 b <- 20 ←
14 //escrevendo na tela
15 escreval (a+b)
16
17 Fimalgoritmo
18
```

Fonte: Elaborado pelo autor (2023).

Tela de execução do algoritmo:

Fonte: Elaborado pelo autor (2023).

Operadores aritméticos



Para realizar operações matemáticas utilizaremos os operadores aritméticos abaixo:

Operador	Descrição
+	Para realizar a adição
-	Para realizar a subtração
*	Para realizar a multiplicação
/	Para realizar a divisão

Fonte: Elaborado pelo autor (2023).

Neste exemplo, faremos várias operações matemáticas com os números inteiros.

Note que nas linhas 13 e 16 estamos lendo os valores para as variáveis “a” e “b”. Para que nas

```
1 Algoritmo "Operacoes matematicas"
2
3 Var
4 //Seção de Declarações das variáveis
5
6 a,b:inteiro ←
7
8 Inicio
9 //Seção de comandos, procedimento, funções, operadores, etc...
10
11
12 escreval ("Digite um valor para A:")
13 leia (a) ←
14
15 escreval ("Digite um valor para B:")
16 leia (b) ←
17
18
19 escreval (" Resultado da soma:", A+B)
20 escreval (" Resultado da subtração:", A-B)
21 escreval (" Resultado da divisão:", A/B)
22 escreval (" Resultado do resto da divisão:", A MOD B)
23 escreval (" Resultado da multiplicação", A*B)
24
25 Fimalgoritmo
26
```

próximas linhas fazermos diversos cálculos com os valores digitados pelo usuário:

Fonte: Elaborado pelo autor (2023).

Veja abaixo a tela de execução.

```
c:\ Console simulando o modo texto do MS-DOS
2 Digite um valor para A:
10
Digite um valor para B:
20
Resultado da soma: 30
Resultado da subtração: -10
```

Fonte: Elaborado pelo autor (2023).

Lendo os valores de uma única vez

Note na linha 13 que estamos lendo os valores para as variáveis de uma única vez, esse método de leitura é uma boa forma para diminuir a quantidade de linhas do nosso código.

```
1 Algoritmo "Operacoes matematicas"
2
3 Var
4 //Seção de Declarações das variáveis
5
6 a,b:inteiro ←
7
8 Inicio
9 //Seção de comandos, procedimento, funções, operadores, etc...
10
11
12 escreval ("Digite um valor para A e B:")
13 leia (a,b) ←
14
15 escreval (" ")
16
17 escreval (" Resultado da soma:", a+b)
18 escreval (" Resultado da subtração:", a-b)
19 escreval (" Resultado da divisão:", a/b)
20 escreval (" Resultado do resto da divisão:", a MOD b)
21 escreval (" Resultado da multiplicação", a*b)
22
23 Fimalgoritmo
```

Fonte: Elaborado pelo autor (2023).

Veja abaixo a tela de execução.

The screenshot shows a terminal window titled "Console simulando o modo texto do MS-DOS". The window displays the following interaction:

```
Digite um valor para A e B:
5
2

Resultado da soma: 7
Resultado da subtração: 3
Resultado da divisão: 2.5
Resultado do resto da divisão: 1
Resultado da multiplicação 10

>>> Fim da execução do programa !
```

Fonte: Elaborado pelo autor (2023).

Criando variáveis do tipo real:

Para criar uma variável do tipo real, basta colocar o nome_da_variável: real, assim como feito na linha 7.

```
1 Algoritmo "Variável do tipo inteiro"
2 // Disciplina : [Linguagem e Lógica de Programação]
3
4 Var
5 // Seção de Declarações das variáveis
6 a,b:inteiro
7 media:real ←
8 Inicio
9 // Seção de Comandos, procedimento, funções, operadores, etc...
10
11 //atribuindo valores para as variáveis "a" e "b"
12 a <- 10
13 b <- 20
14
15 // atribuindo para a variável "média" o cálculo da média
16 media <- (a+b)/2 ←
17
18 //Escrevendo a média na tela
19 escreval (media)
20
21 Fimalgoritmo
```

Fonte: Elaborado pelo autor (2023).

Abaixo segue a tela de execução do algoritmo.

The screenshot shows a DOS-style terminal window titled "Console simulando o modo texto do MS-DOS". The output of the program is displayed:

```
15
>>> Fim da execução do programa !
```

Fonte: Elaborado pelo autor (2023).

Criando uma variável lógica (booleana):

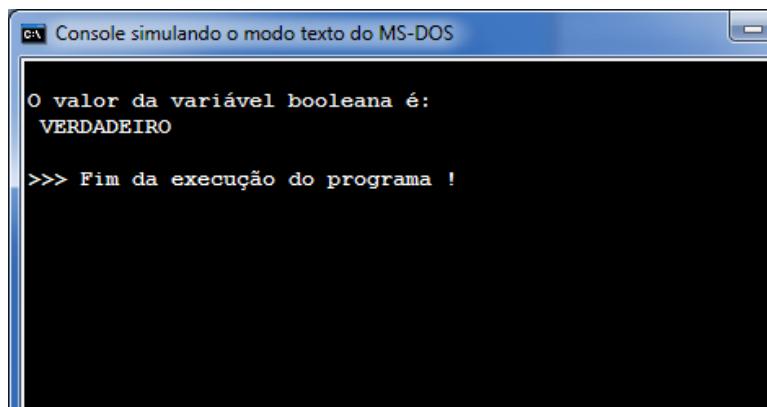
Para criar uma variável do tipo lógico, basta colocar o nome_da_variável : logico, assim como feito

```
1 Algoritmo "variaveis logicas"
2 // Disciplina : [Linguagem e Lógica de Programação]
3
4 Var
5 // Seção de Declarações das variáveis
6 booleana:logico ←
7
8 Inicio
9 // Seção de Comandos, procedimento, funções, operadores, etc...
10
11 //Atribuindo o valor de "verdadeiro" para a variável
12 booleana <- verdadeiro ←
13
14 //Escrevendo o valor da variável na tela
15
16 escreval("O valor da variavel booleana é", booleana)
17
18 Fimalgoritmo
19
```

na linha 6 do exemplo abaixo:

Fonte: Elaborado pelo autor (2023).

Tela de execução do algoritmo:



Fonte: Elaborado pelo autor (2023).

Criando uma variável do tipo caractere:

Para criar uma variável do tipo caractere, basta colocar o nome_da_variável: caracter, assim como feito na linha 7.

```
1 Algoritmo "variaveis_caracter"
2 // Disciplina : [Linguagem e Lógica de Programação]
3
4 Var
5 //Seção de Declarações das variáveis
6
7 nome, sobrenome:caracter ←
8
9 Inicio
10 //Seção de comandos, procedimento, funções, operadores, etc...
11
12 escreval ("Digite o seu primeiro nome:")
13 leia (nome)
14
15 escreval ("Digite o seu sobrenome")
16 leia (sobrenome)
17
18 //Escrevendo e concatenando na tela o texto junto com as variáveis
19 escreval ("O seu nome é:", nome, " ",sobrenome)
20
21
22 Fimalgoritmo
```

Fonte: Elaborado pelo autor (2023).

Abaixo o algoritmo em execução:

The screenshot shows a terminal window titled "Console simulando o modo texto do MS-DOS". The window contains the following text:
Digite o seu primeiro nome:
Rafaela
Digite o seu sobrenome
Silva
O seu nome é:Rafaela Silva
->>> Fim da execução do programa !

Fonte: Elaborado pelo autor (2023).



RESUMO

As variáveis são como "caixinhas" onde podemos guardar diferentes tipos de informações, como números, palavras, ou valores verdadeiros ou falsos. Elas são muito importantes para os programas, pois nos permitem trabalhar com diferentes tipos de dados e realizar cálculos ou tomadas de decisão.



ATIVIDADE DE FIXAÇÃO

- 1.** Qual opção representa variáveis do tipo inteiro?
 - a) 1, 2.5, 3, 4.1.
 - b) 50, 34, 67,45, 76.5.
 - c) 40, 56.33, 34, 24,59.
 - d) 2.8,10, 11, 12,17.
- 2.** O que são variáveis do tipo “lógico”? Explique com suas palavras.
- 3.** É possível realizar operações entre números Inteiros e Números reais, porém se o resultado for um número Real, a variável que recebe o resultado precisa ser do tipo Real. Esta afirmação é:
 - a) Verdadeira
 - b) Falsa
- 4.** Defina o que é uma variável do tipo “real”. Dê um exemplo de um número real.
- 5.** Defina o que é uma variável do tipo “caracter”. É possível armazenar somente um único caractere numa variável deste tipo?
- 6.** Faça um programa que calcule o valor de desconto para um determinado produto, faça um Print Screen do código fonte do seu programa e do programa em execução.
 - OBS: No programa deve aparecer o seu nome, deve permitir que o usuário digite o código do produto, o valor do produto, o valor da porcentagem do desconto. No final deve ser calculado e apresentado o valor total a pagar.

Segue abaixo um exemplo:

Nome do aluno: >>>Aqui deve aparecer o seu nome <<< Digite o código do produto: 1515

Digite o valor do produto: 250

Digite a porcentagem do desconto: 25 O

Valor total a pagar é de: R\$ 143.5

Fluxogramas

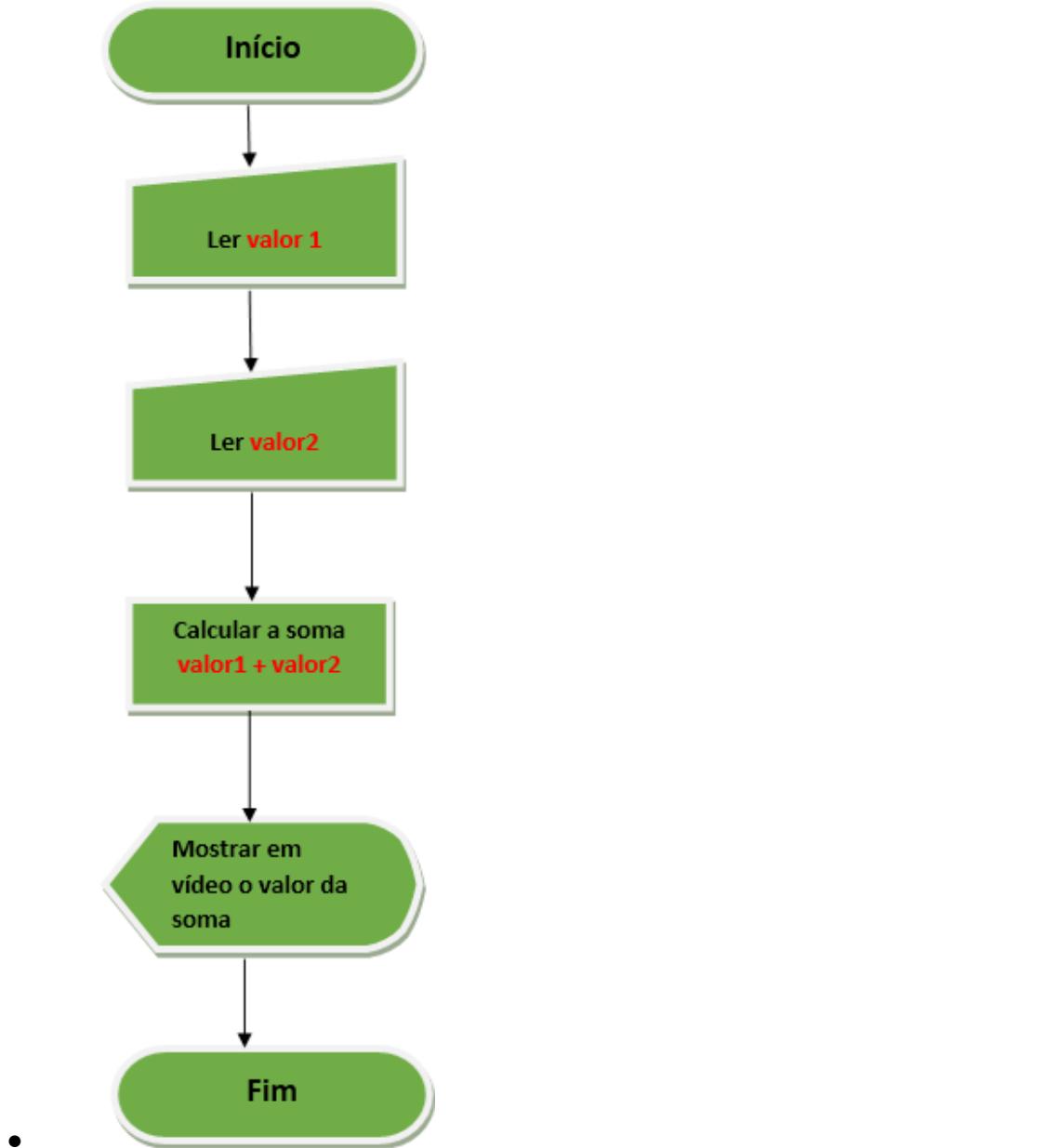
O fluxograma é uma representação gráfica de um processo ou ação, usando símbolos e setas para mostrar a sequência do fluxo. Ele ajuda no planejamento e desenvolvimento de sistemas complexos. Para escrever bons algoritmos, é importante seguir algumas regras básicas, como ter um início e fim claros, manter uma sequência lógica das operações, usar apenas um verbo por linha de comando e ser objetivo. Os fluxogramas são especialmente úteis na fase inicial de projetar e definir um sistema.

Exemplo 1 – Somando dois valores

Passo a Passo:

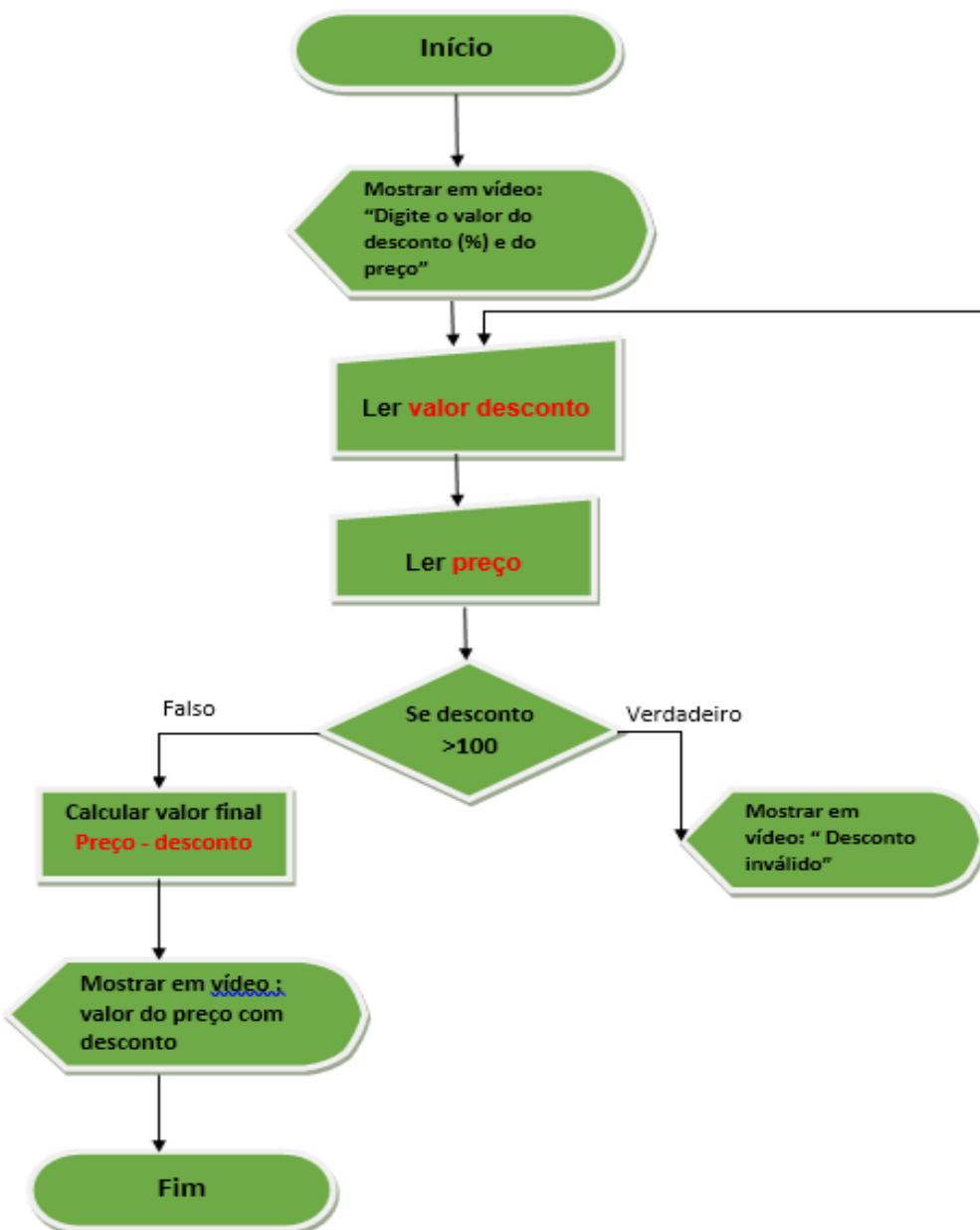
- Inicie o fluxograma.
- Defina as variáveis para armazenar os valores a serem somados.
- Leia o primeiro valor.
- Leia o segundo valor.
- Some os dois valores.
- Armazene o resultado da soma em uma variável.
- Mostre o resultado da soma.
- Finalize o fluxograma.

Lembrando que cada passo é representado por um símbolo específico no fluxograma, como por exemplo, o símbolo de entrada para ler os valores, o símbolo de processamento para realizar a soma e o símbolo de saída para mostrar o resultado.



Fonte: Elaborado pelo autor (2023).

Exemplo 2 – Utilizando condicionais



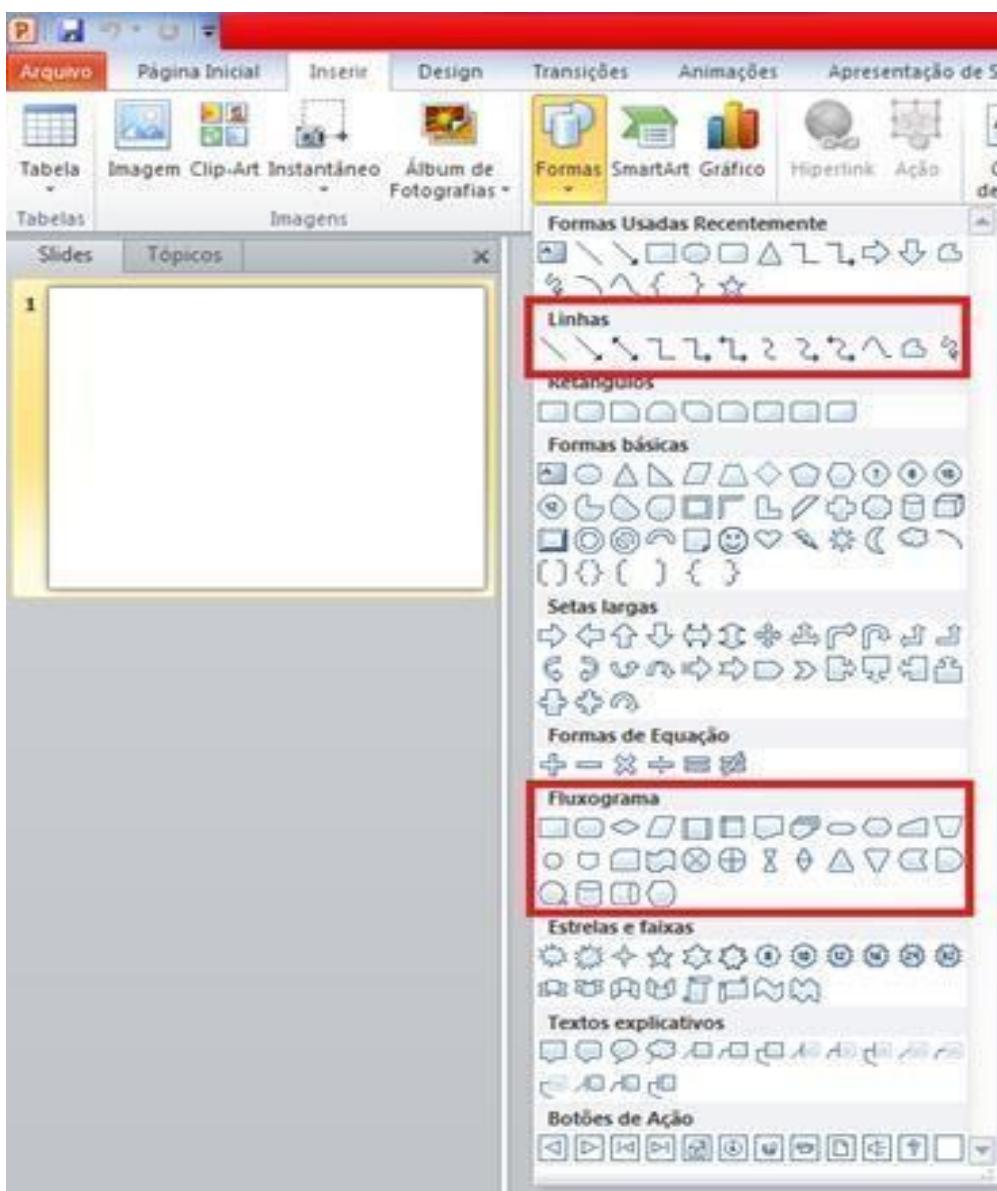
Fonte: Elaborado pelo autor (2023).

Criando fluxogramas pelo Power Point

Para criar fluxogramas no PowerPoint, siga estes passos simples:

- Clique na guia "Inserir".
- Em seguida, clique em "Formas".

- Lá, você encontrará os símbolos utilizados nos fluxogramas, bem como as setas que são usadas para conectar os elementos do fluxo.



Fonte: Elaborado pelo autor (2023).



RESUMO

Para que um algoritmo seja compreendido por todos, é necessário usar uma forma de representação que seja clara para todos. Um fluxograma é uma maneira gráfica de representar um algoritmo. Isso ajuda a planejar e organizar sistemas mais complexos. Em sistemas simples, é possível pular essa etapa, mas em sistemas maiores, é importante usar fluxogramas para evitar problemas de organização.



Para mais dicas acesse no QRcode:

Fonte: <https://youtu.be/HB2bUfhpRqo>



ATIVIDADE DE FIXAÇÃO

1. Qual a utilidade em utilizar um fluxograma?
2. Em qual fase do projeto os fluxogramas são mais úteis?
3. Através do próprio Word, crie um fluxograma que calculará a média entre dois valores. Faça todo o processo com base nos exemplos mostrados neste tema ou em aula.
4. Através do próprio Word crie um fluxograma para ler a nota do aluno, se a nota for menor que

7 deverá aparecer uma mensagem: "Aluno reprovado!" Caso contrário deverá aparecer "Passou de ano!".

5. Pesquise na internet três programas gratuitos que são próprios para criar fluxogramas e ou sites online que permitem a criação de fluxogramas gratuitamente.

- Escolha um dos programas e/ou site da questão anterior e refaça o fluxograma feito na questão 3, porém utilizando este programa e/ou site, caso precise instalar algum programa utilize a máquina virtual.

Condicionais



Vídeo: O que é Programação de Computadores:

Fonte: <https://youtu.be/qyfFw6oWuTk>

Disponível em <<https://tinyurl.com/36wxaphva>>. Acesso em: 10 jul. 2023.

As condicionais são situações em que algo só acontece se uma determinada condição for cumprida. Por exemplo, quando uma mãe diz ao filho que ele só pode comer chocolate depois de comer toda a comida, ela está colocando uma condição para que o filho possa ter o chocolate. Se ele cumprir a condição, ele pode comer o chocolate, caso contrário, ele não pode.

Operadores de comparação

Para criarmos condicionais utilizamos muitas vezes os operadores de comparação.

Descrição do símbolo	Usar no VISUALG
Menor	<
Maior	>
Menor ou igual	<=
Maior ou igual	>=
Igual	=
Diferente	<>

Fonte: Elaborado pelo autor (2023).

Condicionais Simples

A condicional simples funciona da seguinte maneira, utilizamos:

Se (condição for verdadeira) então

No exemplo abaixo se o valor da variável for menor que 10 então aparecerá na tela a mensagem “O valor digitado é menor que 10.”



Fonte:<https://youtu.be/ovDcfdiUsjc>

Note que a condicional “se” deve ser finalizada com um “fimse”.

```
1 Algoritmo "condicional_simples"
2 // Disciplina : [Linguagem e Lógica de Programação]
3
4 Var
5 // Seção de Declarações das variáveis
6
7 valor:inteiro
8
9 Início
10 // Seção de Comandos, procedimento, funções, operadores, etc...
11
12 escreval("Escreva um valor:")
13
14 leia (valor)
15
16 // Condicional Simples
17
18 se (valor < 10) entao ←
19
20   escreval ("O valor digitado é menor que 10")
21
22 fimse
23
24 Fimalgoritmo
```

Fonte: Elaborado pelo autor (2023).

Tela de execução do algoritmo:

The screenshot shows a terminal window titled "Console simulando o modo texto do MS-DOS". The window contains the following text:
Escreva um valor:
5
O valor digitado é menor que 10
>>> Fim da execução do programa !

Fonte: Elaborado pelo autor (2023).

Condicionais Compostas

Exemplo1:

No exemplo abaixo note que se a condição for verdadeira, ou seja, (nota >=7) o algoritmo exibirá “Parabéns! Continue assim!” Caso contrária exibirá “Você precisa estudar mais!”

```
1 Algoritmo "condicional_composta"
2 // Disciplina : [Linguagem e Lógica de Programação]
3
4 Var
5 // Seção de Declarações das variáveis
6
7 nota:inteiro
8
9 Inicio
10 // Seção de Comandos, procedimento, funções, operadores, etc...
11
12 escreval("Escreva o valor da nota")
13
14 leia (nota)
15
16 // Condicional Composta
17
18 se (nota >= 7) entao
19
20     escreval ("Parabéns! Continue assim!")
21
22     senao
23
24         escreval ("Você precisa estudar mais!")
25
26 fimse
27
28 Fimalgoritmo
```

Fonte: Elaborado pelo autor (2023).

Abaixo segue as telas de execução:

```
c:\ Console simulando o modo texto do MS-DOS
Escreva o valor da nota:
6,9
hummmmm você precisa estudar mais heim!
>>> Fim da execução do programa !
```

Fonte: Elaborado pelo autor (2023).

c:\ Console simulando o modo texto do MS-DOS

Escreva o valor da nota
7,5
Parabéns! Continue assim!
>>> Fim da execução do programa !

Fonte: Elaborado pelo autor (2023).

Exemplo 2:

```
7 Var
8 // Seção de Declarações das variáveis
9
10 NOTA: real
11
12
13 Inicio
14 // Seção de Comandos, procedimento, funções, operadores, etc...
15
16 escreval ("Escreva o valor da nota:")
17
18 leia(NOTA)
19
20 // Condicional Composta
21
22 se (NOTA >= 7) então
23
24     escreval ("Parabéns pela nota!")
25
26 senao
27     se (NOTA <5) então
28
29         escreval("Você precisa estudar urgentemente!")
30
31     senao
32
33         escreval("A sua nota não está tão ruim, mas pode melhorar.")
34
35 fimse
36
37 fimse
38
39 Fimalgoritmo
```

Fonte: Elaborado pelo autor (2023).

Abaixo segue a tela de execução:

```
c:\> Console simulando o modo texto do MS-DOS  
  
Escreva o valor da nota:  
5  
A sua nota não está tão ruim, mas pode melhorar.  
>>> Fim da execução do programa !
```

Fonte: Elaborado pelo autor (2023).

Condicionais Múltiplos e Conectivos lógicos

Conectivo lógico “E”

Para que as ações sejam executadas é necessário que todas as condições sejam verdadeiras, do contrário não serão executadas.

Exemplo 1:



Fonte: Elaborado pelo autor (2023)

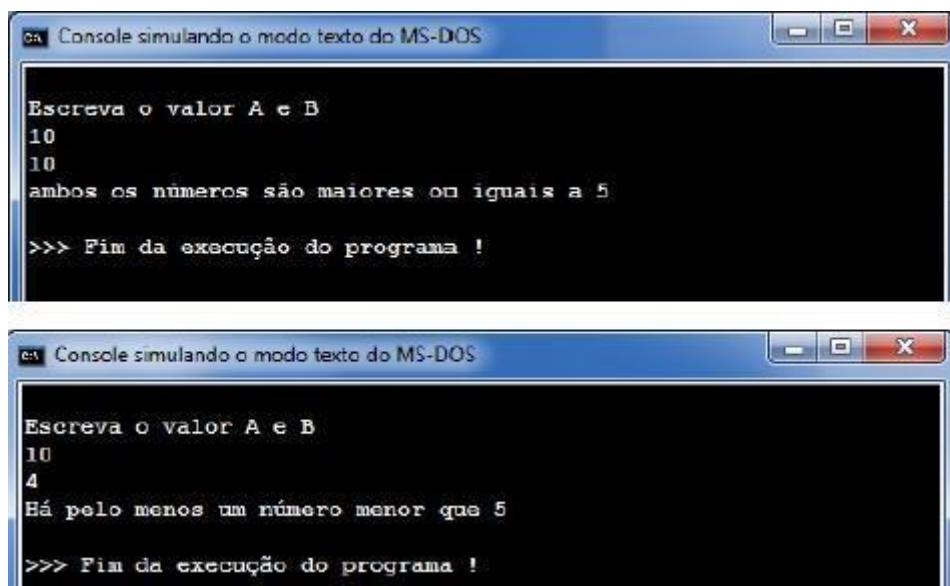
No exemplo acima, a garota tem duas condições para sair, essas duas condições (receber o salário e não chover) precisam ser verdadeiras para que ela saia, do contrário ela não irá sair.

Exemplo 2:

```
7 Var
8 // Seção de Declarações das variáveis
9
10 A,B: real
11
12 Início
13 // Seção de Comandos, procedimento, funções, operadores, etc...
14
15 escreval ("Escreva o valor A e B")
16
17 leia(A,B)
18
19 // Condicional múltipla com conectivo lógico "E"
20
21 se (A >= 5) e (B >= 5) entao
22
23     escreval ("ambos os números são maiores ou iguais a 5")
24
25 senao
26
27     escreval ("Há pelo menos um número menor que 5")
28
29 fimse
30
31 Fimalgoritmo
```

Fonte: Elaborado pelo autor (2023).

Segue abaixo as telas de execução:



Fonte: Imagens elaboradas pelo autor (2023).

Conejivo lógico “OU”

Para que as ações sejam executadas é necessário que qualquer uma das condições sejam verdadeiras.

Exemplo 1:

No exemplo abaixo, se qualquer uma das condições for verdadeira (passa o ônibus A ou B) ele conseguirá chegar no horário. Se nenhum dos ônibus passar ele se atrasará.



Fonte: Elaborado pelo autor (2023).

Exemplo 2:

```
7 Var
8 // Seção de Declarações das variáveis
9
10 A,B: real
11
12 Inicio
13 // Seção de Comandos, procedimento, funções, operadores, etc...
14
15 escreval ("Escreva o valor A e B")
16
17 leia(A,B)
18
19 // Condicional múltipla com conectivo lógico "OU"
20
21 se (A > 5) ou (B > 5) entao
22
23     escreval ("Ao menos um dos números é maior que 5")
24
25 senao
26
27     escreval ("Ambos os números são menores ou iguais a 5")
28
29 fimse
30
31 Fimalgoritmo
```

Fonte: Elaborado pelo autor (2023).

Abaixo seguem as telas de execução:

The image contains two screenshots of a DOS console window. Both windows have a title bar 'Console simulando o modo texto do MS-DOS'. The first window shows the following interaction:
Escreva o valor A e B
10
4
Ao menos um dos números é maior que 5
>>> Fim da execução do programa !
The second window shows the following interaction:
Escreva o valor A e B
4
2
Ambos os números são menores ou iguais a 5
>>> Fim da execução do programa !

Fonte: Imagens elaboradas pelo autor (2023).

Escolha caso

A escolha caso é muito utilizado em telas com menus de interação com o usuário.

```
7 Var
8 // Seção de Declarações das variáveis
9
10 VALOR1,VALOR2:real
11 OPCAO: inteiro
12
13 Inicio
14 // Seção de Comandos, procedimento, funções, operadores, etc...
15
16 escreval ("Digite o valor 1 e o valor 2")
17 leia (VALOR1,VALOR2)
18
19 escreval ("Escolha o que você deseja calcular")
20 escreval ("Digite 1 para soma")
21 escreval ("Digite 2 para subtração")
22
23 //ESCOLHA CASO
24
25 leia (OPCAO)
26
27 escolha (OPCAO)
28     caso 1
29         escreval ("O resultado da soma é: ",VALOR1+VALOR2)
30     caso 2
31         escreval ("O resultado da subtração é: ", VALOR1-VALOR2)
32     outrocaso
33         escreval ("Você digitou uma opção inválida")
34 fimescolha
35
36 Fimalgoritmo
```

Fonte: Elaborado pelo autor (2023).

Segue abaixo as telas de execução:

The screenshot shows a terminal window titled "Console simulando o modo texto do MS-DOS". The window displays the following interaction:

```
Digite o valor 1 e o valor 2
20
6
Escolha o que você deseja calcular
Digite 1 para soma
Digite 2 para subtração
1
O resultado da soma é: 26

>>> Fim da execução do programa !
```

```
c:\ Console simulando o modo texto do MS-DOS
Digite o valor 1 e o valor 2
20
6
Escolha o que você deseja calcular
Digite 1 para soma
Digite 2 para subtração
3
Você digitou uma opção inválida
>>> Fim da execução do programa !
```

Fonte: Imagens elaboradas pelo autor (2023).



RESUMO

Utilizamos condicionais todos os dias, elas são muito importantes para tomadas de decisão. Por exemplo, quando alguém pensa, se fizer sol amanhã eu irei para a praia, ir para a praia só será possível se a condição “se fizer sol” for verdadeira. Através das condicionais conseguimos criar algoritmos preparados para diversas situações previstas.



ATIVIDADE DE FIXAÇÃO

1. Desenvolva um programa utilizando uma condicional simples, o algoritmo deve verificar se o valor digitado é menor ou igual a 20. Se a condição for verdadeira, deve aparecer uma mensagem:

"O valor digitado é menor ou igual a 20!".

- OBS: Tire um Print Screen do código fonte e também da tela de execução do programa.

Exemplo de tela:

Autor do programa: AQUI DEVE APARECER O SEU NOME
Atividade 1 – Condicional simples

Digite um valor: 19
O valor digitado é menor ou igual a 20!

Fonte: Elaborado pelo autor (2023).

2. Desenvolva um programa qualquer utilizando a condicional composta.
- OBS: Tire um Print Screen do código fonte e da tela de execução do programa. O seu nome deve aparecer na tela de execução do programa.

3. Desenvolva um programa que aplicará um desconto de 25% ao valor do produto se o cliente possuir um dos seguintes códigos de promoção: “VBFMZB” ou “HT2Y8E”. Caso contrário o cliente deverá pagar o valor cheio sem desconto.

Exemplo de tela:

Autor do programa: AQUI DEVE APARECER O SEU NOME Atividade

3 – Aplicação de desconto caso haja código promocional

Digite o valor do produto: 200

Código de promocional: VBFMZB

O valor a pagar será de: R\$ 150

Fonte: Elaborado pelo autor (2023).

- OBS: Tire um Print Screen do código fonte e da tela de execução do programa

TEMA 03

Valores aleatórios e Laços de Repetição

Habilidades:

- Codificar programas utilizando técnica de programação estruturada.
- Depurar e versionar programas, utilizando ambiente de desenvolvimento integrado.

Os valores aleatórios são muito utilizados por programadores em diversas situações. As linguagens de programação normalmente possuem funções para a criação de valores aleatórios e através delas podemos gerar valores aleatórios para utilizarmos na criação de por exemplo: jogos, senhas, captcha, algoritmos de criptografia etc.

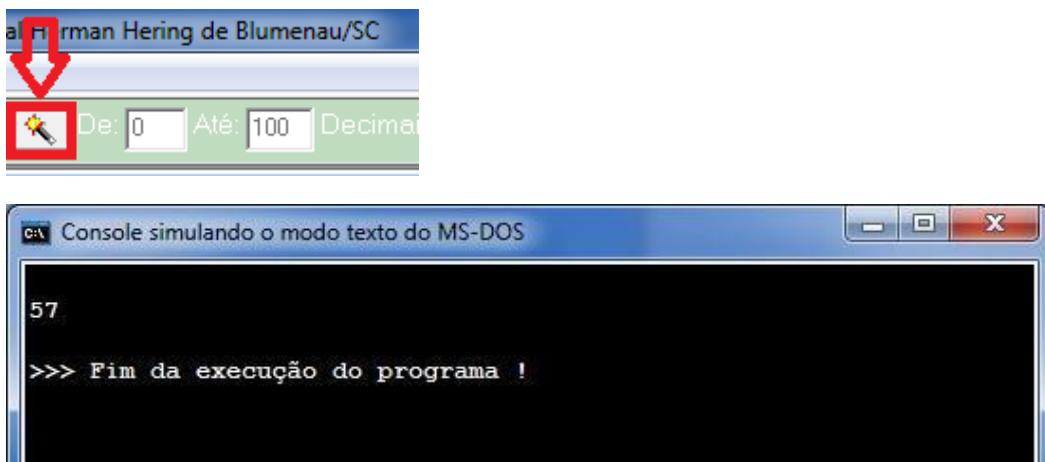
Aleatórios para números

Para criar um número aleatório, por exemplo, de tipo inteiro, é necessário a criação de uma variável que receberá esse valor aleatório, conforme a linha 10. Dentro da seção Início é possível fazer a variável receber o valor aleatório através das tags “aleatorio on” e “aleatorio off”, conforme é possível observar nas linhas entre 15 e 17.

```
7 Var
8 // Seção de Declarações das variáveis
9
10 dado:inteiro
11
12 Inicio
13 // Seção de Comandos, procedimento, funções, operadores, etc...
14
15 aleatorio on
16 leia(dado)
17 aleatorio off
18
19 Fimalgoritmo
```

Fonte: Elaborado pelo autor (2023).

O botão abaixo permite editar o valor de intervalo para a geração do valor aleatório:



Fonte: *Imagens elaboradas pelo autor (2023)*.

Note que no VisualG também é possível utilizar a função “randi()” para gerar número inteiros

Início

dado <- randi(100) // O número dentro dos parênteses é o intervalo que serão gerados os números

aleatórios, no caso, o exemplo acima ficaria assim:

Fonte: *Elaborado pelo autor (2023)*.

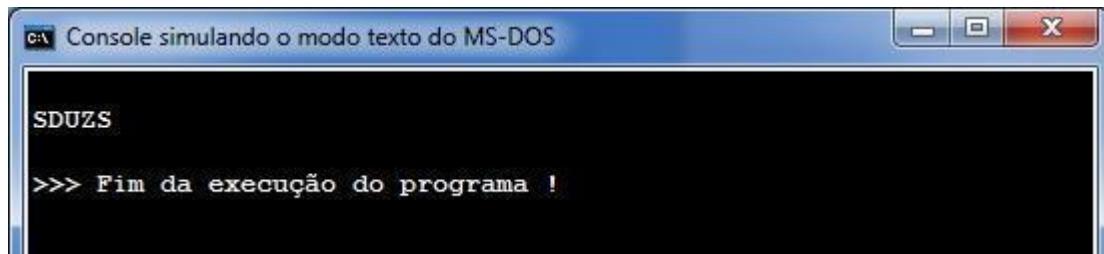
Aleatórios para caracteres

Para criar um valor aleatório do tipo caractere (letras e palavras) o procedimento é parecido ao mostrado anteriormente para inteiros, a diferença é que deverá ser declarada uma variável do tipo caracter que receberá os valores aleatórios.

```
7 Var
8 // Seção de Declarações das variáveis
9
10 palavra:caracter
11
12 Inicio
13 // Seção de Comandos, procedimento, funções, operadores, etc...
14
15 aleatorio on
16 leia(palavra)
17 aleatorio off
18
19 Fimalgoritmo
```

Fonte: Elaborado pelo autor (2023).

Abaixo segue a tela de execução:



Fonte: Elaborado pelo autor (2023).



RESUMO

Verificamos que é possível gerarmos valores aleatórios de números inteiros, números reais e de caracteres. Com eles podemos criar jogos, senhas provisórias, caracteres aleatórios para conceder um código de bônus etc. A utilidade deles dependerá do algoritmo elaborado



ATIVIDADE DE FIXAÇÃO

1. Desenvolva um programa de desafios para o usuário, o desafio dependerá do valor aleatório gerado. Crie um valor aleatório entre 0 a 9. Utilize a condicional “Escolha Caso” (tema 2 da apostila) para definir os desafios, ou seja, você terá que gerar 10 situações utilizando o “Escolha caso”, uma para cada número aleatório do intervalo entre 0 e 9.

- OBS: Tire um Print Screen do código fonte e da tela de execução do programa.

2. Desenvolva um programa para gerar 6 números aleatórios para serem jogados na loteria. Os números devem estar no intervalo de 1 a 60. Segue abaixo o exemplo da tela de execução.

- OBS: Tire um Print Screen do código fonte e da tela de execução do programa.

Exemplo:

```
c:\> Nome do Aluno: AQUI DEVE APARECER O SEU NOME
>>> GERADOR DE NÚMEROS PARA LOTERIA <<<
14
26
19
26
14
53
>>> Fim da execução do programa !
```

Fonte: Elaborado pelo autor (2023).

3. Elabore um jogo chamado “Jogo dos Palpites”, o jogo funcionará da seguinte maneira, dois jogadores farão palpites de números entre 0 e 100 o jogador que chegar mais próximo do valor aleatório ganhará. Segue abaixo as telas de exemplo do funcionamento do jogo.

- OBS: Tire um Print Screen do código fonte e também da tela de execução do programa.

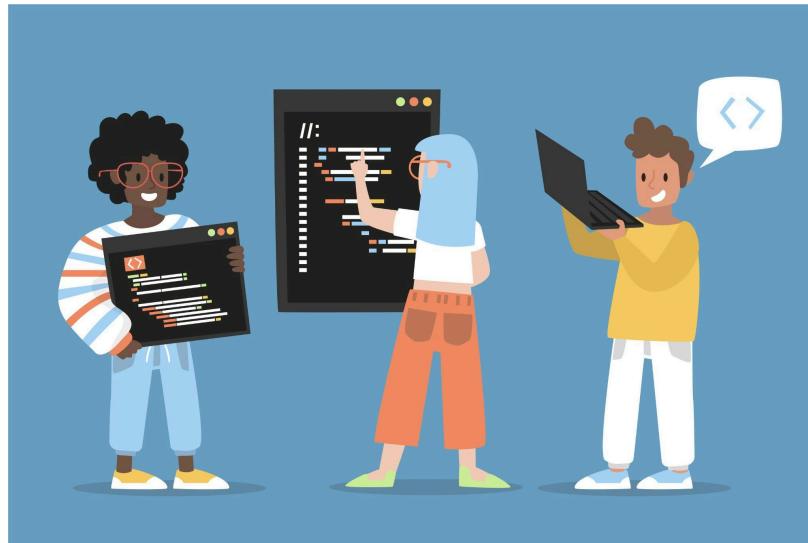
```
c:\> Nome do Aluno: AQUI DEVE APARECER O SEU NOME
>>> JOGO DOS PALPITES <<<
Jogador 1, faça um palpite entre 1 e 100: 56
Jogador 2, faça um palpite entre 1 e 100: 34
O valor aleatório gerado é: 4
O jogador 2 ganhou!
>>> Fim da execução do programa !
```

```
c:\ Console simulando o modo texto do MS-DOS
Nome do Aluno: AQUI DEVE APARECER O SEU NOME
>>> JOGO DOS PALPITES <<<
Jogador 1, faça um palpite entre 1 e 100: 10
Jogador 2, faça um palpate entre 1 e 100: 1
O valor aleatório gerado é: 39
O jogador 1 ganhou!
>>> Fim da execução do programa !
```

```
c:\ Console simulando o modo texto do MS-DOS
Nome do Aluno: AQUI DEVE APARECER O SEU NOME
>>> JOGO DOS PALPITES <<<
Jogador 1, faça um palpate entre 1 e 100: 10
Jogador 2, faça um palpate entre 1 e 100: 10
O valor aleatório gerado é: 23
Houve um empate!
>>> Fim da execução do programa !
```

Fonte: Imagens elaboradas pelo autor (2023).

Laços de repetição



Disponível em:<<https://tinyurl.com/4ayp728w>>. Acesso em 20 jul. 2023.

Os laços de repetição são muito utilizados nos sistemas para fazer algum tipo de consistência de dados do usuário, ocorre que por engano ou distração o usuário pode entrar com informações inválidas que se não forem consistidas podem prejudicar a base de dados dos sistemas. Também

podemos utilizá-los para ler as informações de vetores e matrizes, e são utilizados em algoritmos de ordenação, ou seja, há uma infinidade de utilidades para os laços de repetição.

Enquanto

Neste tipo de laço de repetição enquanto a condição estiver sendo atendida o programa permanecerá em looping.

Exemplo1:

```
1 Algoritmo "Laços de repetição"
2 // Disciplina : [Linguagem e Lógica de Programação]
3
4 Var
5 // Seção de Declarações das variáveis
6
7 NUMERO:inteiro
8
9 Inicio
10 // Seção de Comandos, procedimento, funções, operadores, etc...
11
12 //Nesta parte do código a variável NUMERO receberá o valor "0".
13 NUMERO<-0
14
15
16 // Aqui começará o laço de repetição enquanto
17 enquanto (NUMERO < 5) faca
18
19     escreval (NUMERO)
20
21     //Incrementaremos o valor de 1 a cada repetição
22     NUMERO <- NUMERO+1
23
24
25 //O laço de repetição terminará quando a condição não for mais atendida.
26 fimenquanto
27
28
29 Fimalgoritmo
```

Fonte: Elaborado pelo autor (2023).

Abaixo a tela de execução:

```
c:\ Console simulando o modo texto do MS-DOS
0
1
2
3
4
>>> Fim da execução do programa !
```

Fonte: Elaborado pelo autor (2023).

Exemplo 2:

```
1 Algoritmo "Laços de repetição"
2 // Disciplina : [Linguagem e Lógica de Programação]
3 // Autor:
4 // Data:
5 Var
6 // Seção de Declarações das variáveis
7
8 NOME:caracter
9 TENTATIVA:inteiro
10
11 Inicio
12 // Seção de Comandos, procedimento, funções, operadores, etc...
13
14 NOME <- "Zezinho"
15 TENTATIVA <- 0
16 escreval ("Escreva o seu nome:")
17
18
19 // Aqui começará o laço de repetição enquanto
20 enquanto (NOME <> "Fulano") e (TENTATIVA < 3) faça
21
22     leia (NOME)
23
24     // Incrementaremos o valor de 1 a cada repetição
25     TENTATIVA <- TENTATIVA+1
26
27     se (TENTATIVA = 3) entao
28
29         escreval ("Você excedeu o número de tentativas")
30
31     fimse
32 // O laço de repetição terminará quando a condição não for mais atendida.
33 fimenquanto
34
35
36 Fimalgoritmo
```

Fonte: Elaborado pelo autor (2023).

Seguem abaixo as telas de execução:

```
Console simulando o modo texto do MS-DOS
Escreva o seu nome:
Maria
Jose
Ricardo
Você excedeu o número de tentativas
>>> Fim da execução do programa !
```

```
Console simulando o modo texto do MS-DOS
Escreva o seu nome:
Rafaela
Fulano
>>> Fim da execução do programa !
```

Fonte: Imagens elaboradas pelo autor (2023).

Repita até que

Neste tipo de laço de repetição o algoritmo primeiro fará algo até que a condição seja atendida.

```
1 Algoritmo "repita ate"
2 // Disciplina : [Linguagem e Lógica de Programação]
3 // Professor :
4 // Autor(a)   : Nome do(a) aluno(a)
5 // Data atual  :
6 Var
7 // Seção de Declarações das variáveis
8 NUMERO:inteiro
9
10 Inicio
11 // Seção de Comandos, procedimento, funções, operadores, etc...
12
13 NUMERO <- 0
14
15 repita
16     escreval (NUMERO)
17     NUMERO <- NUMERO + 1
18 ate (NUMERO = 10)
19
20
21 Fimalgoritmo
```

Fonte: Elaborado pelo autor (2023).

Segue abaixo as telas de execução:

```
0
1
2
3
4
5
6
7
8
9

>>> Fim da execução do programa !
```

Fonte: Elaborado pelo autor (2023).

Laço de repetição: Para

Neste tipo de laço de repetição há uma variável que funcionará como contador, já na definição do laço é especificado o número de repetições. A cada repetição a variável contadora (pode ter outro nome) receberá automaticamente o acréscimo de 1.

```
1 Algoritmo "Laco de repeticao Para"
2 // Disciplina : [Linguagem e Lógica de Programação]
3 // Descrição : Exemplo de laço de repetição usando o "PARA"
4 // Autor(a)   : Nome do(a) aluno(a)
5 Var
6 // Seção de Declarações das variáveis
7
8 contador:inteiro
9
10 Inicio
11 // Seção de Comandos, procedimento, funções, operadores, etc...
12
13
14 para contador de 1 ate 5 faca
15
16 escreval (contador * 2)
17
18 fimpara
19
20 Fimalgoritmo
```

Fonte: Elaborado pelo autor (2023).

Abaixo segue a tela de execução:

The screenshot shows a Windows-style console window titled "Console simulando o modo texto do MS-DOS". The window contains the following text:
2
4
6
8
10
>>> Fim da execução do programa !

Fonte: Elaborado pelo autor (2023).



RESUMO

Verificamos que os laços de repetição são: enquanto, repita até que, para. É importante que o programador saiba utilizá-los, pois dependendo do algoritmo um poderá ser mais eficiente do que o outro. Não há um laço de repetição que seja melhor que outro, o melhor muitas vezes será aquele que o programador tenha mais afinidade.



ATIVIDADE DE FIXAÇÃO

1.Imagine que o usuário esqueceu a senha de acesso ao sistema, desenvolva um programa que solicite ao usuário o nome da mãe dele. Se o usuário acertar a resposta, deverá ser gerada uma senha através de valores aleatórios, porém, se ele errar três tentativas, deverá aparecer uma mensagem de erro: “Você excedeu o número de tentativas”.

- OBS: Tire um Print Screen do código fonte e da tela de execução do programa.

2.Desenvolva um programa para calcular o IMC (Índice de Massa Corporal), o algoritmo deverá ler a altura e o peso do usuário e depois calcular o IMC com base na fórmula: $IMC = \text{peso} / (\text{altura})^2$.

- Após o cálculo deve ser informado o valor do IMC do usuário e a classificação com base na tabela abaixo:

Classificação	IMC
Muito abaixo do peso	16 a 16,9
Abaixo do peso	17 a 18,4
Peso normal	18,5 a 24,9
Acima do peso	25 a 29,9
Obesidade Grau I	30 a 34,9
Obesidade Grau II	35 a 40
Obesidade Grau III	> 40

Fonte: Elaborado pelo autor (2023).

- Deve-se utilizar um laço de repetição para consistir se o usuário digitou um valor de peso e altura maiores do que 0. Enquanto ele digitar um valor menor ou igual a zero deverá aparecer uma mensagem de erro de: “Peso e altura devem ser maiores que 0”

- OBS: Tire um Print Screen do código fonte e da tela de execução do programa.

3.Utilizando o laço de repetição PARA, desenvolva um programa que faça uma tabuada, para isso leia um número de 1 a 10, se o usuário digitar um valor fora desse intervalo deve aparecer uma mensagem de erro: “Digite um valor entre 1 e 10.” A tela esperada deve conter os valores da tabuada para o número digitado pelo usuário.

- OBS: Tire um Print Screen do código fonte e da tela de execução do programa.

TEMA 04

Menus e validações de dados

Habilidades:

- Codificar programas, utilizando técnica de programação estruturada.
- Depurar e versionar programas, utilizando ambiente de desenvolvimento integrado.



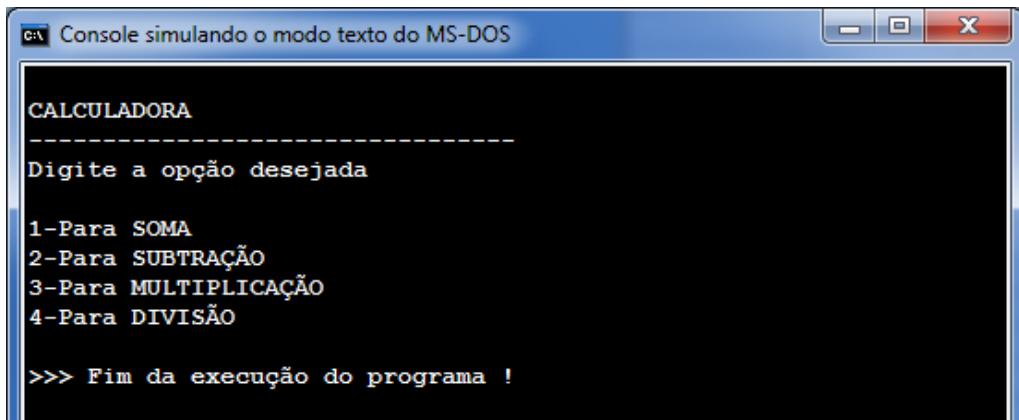
Disponível em <<https://tinyurl.com/58r6x6uk>>. Acesso em: 20 jul. 2023.

Os menus são importantes para permitir que o usuário interaja com o sistema. Eles devem ser fáceis de usar e ter um visual agradável. Os menus são como a embalagem do produto, pois mesmo que o sistema seja bom, se o menu não for bom, a experiência do usuário será prejudicada.

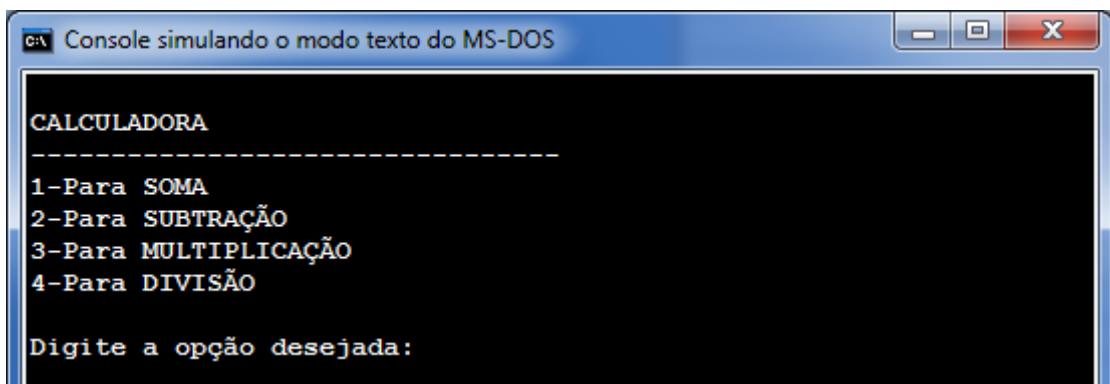
A validação de dados é necessária para lidar com erros que podem ocorrer quando o usuário insere informações no sistema. É importante que o sistema esteja preparado para lidar com esses erros e garantir que os dados inseridos sejam corretos.

Exemplo1 - Calculadora

Segue abaixo um exemplo de aplicação de calculadora:



Fonte: Elaborado pelo autor (2023).



Fonte: Elaborado pelo autor (2023).

Segue abaixo o código fonte do menu:

Note na linha 13 e 16 que foram inseridos caracteres somente para melhorar a estética do menu.

```
1 Algoritmo "Exemplo com menus e validações"
2 // Disciplina : [Linguagem e Lógica de Programação]
3 // Descrição : Aqui você descreve o que o programa faz! (função)
4 // Autor(a)   : Nome do(a) aluno(a)
5 Var
6 // Seção de Declarações das variáveis
7
8
9 Inicio
10 // Seção de Comandos, procedimento, funções, operadores, etc...
11
12 escreval ("CALCULADORA")
13 escreval ("-----") //apenas para enfeitar a tela
14
15 escreval ("Digite a opção desejada")
16 escreval (" ") //apenas para pular uma linha
17
18 escreval ("1-Para SOMA")
19 escreval ("2-Para SUBTRAÇÃO")
20 escreval ("3-Para MULTIPLICAÇÃO")
21 escreval ("4-Para DIVISÃO")
22
23 Fimalgoritmo
24
```

Fonte: Elaborado pelo autor (2023).

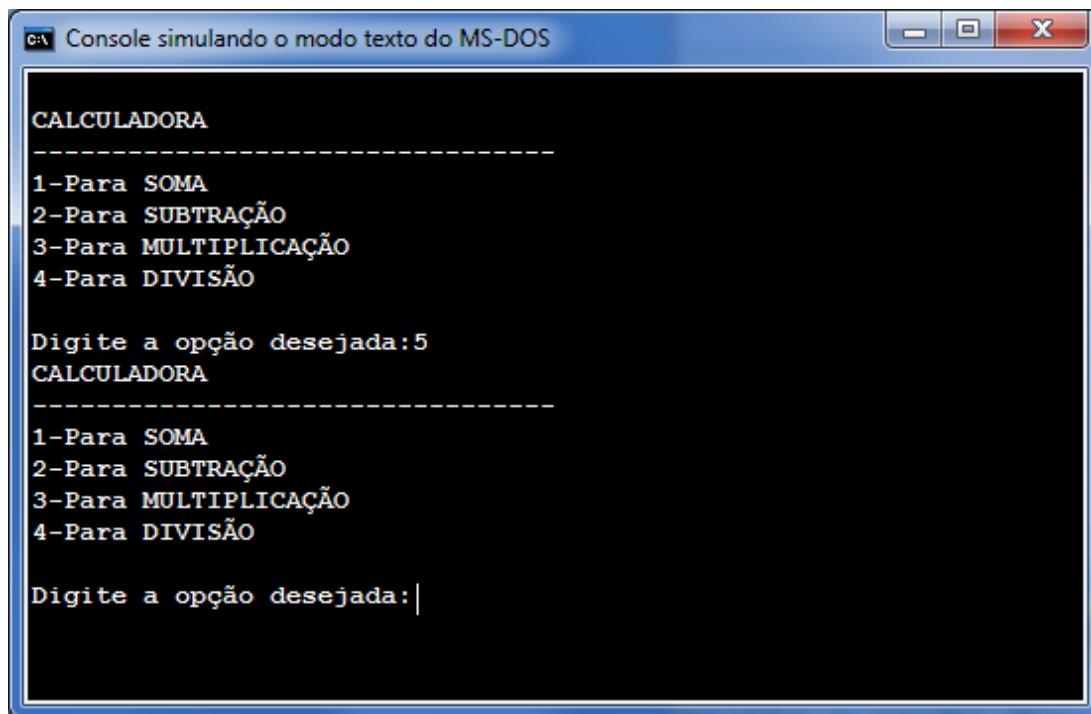
Entendendo o código 1:

Utilizamos um laço de repetição enquanto houver os valores inseridos pelo usuário, ou seja, o programa espera um número entre 1 e 4, enquanto o usuário digitar algo diferente do esperado o mesmo continuará preso ao laço de repetição.

```
5 Var
6 // Seção de Declarações das variáveis
7
8 opcao,valorA,valorB:inteiro
9 resultado:real
10
11 Inicio
12 // Seção de Comandos, procedimento, funções, operadores, etc...
13
14
15 enquanto (opcao < 1) ou (opcao > 4) faça
16
17   escreval ("CALCULADORA")
18   escreval ("-----") //apenas para enfeitar a tela
19
20   escreval ("1-Para SOMA")
21   escreval ("2-Para SUBTRAÇÃO")
22   escreval ("3-Para MULTIPLICAÇÃO")
23   escreval ("4-Para DIVISÃO")
24
25   escreval (" ") //apenas para pular a linha
26
27   escreva ("Digite a opção desejada:")
28
29   leia (opcao)
30
31 fimenquanto
32
33 limpatela
34
```

Fonte: Elaborado pelo autor (2023).

Seguem abaixo as telas de execução:

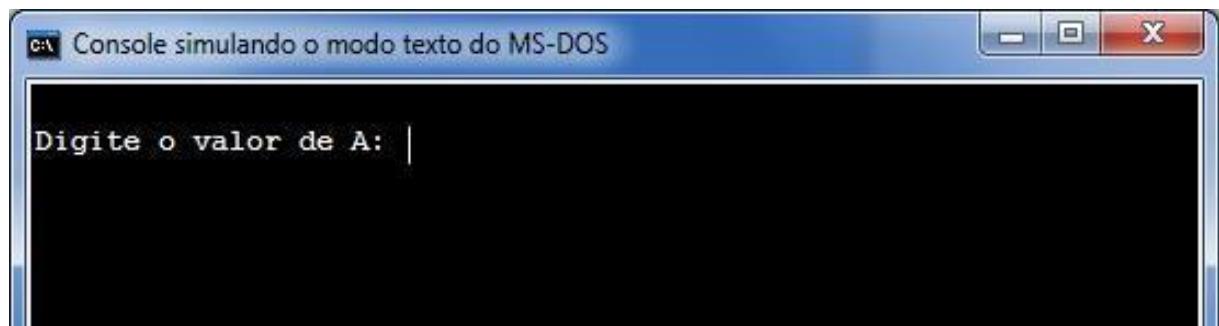


Fonte: Elaborado pelo autor (2023).

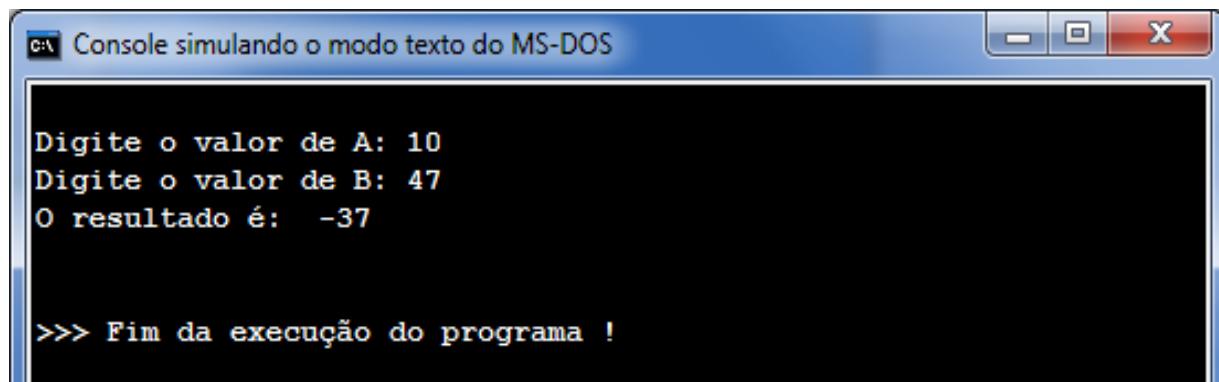
```
Digite a opção desejada:5  
CALCULADORA  
-----  
1-Para SOMA  
2-Para SUBTRAÇÃO  
3-Para MULTIPLICAÇÃO  
4-Para DIVISÃO  
Digite a opção desejada: 2|
```



Fonte: Elaborado pelo autor (2023).



Fonte: Elaborado pelo autor (2023).



```
Digite o valor de A: 10  
Digite o valor de B: 47  
O resultado é: -37  
  
>>> Fim da execução do programa !
```

Fonte: Elaborado pelo autor (2023).

Entendendo o código:

Neste caso foi utilizado a condicional “escolha” para interagir com o valor inserido pelo usuário, inclusive a condicional “escolha” é muito boa para ser utilizada nesses casos.

```
27 escreva ("Digite a opção desejada:")
28
29 leia (opcao)
30
31 fimenquanto
32
33 limpatela
34
35 escreva ("Digite o valor de A: ")
36 leia (valorA)
37
38 escreva ("Digite o valor de B: ")
39 leia (valorB)
40
41 escolha (opcao)
42     caso 1
43
44     resultado <- valorA + valorB
45
46     caso 2
47
48     resultado <- valorA - valorB
49
50     caso 3
51
52     resultado <- valorA * valorB
53
54     caso 4
55
56     resultado <- valorA / valorB
57
58 fimescolha
59
60 escreval ("O resultado é: ", resultado)
61 escreval (" ") //apenas para pular a linha
62
63 Fimalgoritmo
```



Fonte: <https://youtu.be/Y-k5h6n9wAQ>

Fonte: Elaborado pelo autor (2023).



ATIVIDADE DE FIXAÇÃO

NOME DO BANCO

CLIENTE: Nome do cliente

SELEÇÃO UMA OPÇÃO:

Saldo

Extratos (Últimos 5 dias)

3.Empréstimos

Fonte: Elaborado pelo autor (2023).

Desenvolva um programa no VISUALG que possua um menu de um sistema bancário, conforme o exemplo abaixo:

- Se o cliente escolher a opção 1, deverá apagar a tela e em seguida aparecer uma tela com um saldo fictício.
- Se o cliente escolher a opção 2, deverá apagar a tela e em seguida o sistema deverá mencionar os gastos (fictícios) mais recentes do cliente.
- Se o cliente escolher a opção 3, deverá apagar a tela e em seguida aparecer uma mensagem: "Procure a sua agência para a liberação de empréstimos."
- Criar uma consistência fazendo que o usuário digite apenas números entre 1 e 3, se o usuário digitar um número fora desse intervalo, uma mensagem de erro deverá ser apresentada com os dizeres: "Opção inválida!" E em seguida deve ser solicitada a opção novamente para o usuário até que ele digite uma opção válida.
- OBS: Tire um Print Screen do código fonte e também da tela de execução do programa.

2. Desenvolva um programa que fornecerá as informações de notas e faltas dos alunos, conforme o exemplo abaixo:

CENTRAL DO ALUNO

Aluno nome do aluno: Deverá aparecer aqui o seu nome

SELEÇÃO UMA OPÇÃO:

1- Verificar notas e faltas

2- Dúvidas

3- Sair

Fonte: Elaborado pelo autor (2023).

- Ao selecionar a opção 1 deverá:
- ✓ Limpar a tela
- ✓ Aparecer o seu nome, disciplina, quantidade de faltas (fictícias) e notas (fictícias) conforme o exemplo abaixo:

✓ Exemplo:

CENTRAL DO ALUNO		
>> NOTAS E FALTAS <<		
Nome do aluno: Deverá aparecer o seu nome		
Disciplina: Nome da disciplina	Quant. de faltas: 5	Nota: 10
Disciplina: Nome da disciplina	Quant. de faltas: 2	Nota: 4
Disciplina: Nome da disciplina	Quant. de faltas: 3	Nota: 6

Fonte: Elaborado pelo autor (2023).

- Ao selecionar a opção 2, deverá:
- ✓ Limpar a tela
- ✓ Aparecer a tela de dúvidas ao qual o aluno poderá enviar uma mensagem com alguma dúvida para o professor.

✓ Exemplo:

CENTRAL DO ALUNO	
>> DÚVIDAS <<	
Nome do aluno: Deverá aparecer o seu nome	
Digite aqui a sua dúvida:	Gostaria de solicitar a revisão da minha nota da disciplina XYZ

Fonte: Elaborado pelo autor (2023).

- Note que o sistema deverá ler a dúvida do usuário e depois apresentar a mensagem: “A sua dúvida foi enviada para o professor, por favor, aguarde uma resposta dele”.

Exemplo:

CENTRAL DO ALUNO
>>DÚVIDAS <<
Situação: A sua dúvida foi enviada para o professor, por favor, aguarde uma resposta dele.

Fonte: Elaborado pelo autor (2023).

- Se o usuário escolher a opção 3, deverá:
- ✓ Limpar a tela
- ✓ Apresentar a mensagem: “Sessão finalizada! ”.
- ✓ Exemplo:

CENTRAL DO ALUNO
Sessão finalizada!

Fonte: Elaborado pelo autor (2023).

Crie uma consistência fazendo que o usuário digite apenas números entre 1 e 3, se o usuário digitar um número fora desse intervalo, uma mensagem de erro deverá ser apresentada com os dizeres: “Opção inválida! ” E em seguida deve ser solicitada a opção novamente para o usuário até que ele digite uma opção válida.

OBS: Tire um Print Screen do código fonte e da tela de execução do programa.

Tema 05

Vetores

Habilidades:

- Codificar programas utilizando técnica de programação estruturada.
- Depurar e versionar programas, utilizando ambiente de desenvolvimento integrado.



Fonte: <https://youtu.be/GfsQ-caVqYA>

Se quisermos armazenar o nome e as notas de dez alunos, seria complicado criar muitas variáveis separadas para cada um deles. Os vetores podem nos ajudar com essa tarefa. Eles são um tipo de variável que nos permite armazenar várias informações do mesmo tipo, como números ou palavras, em uma única estrutura. Cada informação é armazenada em uma posição específica do vetor, que é como uma caixinha onde guardamos os dados. Assim, podemos acessar cada dado usando um número que indica a posição dele no vetor. Isso facilita bastante o armazenamento e a recuperação das informações dos alunos.

Exemplo:

Vetor para armazenar cinco nomes:

VetNomes:

Maria	João	Rosa	Paula	Fábio
0	1	2	3	4

Exemplo 1: Criando um vetor do tipo inteiro

Uma variável que será transformada em vetor deverá ser declarada normalmente como todas as variáveis, ou seja, dentro da seção de declarações de variáveis, ou dentro de funções e procedimentos para serem usados localmente.

Exemplo 2: criação de vetores

nomeDoVetor: vetor [tamanho do vetor] de tipo (inteiro, real, caractere ou lógico)

No exemplo abaixo, criamos um vetor com o nome vetorDeInteiros para armazenar 4 números inteiros ([0...3]), veja a linha 9 do algoritmo abaixo.

Em seguida, na linha 14, atribuímos a primeira posição de memória do vetor (posição 0) para receber o valor 2.

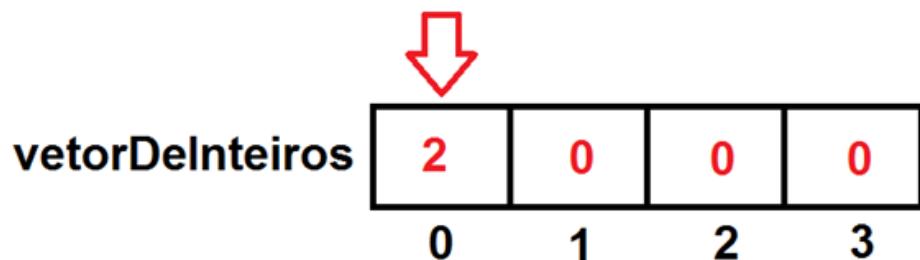
```
1 Algoritmo "Exemplo1 vetores"
2 // Disciplina : [Linguagem e Lógica de Programação]
3
4
5 Var
6 // Seção de Declarações das variáveis
7
8 // Reservando 4 posições na memória para o vetor de inteiros
9 vetorDeInteiros:vetor[0..3] de inteiro
10
11 Inicio
12 // Seção de Comandos, procedimento, funções, operadores, etc...
13
14 vetorDeInteiros[0] <- 2
15
16 escreval("Na posição 0 o valor é:", vetorDeInteiros[0])
17
18 Fimalgoritmo
```



Fonte: Elaborado pelo autor (2023).

Veja abaixo como ficou o armazenamento dos dados no vetor vetorDeInteiros, na posição 0 está armazenado o valor 2, já, nas demais posições do vetor, por padrão para vetores inteiros ou reais receberão o valor inicial de 0, podendo serem alterados quando assim desejarmos.

Representação do vetor:



Fonte: Elaborado pelo autor (2023).

Abaixo segue a tela de execução:

The screenshot shows a Windows-style console window titled "Console simulando o modo texto do MS-DOS". The window contains the following text:
Na posição 0 o valor é: 2
>>> Fim da execução do programa !

Fonte: Elaborado pelo autor (2023).

Exemplo 3: Criando um vetor do tipo caractere

No exemplo abaixo, note que o processo de criação de vetores para caracteres é parecido com o de números inteiros e reais, a diferença é que declaramos o vetor como do tipo caracter, veja a linha 9 do algoritmo abaixo.

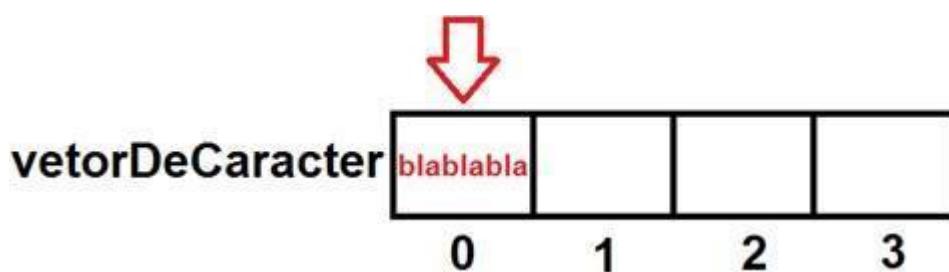
Note que na posição 0 do vetor foi atribuído o valor de "blablabla", veja a linha 14

```
1 Algoritmo "Exemplo3 vetores"
2 // Disciplina : [Linguagem e Lógica de Programação]
3
4
5 Var
6 // Seção de Declarações das variáveis
7
8 // Reservando 4 posições na memória para o vetor de caracter
9 vetorDeCaracter:vetor[0..3] de caracter
10
11 Inicio
12 // Seção de Comandos, procedimento, funções, operadores, etc...
13
14 vetorDeCaracter[0] <- "blablabla"
15
16 escreval("Na posição 0 o valor é:", vetorDeCaracter[0])
17
18 Fimalgoritmo
```

Annotations on the pseudocode highlight specific parts of the code. Red arrows point to the array declaration "vetorDeCaracter:vetor[0..3] de caracter" (line 9), the assignment "vetorDeCaracter[0] <- "blablabla"" (line 14), and the output statement "escreval("Na posição 0 o valor é:", vetorDeCaracter[0])" (line 16). A red circle also highlights the word "Inicio" (line 11).

Fonte: Elaborado pelo autor (2023).

Representação do vetor:



Fonte: Elaborado pelo autor (2023).

Veja acima como ficou o armazenamento dos dados no vetor **vetorDeCaracter**, na posição 0 está armazenado o valor “blablabla”, já, nas demais posições do vetor, por padrão para vetores do tipo caracteres receberão o valor inicial “espaço vazio”, podendo serem alterados quando assim desejarmos.

Abaixo segue a tela de execução:

```
c:\> Console simulando o modo texto do MS-DOS
Na posição 0 o valor é:blablabla
>>> Fim da execução do programa !
```

Fonte: Elaborado pelo autor (2023).

Exemplo 4: Criando um vetor do tipo lógico

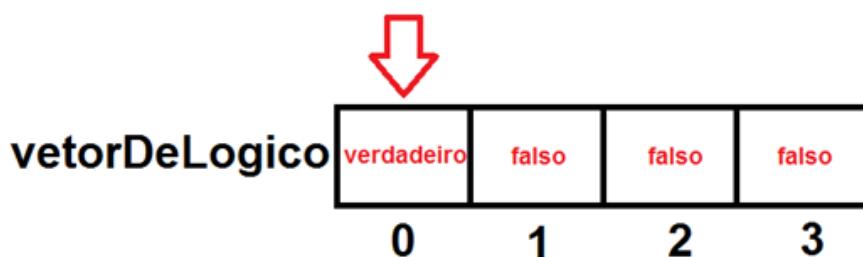
No exemplo abaixo, note que o processo de criação de vetores para o tipo “lógico” é parecido com os anteriores, a diferença é que declaramos o vetor como do tipo “lógico”, veja a linha 9 do algoritmo abaixo.

Note que na posição 0 do vetor foi atribuído o valor verdadeiro veja a linha 14:

```
1 Algoritmo "Exemplo4 vetores"
2 // Disciplina : [Linguagem e Lógica de Programação]
3
4
5 Var
6 // Seção de Declarações das variáveis
7
8 // Reservando 4 posições na memória para o vetor de lógico
9 vetorDeLogico:vetor[0..3] de lógico
10
11 Inicio
12 // Seção de Comandos, procedimento, funções, operadores, etc...
13
14 vetorDeLogico[0] <- verdadeiro
15
16 escreval("Na posição 0 o valor é:", vetorDeLogico[0])
17
18 Fimalgoritmo
```

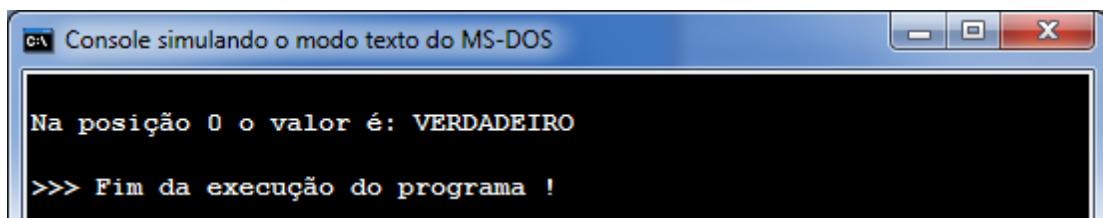
Fonte: Elaborado pelo autor (2023).

Representação do vetor:



Fonte: Elaborado pelo autor (2023).

Abaixo segue a tela de execução:



Fonte: Elaborado pelo autor (2023).

Exemplo 5: Exibindo as informações armazenadas nos vetores

A partir da linha 18, é possível visualizar uma maneira de mostrarmos as informações armazenadas num vetor. Esta não é a melhor forma de mostrarmos os dados de um vetor, pois além de ser trabalhosa, polui o código fonte com linhas desnecessárias.

```
1 Algoritmo "Exemplo5_exibindo"
2 // Disciplina : [Linguagem e Lógica de Programação]
3
4
5 Var
6 // Seção de Declarações das variáveis
7
8 // Reservando 4 posições na memória para o vetor de inteiros
9 vetorDeInteiros:vetor[0..3] de inteiro
10
11 Início
12 // Seção de Comandos, procedimento, funções, operadores, etc...
13
14 vetorDeInteiros[0] <- 2
15
16 //Exibindo os valores armazenados no vetor
17
18 escreval("Na posição 0 o valor é:", vetorDeInteiros[0])
19 escreval("Na posição 1 o valor é:", vetorDeInteiros[1])
20 escreval("Na posição 2 o valor é:", vetorDeInteiros[2])
21 escreval("Na posição 3 o valor é:", vetorDeInteiros[3])
22
23 Fimalgoritmo
```

Fonte: Elaborado pelo autor (2023).

Abaixo seguem os valores armazenados no vetor:

```
c:\ Console simulando o modo texto do MS-DOS
Na posição 0 o valor é: 2
Na posição 1 o valor é: 0
Na posição 2 o valor é: 0
Na posição 3 o valor é: 0
>>> Fim da execução do programa !
```

Fonte: Elaborado pelo autor (2023).

A melhor forma de exibir os valores armazenados nos vetores é da maneira abaixo:

No exemplo abaixo, foi criado um laço de repetição do tipo “Para” com o objetivo de ler os números inteiros digitados para as quatro posições de um determinado vetor. Veja entre as linhas 19 e 23. Outro laço de repetição foi criado para mostrar os valores armazenados no vetor, veja as linhas entre 29 e

31. Os laços de repetição utilizam uma variável como contador do tipo inteiro que será responsável por armazenar a posição de memória do vetor.

```
1 Algoritmo "Exemplo5_exibindo"
2 // Disciplina : [Linguagem e Lógica de Programação]
3
4
5 Var
6 // Seção de Declarações das variáveis
7
8 // Reservando 4 posições na memória para o vetor de inteiros
9 vetorDeInteiros:vetor[0..3] de inteiro
10 contador:inteiro
11
12 Inicio
13
14 // Seção de Comandos, procedimento, funções, operadores, etc...
15
16
17 //lendo os valores para o vetor
18
19 escreval("Digite os valores para o vetor:")
20
21 para contador de 0 ate 3 faca
22     leia (vetorDeInteiros[contador])
23 fimpara
24
25 //Exibindo os valores armazenados no vetor
26
27 limpatela
28
29 para contador de 0 ate 3 faca
30     escreval("Na posição", contador, " o valor é:", vetorDeInteiros[contador])
31 fimpara
32
33 Fimalgoritmo
```

Fonte: Elaborado pelo autor (2023).

Abaixo seguem as telas de execução do algoritmo:

```
c:\ Console simulando o modo texto do MS-DOS
Digite os valores para o vetor:
15
23
48
72|
```



```
c:\ Console simulando o modo texto do MS-DOS
Na posição 0 o valor é: 15
Na posição 1 o valor é: 23
Na posição 2 o valor é: 48
Na posição 3 o valor é: 72
>>> Fim da execução do programa !
```

Fonte: Imagens elaboradas pelo autor (2023).



ATIVIDADE DE FIXAÇÃO

1.O que são vetores?

2.Como referenciamos um elemento específico de um vetor?

3.Desenvolva um programa que utilizará um vetor para armazenar 5 números inteiros digitados pelo usuário, e depois mostre esses 5 números em tela.

Exemplo:

Digite 5 números

5

2

20

17

4

Fonte: Elaborado pelo autor (2023).

- Limpe a tela e exiba os números conforme o exemplo abaixo:

Os números

digitados são: 5

3

22

16

4

Fonte: Elaborado pelo autor (2023).

4.Desenvolva um programa que utilizará um vetor que receberá 5 números inteiros que serão gerados de forma aleatória (entre 0 e 100) e depois mostre esses números aleatórios armazenados no vetor em tela.

Exemplo:

Os números aleatórios gerados são:

15

100

48

2

0

Fonte: Elaborado pelo autor (2023).

5. Desenvolva um programa utilizando vetores que irão ler do usuário 5 nomes de alunos e 5 notas de AV1 e 5 notas de AV2. O algoritmo deverá calcular a média aritmética simples das notas dos alunos e exibi-las.

Primeiro passo: O algoritmo deve limpar a tela toda vez que o usuário digitar as notas de cada aluno, isso deve ser feito para não poluir a tela.

NOTAS DOS ALUNOS

Nome do aluno:

Rodolfo Nota AV1:

7

Nota AV2: 10

Fonte: Elaborado pelo autor (2023).

- Passo 2: Mostre os nomes e as notas dos alunos conforme exemplo abaixo:

MÉDIA DOS ALUNOS

Nome: Rodolfo	Nota AV1: 7	Nota AV2: 10	Média: 8,5
Nome: Vera	Nota AV1: 5	Nota AV2: 7	Média: 6
Nome: Daniela	Nota AV1: 7	Nota AV2: 6	Média: 6,5
Nome: Lenir	Nota AV1: 6	Nota AV2: 6	Média: 6
Nome: Leoni	Nota AV1: 7	Nota AV2: 9	Média: 8

6. Desenvolva um programa que irá ler 10 números inteiros digitados pelo usuário, esses números devem ser armazenados num vetor, em seguida mostre em tela somente os números pares armazenados no vetor.

Exemplo:

Digite 10 números:

50

60

20

15

19

8

12

15

110

54

Fonte: Elaborado pelo autor (2023).

Em seguida a tela deve ser apagada, e depois mostrar os números pares armazenados no vetor:

Os números pares armazenados no vetor
são: 50

60

20

8

12

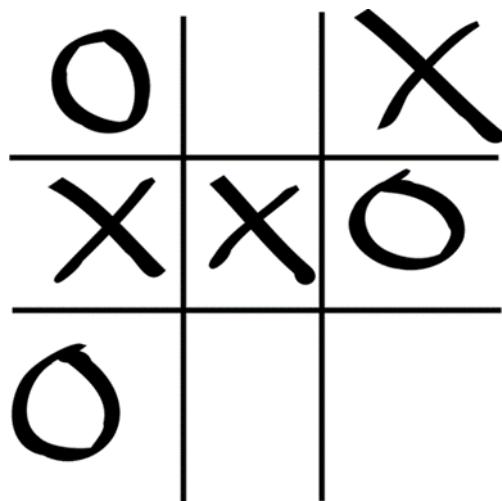
Fonte: Elaborado pelo autor (2023).

TEMA 06

Matrizes

Habilidades:

- Codificar programas utilizando técnica de programação estruturada.
- Depurar e versionar programas, utilizando ambiente de desenvolvimento integrado.



Fonte: Elaborado pelo autor (2023).

Fonte: <https://youtu.be/w7t0rJJINl8>

As matrizes são como se fossem conjuntos de vetores, em vez de termos um vetor em que podemos acessar as posições em apenas uma linha, com as matrizes podemos acessar várias linhas diferentes. Assim como os vetores, as matrizes podem armazenar dados de números inteiros, reais, caractere e lógico. Imagine por exemplo, que queiramos armazenar três notas de 40 alunos, então poderíamos usar uma matriz para isso, segue exemplo:

	0	1	2
0	10	7,5	5,5
1	5	3,5	7,5
2	8,5	6,5	10,0
40	2,5	1,5	0

Fonte: Elaborado pelo autor (2023).

Cada posição da matriz é representada por um índice, na matriz temos dois índices um na vertical e outro na horizontal.

Exemplo 1

Para criarmos uma matriz no VISUALG é necessário declará-la na seção de declarações de variáveis.

- No exemplo abaixo é possível notar na linha 7 que foi criado uma matriz com o nome de matriz, como um conjunto de dois vetores que podem armazenar até 4 números inteiros cada.
- Entre a linha 15 e 18, a linha 0 da matriz receberá 4 números inteiros para cada índice da linha 0.
- Entre a linha 21 e 24, a linha 1 da matriz receberá 4 números inteiros para cada índice da linha 1.
- Para exibir os valores armazenados na matriz, utilizamos laços de repetição do tipo PARA, é possível visualizar o laço a partir da linha 28.

```

1 Algoritmo "Exemplo1_Matriz"
2 // Disciplina : [Linguagem e Lógica de Programação]
3
4 Var
5 // Seção de Declarações das variáveis
6
7 matriz : vetor[0..3,0..3] de inteiro ←
8 contador1,contador2: inteiro
9
10 Inicio
11 // Seção de Comandos, procedimento, funções, operadores, etc...
12
13 //Linha 0 ←
14
15 matriz [0,0] <- 0 ←
16 matriz [0,1] <- 1 ←
17 matriz [0,2] <- 0 ←
18 matriz [0,3] <- 1 ←
19
20 //Linha 1 ←
21 matriz [1,0] <- 0 ←
22 matriz [1,1] <- 0 ←
23 matriz [1,2] <- 1 ←
24 matriz [1,3] <- 0 ←
25
26 //Exibindo os valores da matriz
27
28 para contador1 de 0 ate 3 faca ←
29   escreval (matriz[0,contador1]) ←
30 fimpara ←
31
32 para contador1 de 0 ate 3 faca ←
33   escreval (matriz[1,contador1]) ←
34 fimpara ←
35
36 Fimalgoritmo

```

Fonte: Elaborado pelo autor (2023).

Abaixo segue a tela de execução:

```

0
1
0
1
0
0
1
0
>>> Fim da execução do programa !

```

Fonte: Elaborado pelo autor (2023).

Exemplo 2:

No exemplo anterior, perceba que a visualização dos dados da matriz é ruim, para melhorar a visualização é necessário alterar o laço de repetição utilizado para mostrar os dados armazenados na matriz.

Entre as linhas 28 e 33 há um laço de repetição PARA dentro de outro laço de repetição PARA. Além disso, está sendo utilizado o comando escreva em vez de escreval, ou seja, a cada repetição os números da linha serão exibidos um do lado do outro e quando acabar o laço o comando escreval que está na linha 32 fará o pulo de linha. As variáveis contador1 e contador2 são os índices da matriz, através deles percorremos as linhas e colunas da matriz através dos laços de repetição.

```
1 Algoritmo "Exemplo1_Matriz"
2 // Disciplina : {Linguagem e Lógica de Programação}
3
4 Var
5 // Seção de Declarações das variáveis
6
7 matriz : vetor[0..3,0..3] de inteiro
8 contador1,contador2: inteiro
9
10 Inicio
11 // Seção de Comandos, procedimento, funções, operadores, etc...
12
13 //Linha 0
14
15 matriz [0,0] <- 0
16 matriz [0,1] <- 1
17 matriz [0,2] <- 0
18 matriz [0,3] <- 1
19
20 //Linha 1
21 matriz [1,0] <- 0
22 matriz [1,1] <- 0
23 matriz [1,2] <- 1
24 matriz [1,3] <- 0
25
26 //Exibindo os valores da matriz
27
28 para contador1 de 0 ate 1 faca
29     para contador2 de 0 ate 3 faca
30         escreva (matriz[contador1,contador2])
31     fimpara
32     escreval(" ")
33 fimpara
34
35 Fimalgoritmo
```



Fonte: Elaborado pelo autor (2023).

Abaixo segue a tela de execução, note que a visualização dos dados da matriz desta maneira facilita relacionarmos os valores armazenados nos índices da matriz.

```
c:\> Console simulando o modo texto do MS-DOS
0 1 0 1
0 0 1 0
>>> Fim da execução do programa !
```

Fonte: Elaborado pelo autor (2023).

Exemplo 3:

No exemplo abaixo, foi alterado o algoritmo para ler os números digitados pelo usuário, é importante ressaltar que os laços de repetição utilizados para ler os números são praticamente os mesmos daqueles utilizados para exibir os dados armazenados na matriz, basta comparar as linhas entre 15 e 20 com as linhas entre 24 e 29, perceba que o que muda é que numa terá o comando leia e na outra teremos o comando escreva para exibir os valores da matriz.

```
1 Algoritmo "Exemplo1_Matriz"
2 // Disciplina : [Linguagem e Lógica de Programação]
3
4 Var
5 // Seção de Declarações das variáveis
6
7 matriz : vetor[0..3,0..3] de inteiro
8 contador1,contador2: inteiro
9
10 Inicio
11 // Seção de Comandos, procedimento, funções, operadores, etc...
12
13 //Lendo os valores da matriz
14
15 para contador1 de 0 ate 1 faca
16   para contador2 de 0 ate 3 faca
17     leia (matriz[contador1,contador2])
18   fimpara
19   escreval(" ")
20 fimpara
21
22 //Exibindo os valores da matriz
23
24 para contador1 de 0 ate 1 faca
25   para contador2 de 0 ate 3 faca
26     escreva (matriz[contador1,contador2])
27   fimpara
28   escreval(" ")
29 fimpara
30
31 Fimalgoritmo
```

Fonte: Elaborado pelo autor (2023).

Abaixo segue a tela de execução do nosso exemplo:

```
24  
55  
69  
27  
  
56  
44  
11  
28  
  
24 55 69 27  
56 44 11 28  
  
>>> Fim da execução do programa !
```

Fonte: Elaborado pelo autor (2023).



ATIVIDADE DE FIXAÇÃO

1. O que são matrizes?
2. Como podemos referenciar um elemento específico de uma matriz?
3. Qual é a utilidade de uma matriz?
4. Desenvolva um programa que carregue uma matriz 2(linhas) x10(colunas) os nomes dos alunos de uma sala, a sala está dividida em dois grupos de 10 alunos. Após isso, devem ser apresentados os nomes dos alunos dos grupos.
5. Desenvolva um programa com uma matriz 3x3, atribua a cada índice da matriz um valor aleatório (entre 0 e 50). Em seguida imprima todos os dados da matriz e depois somente os valores armazenados na diagonal principal da matriz.

Exemplo:

Os dados da matriz são:

15	22	19
13	35	49
6	12	25

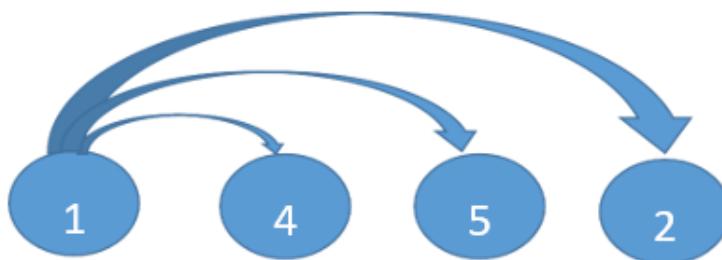
Fonte: Elaborado pelo autor (2023).

- Apague a tela e em seguida mostre os valores localizados na diagonal principal.

Algoritmos de ordenação

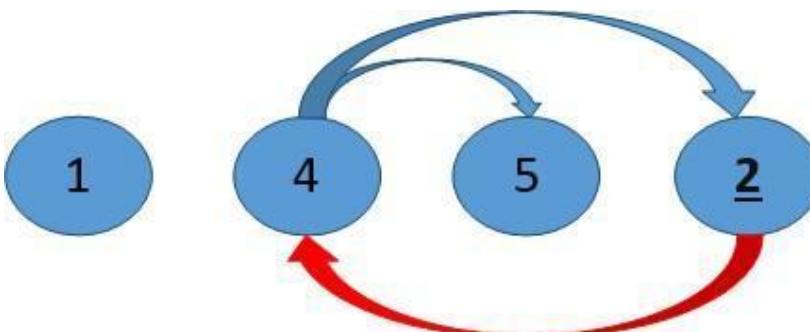
Ordenar significa colocar elementos em uma ordem específica. Podemos ordenar números e palavras de forma crescente (do menor para o maior) ou decrescente (do maior para o menor). A ordenação dos dados pode ser útil para facilitar a visualização e o acesso às informações.

Vamos falar sobre um algoritmo de ordenação simples e conhecido, chamado Bubble Sort (ou ordenação por flutuação). O Bubble Sort percorre várias vezes uma lista de elementos e, a cada passagem, faz o maior elemento "flutuar" para o topo da sequência. Esse movimento de "flutuação" é como bolhas subindo à superfície, por isso o nome Bubble Sort. Ele é usado principalmente quando temos uma pequena quantidade de dados para ordenar.



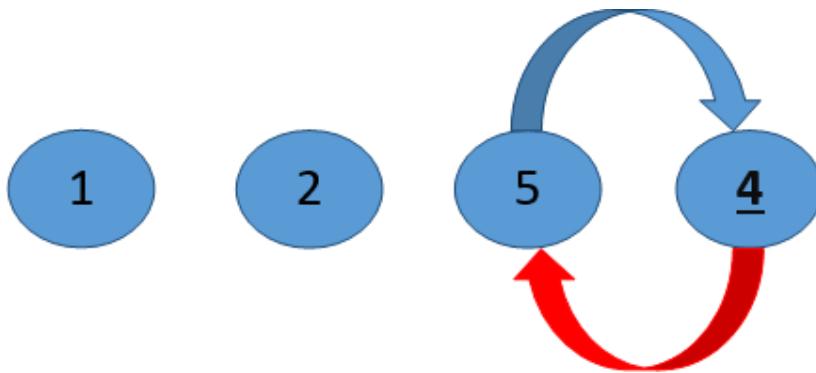
Fonte: Elaborado pelo autor (2023).

Após a bola 1 ter sido comparada com todas as bolas, a próxima a ser comparada com as demais é a bola 4 que está na segunda posição do vetor. Note que 4 não é maior que 5 então não será feita a troca entre essas bolas, porém 4 é maior que 2, então os valores de 4 e 2 são trocados.



Fonte: Elaborado pelo autor (2023).

O que restou foi comparar o penúltimo valor com o último, no caso comparar o 5 com o 4, como 4 é menor que 5 então será feita a troca.



Fonte: Elaborado pelo autor (2023).

Pronto! Agora temos o nosso vetor ordenado.



Fonte: Elaborado pelo autor (2023).

Algoritmo de ordenação “Bubble Sort” parte 1/2:

```
1 Algoritmo "Algoritmo de ordenação"
2 // Disciplina : [Linguagem e Lógica de Programação]
3
4 Var
5 // Seção de Declarações das variáveis
6
7 vet:vetor[1..6] de inteiro
8 contador1, contador2,troca:inteiro
9
10 Inicio
11 // Seção de Comandos, procedimento, funções, operadores, etc...
12 escreval ("Números fora de ordem ")
13
14 //gerando os números aleatórios
15 para contador1 de 1 ate 6 faça
16
17     vet[contador1] <- randi(100)
18
19 fimpara
20
21 //Visualizando os números aleatórios gerados
22
23 para contador1 de 1 ate 6 faça
24     escreval (vet[contador1])
25 fimpara
26
27 escreval (" ")
28 escreval (" ")
```

Fonte: Elaborado pelo autor (2023).

Continuação do algoritmo parte 2/2:

```
29
30 //Aqui será feita a ordenação do vetor
31 para contador1 de 1 ate 5 faca
32     para contador2 de contador1+1 ate 6 faca
33
34     // Comparando os valores
35     se vet[contador1] > vet [contador2] entao
36         troca <- vet[contador1]
37         vet[contador1] <- vet[contador2]
38         vet[contador2] <- troca
39
40     fimse
41
42
43     fimpara
44
45 fimpara
46
47 escreval ("Números ordenados ")
48 escreval (" ")
49
50 para contador1 de 1 ate 6 faca
51     escreval (vet[contador1])
52 fimpara
53
54 Fimalgoritmo
```

Fonte: Elaborado pelo autor (2023).

Abaixo segue a tela de execução:

```
Números fora de ordem
7
59
1
16
2
26

Números ordenados

1
2
7
16
26
59

>>> Fim da execução do programa !
```

Fonte: Elaborado pelo autor (2023).



ATIVIDADE DE FIXAÇÃO

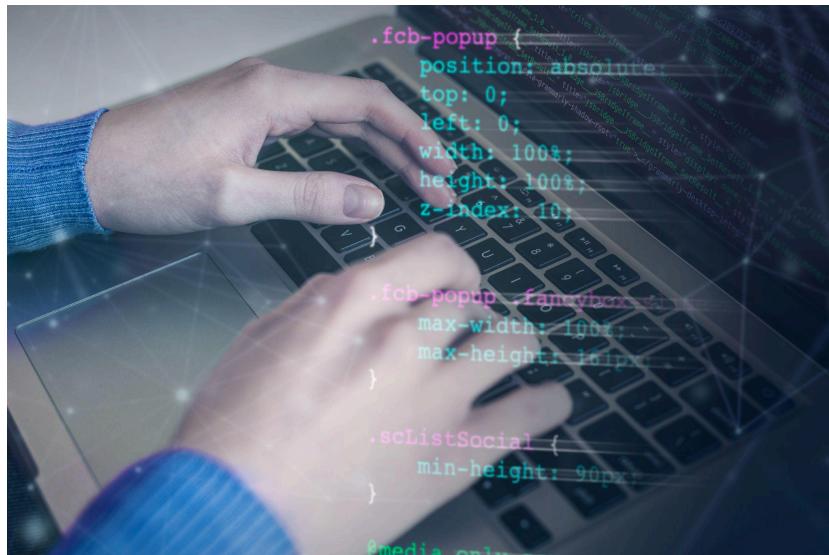
1. Explique como funciona o algoritmo de ordenação Bubble Sort.
2. Com base no funcionamento do algoritmo Bubble Sort, é correto afirmar que ele é ideal para ordenar pequenas listas de números? Justifique.
3. Desenvolva um programa que irá ler 10 números do usuário, utilize o Bubble Sort para ordená-los.
4. Desenvolva um programa utilizando o algoritmo de ordenação Bubble Sort para ordenar 10 números inseridos pelo usuário. Porém, após ordenar a lista, devem ser mostrados em ordem somente os números pares.
5. Desenvolva um programa que irá ler do usuário 10 nomes e 10 notas dos alunos. Utilize o algoritmo de ordenação Bubble Sort para ordenar as notas, importante ressaltar que os respectivos nomes dos alunos devem ser ordenados junto com as notas. Mostre os valores ordenados.
6. Pesquise e cite ao menos 4 algoritmos de ordenação.

TEMA 07

Procedimento

Habilidades:

- Codificar programas, utilizando técnica de programação estruturada.
- Depurar e versionar programas, utilizando ambiente de desenvolvimento integrado.



Disponível em <<https://tinyurl.com/4r92fpfh>>. Acesso em 20 jul. 2023.

Podemos dividir um grande programa em porções menores denominadas de módulos, esses módulos podem consistir em arquivos de código-fonte ou apenas rotinas. Essas rotinas podem ser classificadas em dois tipos: funções e procedimentos. Neste tema abordaremos os procedimentos. Os procedimentos são rotinas que executam uma determinada tarefa, utilizando ou não argumentos recebidos, sem retornar um valor.

Quando temos um programa grande, podemos dividir em partes menores chamadas de módulos. Esses módulos podem ser arquivos de código ou apenas rotinas. Existem dois tipos de rotinas: funções e procedimentos. Neste tema, vamos falar sobre os procedimentos. Os procedimentos são rotinas que realizam uma tarefa específica, podendo receber ou não argumentos, mas não retornam um valor.

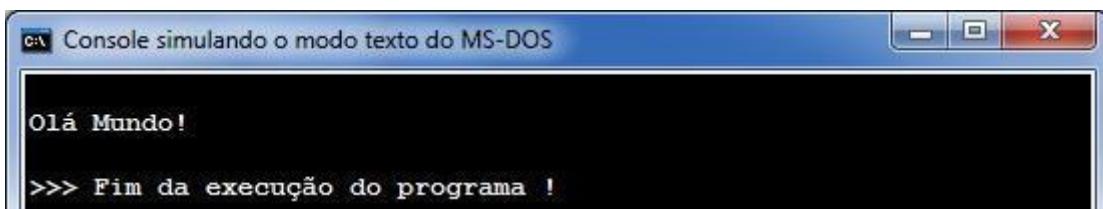
Exemplo 1:

Os procedimentos são criados fora da seção Início, veja que entre as linhas 8 e 11 foi criado um procedimento de nome “saudação”, toda vez que ele for chamado aparecerá uma mensagem “Olá Mundo! ”.

```
1 Algoritmo "Exemplo1_Procedimento"
2 // Disciplina : [Linguagem e Lógica de Programação]
3
4 Var
5 // Seção de Declarações das variáveis
6
7 //Criando um procedimento
8 procedimento saudacao ←
9 inicio
10     escreval("Olá Mundo!")
11 fimprocedimento
12
13 Inicio
14 // Seção de Comandos, procedimento, funções, operadores, etc...
15
16 //chamando um procedimento
17 saudacao ←
18
19 Fimalgoritmo
```

A chamada do procedimento está sendo feita na linha 17.

Fonte: Elaborado pelo autor (2023).



Fonte: Elaborado pelo autor (2023).

Exemplo 2: Procedimento com parâmetros

Parâmetros são valores decorrentes de uma variável ou de uma expressão, que pode ser enviado para um procedimento ou função.

No caso abaixo queremos que o procedimento receba o nome do usuário e faça a saudação personalizada para o usuário.

Note que na linha 7, declaramos uma variável chamada de “nome” como do tipo caracter.

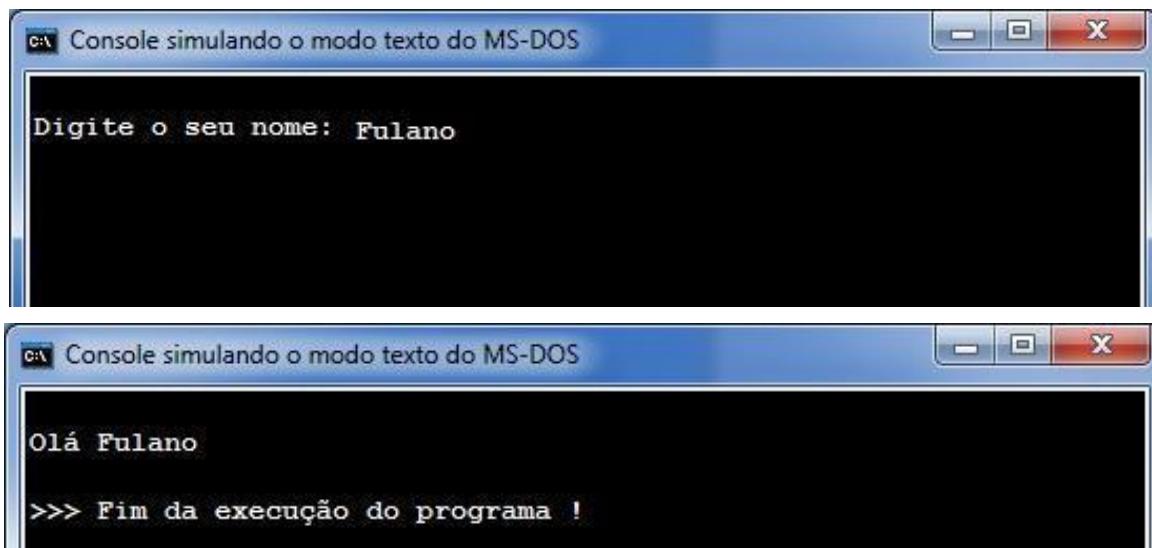
Na linha 25 lemos o nome digitado pelo usuário e na linha 31 chamamos o procedimento “saudacao” passando o nome do usuário como parâmetro.

```
1 Algoritmo "Exemplo_Procedimento_Parametros"
2 // Disciplina : [Linguagem e Lógica de Programação]
3
4 Var
5 // Seção de Declarações das variáveis
6
7 nome:caracter ←
8
9 //Criando um procedimento com parâmetros
10
11 procedimento saudacao (nomeUsuario:caracter)
12 inicio
13     escreval("Olá ",nomeUsuario)
14 fimprocedimento
15
16
17 Inicio
18 // Seção de Comandos, procedimento, funções, operadores, etc...
19
20
21
22 Escreva("Digite o seu nome: ")
23
24 //Lendo o nome do usuário
25 leia(nome) ←
26
27 //chamando um procedimento
28
29 limpatela
30
31 saudacao (nome)
32
33 Fimalgoritmo
```

Entre as linhas 11 e 14 está o nosso procedimento “saudacao”.

Fonte: Elaborado pelo autor (2023).

Abaixo seguem as telas de execução:



Fonte: Imagens elaboradas pelo autor (2023).

Exemplo 3: Procedimento utilizando variáveis globais

Neste exemplo, é possível verificar que o procedimento procComVariavelGlobal faz o incremento de mais 1 para a variável, variavelGlobal, quando o mesmo é acionado (linha 13).

Note que na linha 21 a variavelGlobal recebe o valor de 0, e logo em seguida na linha 23 é solicitado a exibição em tela do valor da variável, que no caso é 0.

Porém, note que ao chamarmos o procedimento “procComVariavelGlobal” (linha 27), como mencionado anteriormente o procedimento irá somar o valor de 1 ao valor armazenado na variável variavelGlobal, ocorre que ao exibirmos novamente o valor da variável “variavelGlobal” a mesma possuirá o valor atualizado de 1. Veja a tela de execução desse algoritmo.

Ou seja, ao utilizarmos variáveis globais nos procedimentos, temos que ter em mente que o valor da variável poderá ser alterado a cada chamada do procedimento.

Tela de execução:

```
Console simulando o modo texto do MS-DOS
O valor da variável global atual é: 0
O valor da variável global atual é: 1
>>> Fim da execução do programa !
```

Fonte: Elaborado pelo autor (2023).

Exemplo 4: Procedimento com variável local

Para declarar uma variável local dentro de um procedimento ou função, basta utilizar a tag “Var” e criar a variável dentro dessa seção, veja entre as linhas 9 e 11.

```
1 Algoritmo "Exemplo4_proc_variavel_global"
2 // Disciplina : [Linguagem e Lógica de Programação]
3
4 Var
5 // Seção de Declarações das variáveis
6
7 variavelGlobal:inteiro
```

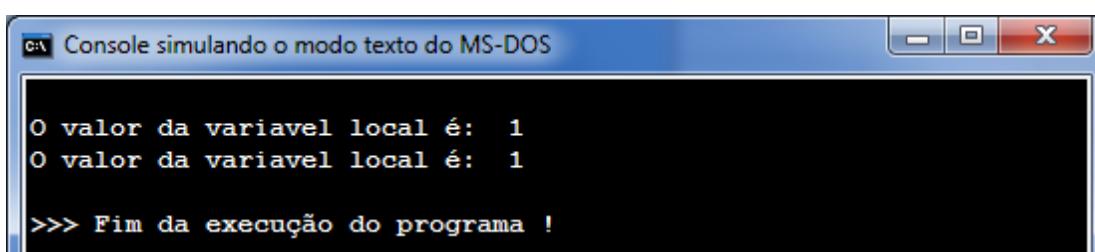
Fonte: Elaborado pelo autor (2023).

O restante do algoritmo é igual ao do exemplo anterior, a única diferença é que estamos utilizando uma variável local (“variavelLocal”).

```
1 Algoritmo "Exemplo5_proc_variavel_global"
2 // Disciplina : [Linguagem e Lógica de Programação]
3
4 Var
5 // Seção de Declarações das variáveis
6
7 procedimento incrementaVariavelLocal
8
9 var
10
11 variavelLocal:inteiro ←
12
13 inicio
14     variavelLocal <- variavelLocal+1
15
16     escreval ("O valor da variável local é: ",variavelLocal)
17
18 fimprocedimento
19
20
21 Inicio
22 // Seção de Comandos, procedimento, funções, operadores, etc...
23
24 // chamando o procedimento que utiliza variável local
25
26 incrementaVariavelLocal ←
27
28 // chamando pela segunda vez o mesmo procedimento
29
30 incrementaVariavelLocal ←
31
32 Fimalgoritmo
```

Fonte: Elaborado pelo autor (2023).

Note que na tela de execução, não importa quantas vezes chamamos o procedimento “incrementaVariavelLocal” sempre o valor será o mesmo (nesse algoritmo), pois cada vez que ele é chamado a variável terá sempre 0 e depois será incrementado +1.



Fonte: Elaborado pelo autor (2023).



ATIVIDADE DE FIXAÇÃO

1. O que são módulos?
2. O que são procedimentos?
3. O que são parâmetros?
4. Com base no exemplo 2, qual é a utilidade em passarmos parâmetros para os procedimentos?
5. Explique o que você entendeu sobre os exemplos 3 e 4, onde num exemplo o procedimento utiliza uma variável global e no outro é utilizado uma variável local.
6. Desenvolva um programa que possuirá o menu de um sistema bancário, conforme o exemplo abaixo, porém, para cada tela do menu deverá ser criado um procedimento, ou seja, um procedimento para o menu principal, outro para saldo, extratos e empréstimos.

NOME DO BANCO
CLIENTE: Nome do cliente
SELECIONE UMA OPÇÃO:
1- Saldo
2- Extratos (Últimos 5 dias)
3- Empréstimos

Fonte: Elaborado pelo autor (2023).

- Se o cliente escolher a opção 1, deverá apagar a tela e em seguida aparecer uma tela com um saldo fictício.
- Se o cliente escolher a opção 2, deverá apagar a tela e em seguida o sistema deverá mencionar os gastos (fictícios) mais recentes do cliente.
- Se o cliente escolher a opção 3, deverá apagar a tela e em seguida aparecer uma mensagem: “Procure a sua agência para a liberação de empréstimos.”
- Criar uma consistência fazendo que o usuário digite apenas números entre 1 e 3, se o usuário digitar um número fora desse intervalo, uma mensagem de erro deverá ser apresentada com os dizeres: “Opção inválida! ” E em seguida deve ser solicitada a opção novamente para o usuário até que ele digite uma opção válida.
- OBS: Tire um Print Screen do código fonte e também da tela de execução do programa.

Funções



Disponível em <<https://tinyurl.com/yxa8ukau>>. Acesso em 20 jul. 2023.

As funções em programação são similares às existentes na Matemática, ou seja, elas operam com valores passados como parâmetros e devolvem um valor resultante dos cálculos. Assim como os procedimentos, as funções são ótimas para atribuirmos para elas as tarefas repetitivas.

Por exemplo, podemos criar uma função para calcular a média, então toda vez que for necessário calcular a média poderemos utilizar essa função.

Exemplo1: Funções sem parâmetros

Assim como nos procedimentos, as funções são criadas fora da seção “Início”, conforme a imagem abaixo, veja as linhas entre 9 e 14.

Note que após ser chamada retornará um número inteiro, no caso será o número 15 (linha 12). Na linha 20 é feita a chamada da função `retornalntero`.

E na linha 23 é exibido o valor retornado pela função.

```
1 Algoritmo "Exemplo1_Funcoes"
2 // Disciplina : [Linguagem e Lógica de Programação]
3
4 Var
5 // Seção de Declarações das variáveis
6
7 valor1:inteiro
8
9 funcao retornaInteiro:inteiro
10 inicio
11
12     retorne 15
13
14 fimfuncao
15
16 Inicio
17 // Seção de Comandos, procedimento, funções, operadores, etc...
18
19 // Atribui para a variável valor1 o retorno da função retornaInteiro
20 valor1 <- retornaInteiro ←
21
22 // Abaixo escreveremos na tela o valor atribuído na variável pela função
23 escreval ("O valor é: ", valor1) ←
24
25 Fimalgoritmo
26
27
```

Fonte: Elaborado pelo autor (2023).

Segue a tela de execução abaixo:

```
Console simulando o modo texto do MS-DOS
O valor é: 15
>>> Fim da execução do programa !
```

Fonte: Elaborado pelo autor (2023).

Exemplo 2: Funções com parâmetros

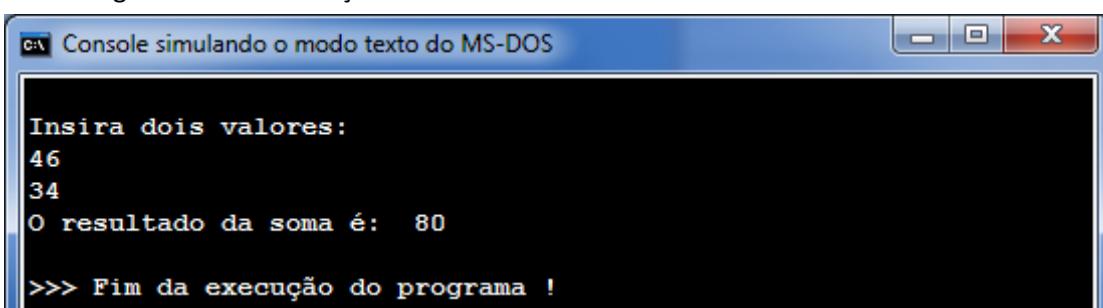
Já no exemplo abaixo, são lidos dois números inteiros através do usuário (linha 25), esses números são passados como parâmetros na chamada da função (linha 29).

A função retornaSoma, recebe os dois números e calcula a soma (da linha 12 até 17). Após, o valor da soma retornada pela função retornaSoma é exibido em tela (linha 31).

```
1 Algoritmo "Exemplo2_Funcoes_com_parâmetros"
2 // Disciplina : [Linguagem e Lógica de Programação]
3
4 Var
5 // Seção de Declarações das variáveis
6
7 resultado,valor1,valor2:inteiro
8
9
10 // Função com parâmetros
11
12 funcão retornaSoma (a,b:inteiro):inteiro
13 inicio
14
15     retorne a + b
16
17 fimfunção
18
19 Inicio
20 // Seção de Comandos, procedimento, funções, operadores, etc...
21
22
23 escreval ("Insira dois valores:")
24 //recebendo os valores
25 leia (valor1,valor2) ←
26
27 // Chamando a função retornaSoma passando dois parâmetros
28 // A função retornaSoma irá retornar a soma desses dois parâmetros
29 resultado <- retornaSoma (valor1, valor2) ←
30
31 escreval ("O resultado da soma é: ", resultado)
32
33
34 Fimalgoritmo
```

Fonte: Elaborado pelo autor (2023).

Abaixo segue a tela de execução:



Fonte: Elaborado pelo autor (2023).



ATIVIDADE DE FIXAÇÃO

1. O que são funções?
2. Qual a diferença entre uma função e um procedimento?
3. Com base no exemplo 2, desenvolva um programa no VISUALG que irá ler dois valores inteiros. Crie uma função para cada uma das seguintes operações matemáticas: soma, subtração, multiplicação e divisão.
 - No final, o resultado das operações deverá ser exibido em tela.
4. Desenvolva um programa que possuirá uma função chamada de pagamento, ela deverá receber as horas trabalhadas e o valor das horas. No final o valor a ser pago para o trabalhador deverá ser retornado pela função e depois exibido em tela.
5. Desenvolva um programa que possuirá uma função que receba o nome do usuário e a idade dele, a função deverá retornar quantos dias de vida o usuário já viveu. Por exemplo:

DIAS DE VIDA

Digite o seu nome: Maria

Digite a sua idade:34

Fonte: Elaborado pelo autor (2023).

- Limpe a tela e faça aparecer a mensagem abaixo:

Olá Maria, você já viveu 12410 dias!

Fonte: Elaborado pelo autor (2023).

TEMA 08

Técnicas e dicas de programação

Habilidades:

- Codificar programas utilizando técnica de programação estruturada.
- Depurar e versionar programas, utilizando ambiente de desenvolvimento integrado.



Fonte: Elaborado pelo autor (2023).

Fonte: https://youtu.be/DmSNs_cmTQQ

Para que os projetos de desenvolvimento de softwares ocorram bem, é necessário que o programador siga os padrões e boas práticas estabelecidas, assim, evitando ou minimizando erros decorrentes de falta de planejamento ou padronização de código.

É sempre bom o programador ter em mente que outros programadores poderão mexer no código do seu programa, e para facilitar esta tarefa de entendimento do código nada melhor que seguir as normas.

Identificadores

Em programação, um identificador é um nome atribuído a uma variável de memória, uma estrutura de dados, uma classe, um objeto, um procedimento, uma função, um comando ou palavra reservada da linguagem.

Algumas regras são estabelecidas para definição de identificadores, que variam conforme a linguagem, tais como:

- Um identificador pode ter até 32 caracteres de comprimento.
- O primeiro caractere deve ser uma letra do alfabeto.
- Os demais caracteres podem ser letras, números ou o sinal de sublinhado, também conhecido como underscore (_).
- Não são aceitos sinais de pontuação, caracteres acentuados, cedilha (ç) ou espaço em

branco.

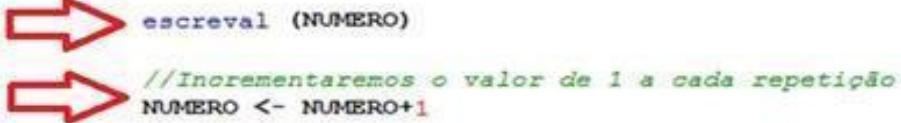
São exemplos válidos de identificadores: *Nome_Cliente*, *Data_Emissao* e *Valores_1_a_100*.

Não são exemplos válidos de identificadores: *Nome do cliente*, *Data_Emissão*,
MensagemAviso!!!, *2aSemana*

Indentação de Código

Conforme o exemplo abaixo, indentar o código significa recuar partes do código a fim de facilitar o entendimento do código. Um código sem indentação torna maçante o processo de entendimento.

```
1 Algoritmo "Laços de repetição"
2 // Discipline : {Linguagem e Lógica de Programação}
3
4 Var
5 // Seção de Declarações das variáveis
6
7 NUMERO:inteiro
8
9 Inicio
10 // Seção de Comandos, procedimento, funções, operadores, etc...
11
12 // Nesta parte do código a variável NUMERO receberá o valor "0".
13 NUMERO<-0
14
15
16 // Aqui começará o laço de repetição enquanto
17 enquanto (NUMERO < 5) faça
18
19     escreval (NUMERO)
20
21     // Incrementaremos o valor de 1 a cada repetição
22     NUMERO <- NUMERO+1
23
24
25 // O laço de repetição terminará quando a condição não for mais atendida.
26 fimenquanto
27
28
29 Fimalgoritmo
```



Fonte: Elaborado pelo autor (2023).

Testes

Efetuar vários testes nos programas é uma boa prática. Criar testes automatizados também deve ser considerado.

Não poupe esforços para testar um programa antes de entregar ao cliente.

Fluxogramas

Como visto no tema sobre fluxogramas, os fluxogramas são muito úteis nas etapas iniciais dos projetos, pois nos dão a real dimensão de como deverá funcionar o programa.

Comentários

Dependendo da linguagem de programação a forma de comentar o código mudará. Abaixo segue um exemplo em como podemos utilizar os comentários.

```
16 // Aqui começará o laço de repetição enquanto
17 enquanto (NUMERO < 5) faça
18
19     escreval (NUMERO)
20
21     //Incrementaremos o valor de 1 a cada repetição
22     NUMERO <- NUMERO+1
23
24
25 //O laço de repetição terminará quando a condição não for mais atendida.
26 fimenquanto
```

Fonte: Elaborado pelo autor (2023).

Note que comentar o código é uma boa prática pois facilita o processo de entendimento do código. Imagine que nem sempre será você que irá efetuar a manutenção do sistema.

Diminuir as linhas de código

Diminuir a quantidade de linhas de código, também é uma boa prática, muitas vezes menos é mais.

Por exemplo:

Em vez de escrever:

Leia (a) Leia (b)

Podemos escrever desta maneira:

Leia (a, b)

Fazer consistências de dados

Campos de formulários ou variáveis que recebem valores através da interação com usuário, devem receber consistência de dados.

Por exemplo:

Ao ler o nome de alguma pessoa, devemos consistir, por exemplo, se:

- A pessoa digitou alguma coisa;
- Se há somente espaços em branco;
- Se digitou números ou caracteres não esperados como: *&%#\$%#! Ao ler a idade de alguém, devemos consistir, por exemplo, se:
 - A idade é maior que zero;
 - Verificar se o usuário não digitou letras ou caracteres inválidos;

Tenha um controle de versão

Tenha um controle de versão do programa, a cada alteração do programa, seja organizado para manter o controle de versão atualizado especificando a nova versão e também registrando o que foi alterado na nova versão. Quando há muitos programadores trabalhando num mesmo sistema simultaneamente é importante considerar um software para fazer o controle de versão do sistema, pois o processo manual é suscetível a muitas falhas.

Idioma

Como muitas linguagens utilizam comandos em inglês, é recomendável ter um conhecimento no mínimo básico de inglês, assim você terá condições de interpretar melhor as mensagens de erro e poderá fazer pesquisas na internet em páginas de língua inglesa.



ATIVIDADE DE FIXAÇÃO

- 1.** É possível escrever nomes de variáveis desta maneira: Nome do cliente, Data_Emissão, MensagemAviso!!!, 2aSemana? Justifique.

- 2.** O que é indentação de código? Qual é a importância de indentar o código?

- 3.** Faça uma pesquisa sobre quais são os principais tipos de testes de software.

- 4.** Em qual etapa do projeto os fluxogramas são muito úteis?

- 5.** Qual a importância de se fazer comentários nos códigos dos programas?

- 6.** O que é validação de dados? Qual é a importância em validar os dados digitados pelos usuários?

- 7.** O que é controle de versão?

- 8.** Quais são os tipos de controle de versão? Pesquise.

- 9.** Por que é importante ter no mínimo o conhecimento básico de inglês para ser programador? Justifique.

TEMA 09

Conceitos básicos de Python

Habilidades:

- Codificar programas utilizando técnica de programação estruturada.
- Depurar e versionar programas, utilizando ambiente de desenvolvimento integrado.



Fonte: Acervo do autor (2023).

Criado por Guido van Rossum nos anos 1990, atualmente a linguagem é mantida nas versões 2.7, 3.4 e 3.7. Multiplataforma, o interpretador Python está disponível para Microsoft Windows e diversos Unix-like. Python é uma linguagem de alto nível, multiparadigma e interpretada, que pode ser executada em modo script e/ou em modo interativo. A sintaxe de Python é simples e fácil de ser aprendida. O que delimita um bloco na linguagem Python são a indentação, e não o bloco de chaves ({}), como é em C++ ou no uso de uma palavra reservada, como na linguagem Pascal.

Quando um programa é criado, o código-fonte é salvo em um arquivo-texto com a extensão apropriada para script Python (.py). No terminal, ao invocar o interpretador Python e o script ser criado, o interpretador executa o script e recebe as entradas de dados.

Começando a usar o Python

Instalando o Python

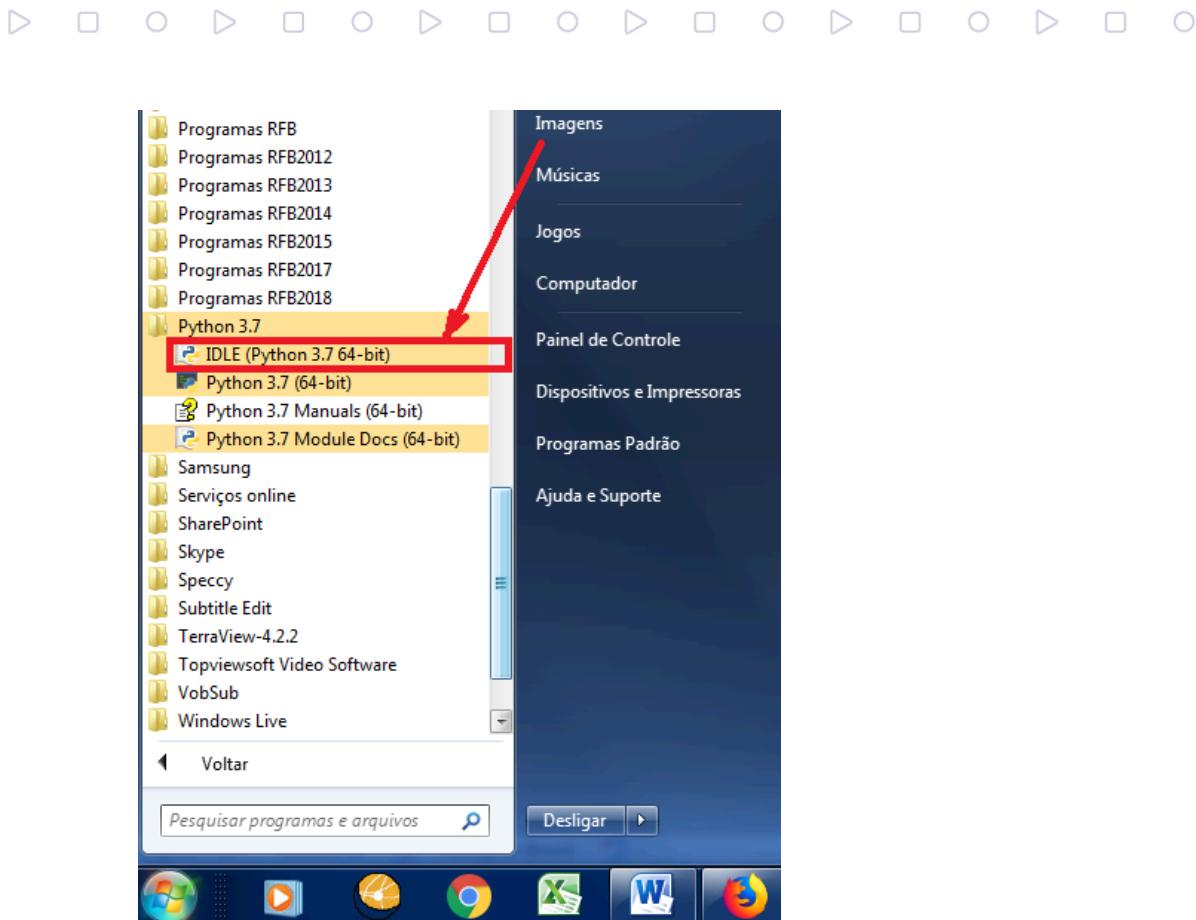
- Acesse o site: <https://www.python.org/>
- Clique em “Download”, será necessário baixar o instalador conforme a versão e tipo de sistema operacional.

OBS: O Sistema Linux já possui o Python instalado, basta executá-lo através do terminal.



Fonte: Elaborado pelo autor (2023).

O IDLE é o ambiente interativo de desenvolvimento.

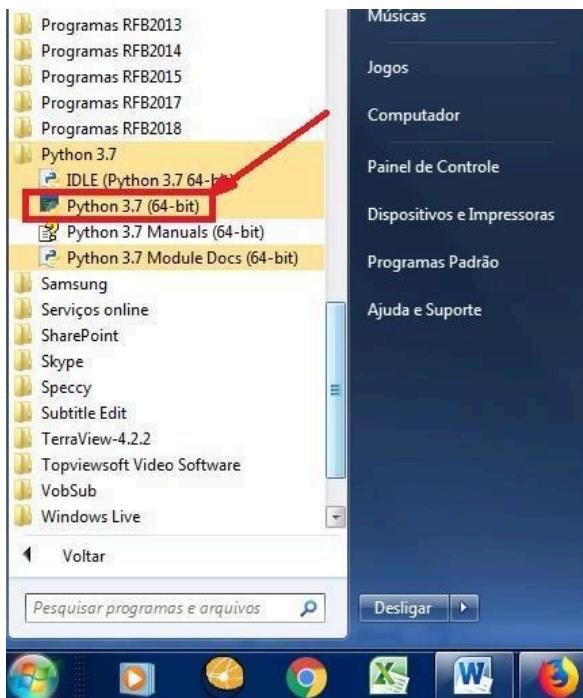


Fonte: Elaborado pelo autor (2023).

Abaixo segue a tela do IDLE (PYTHON SHELL):

A screenshot of the Python 3.7.0 Shell window. The title bar says 'Python 3.7.0 Shell'. The menu bar includes 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The main window displays the Python version information: 'Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32'. It also shows the prompt 'Type "copyright", "credits" or "license()" for more information.' followed by the user input '2' and the prompt '2>>>'. The window has standard Windows-style scroll bars on the right side.

Fonte: Elaborado pelo autor (2023).



O Python 3.7 é o interpretador.

Fonte: Elaborado pelo autor (2023).

Abaixo segue a tela do Python 3.7 (Interpretador):

A screenshot of the Python 3.7 interpreter window. The title bar says 'Python 3.7 (64-bit)'. The window displays the Python command line interface with the following text:

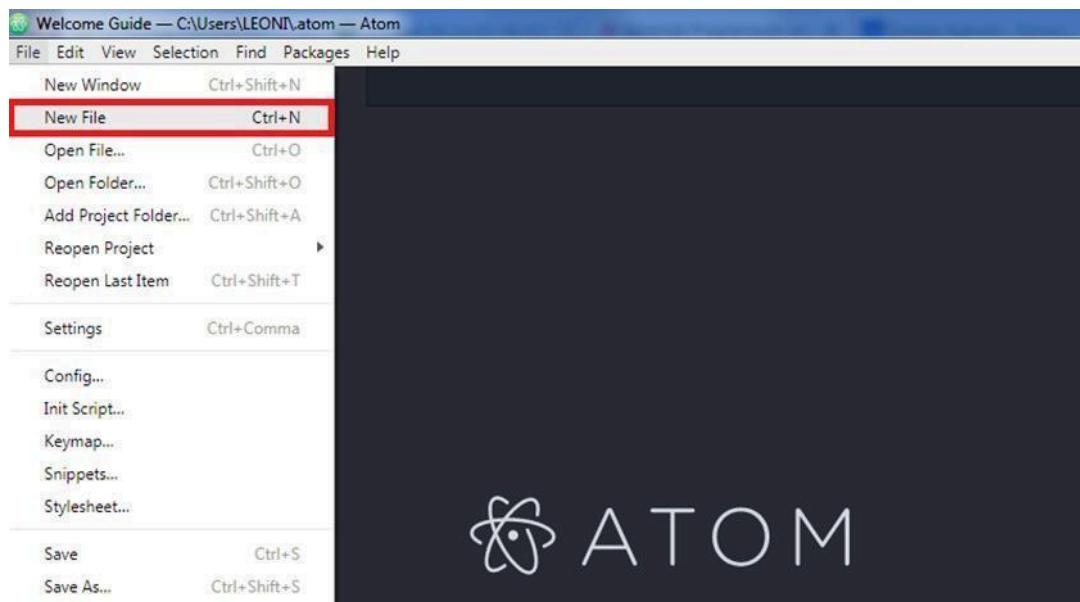
```
Python 3.7.0 <v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51> [MSC v.1914 64 bit (AMD64)]
4>]
Type "help", "copyright", "credits" or "license" for more information.
>>> 1+1
2
>>>
```

The window has a standard Windows-style border with minimize, maximize, and close buttons.

Fonte: Elaborado pelo autor (2023).

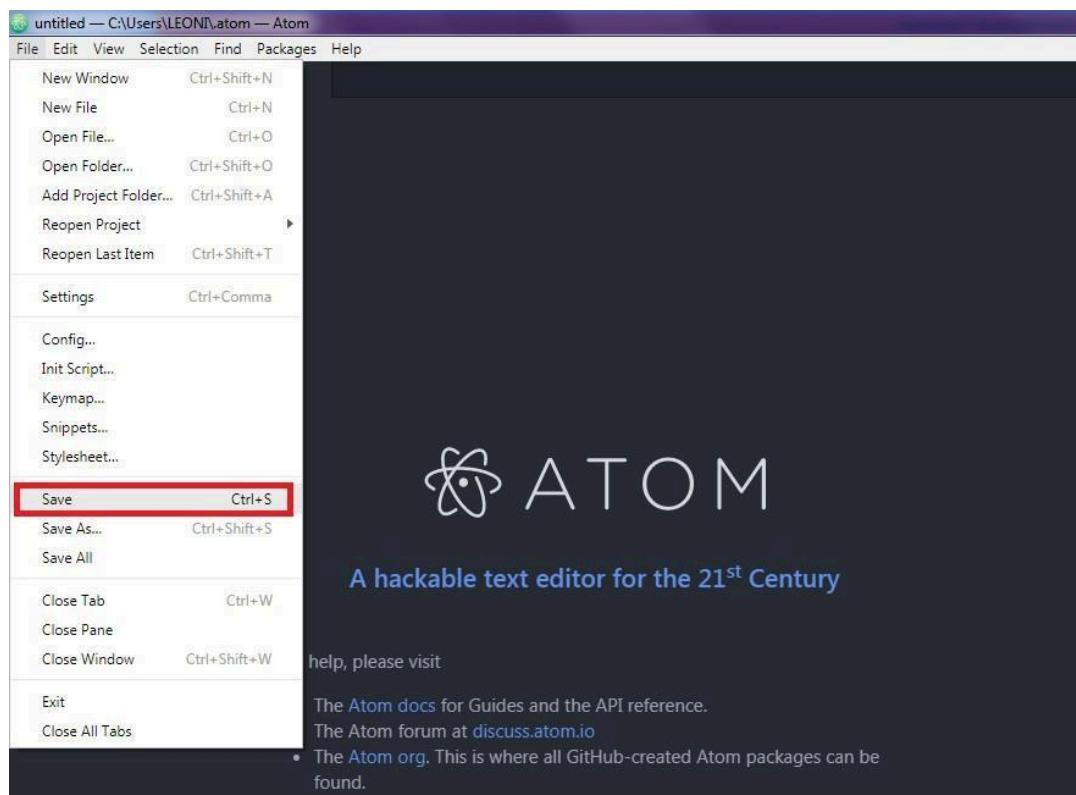
Criando o nosso primeiro programa

O Atom que é um editor de texto (poderoso), será utilizado como interpretador para Python, para isso, é necessário primeiro clicar em File => New File



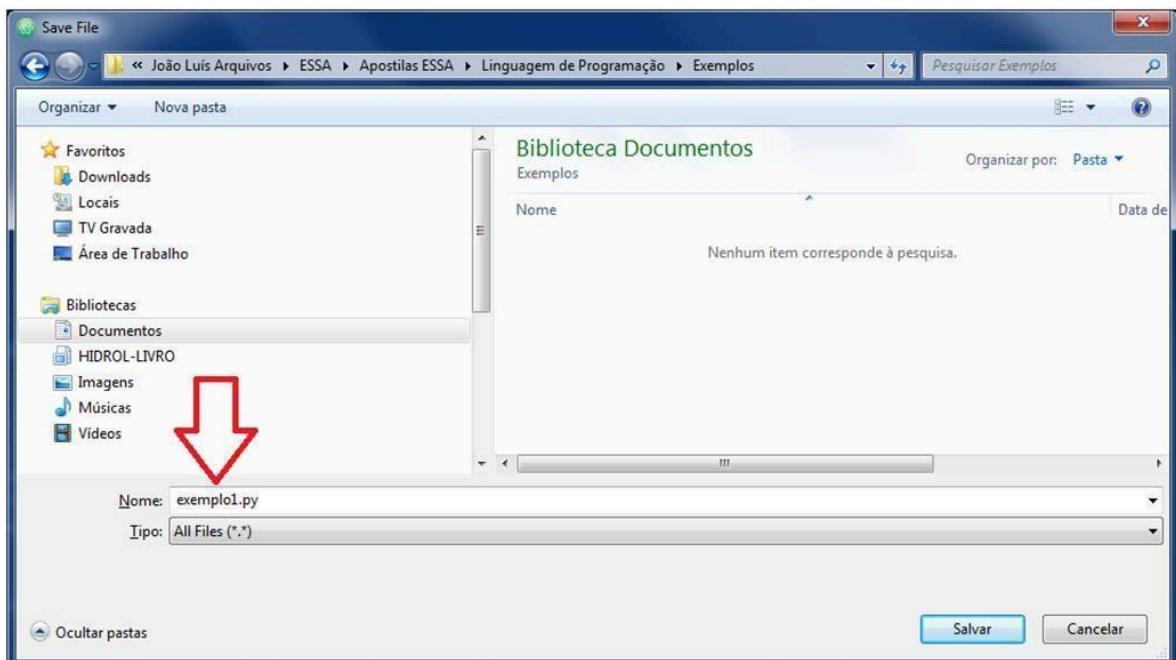
Fonte: Elaborado pelo autor (2023).

Depois clique em “Save”.



Fonte: Elaborado pelo autor (2023).

Salve o arquivo da seguinte maneira: “nome_do_arquivo.py”



Fonte: Elaborado pelo autor (2023).

Digite o comando print ('Olá mundo'), conforme abaixo:

```
exemplo1.py — C:\Python\Exemplos — Atom
File Edit View Selection Find Packages Help
exemplo1.py
1 print('Olá mundo!')
2
```

Fonte: Elaborado pelo autor (2023).

Para executar o programa no Atom basta pressionar o botão “F5” do teclado.

A função print() exibe na tela informações para o usuário, segue abaixo a tela de-execução:

```
C:\Users\LEONI\AppData\Local\Programs\Python\Python36\python.exe
Olá mundo!
Process returned 0 <0x0>      execution time : 0.156 s
Pressione qualquer tecla para continuar. . .
```

Fonte: Elaborado pelo autor (2023).

Também é possível exibir cálculos matemáticos com a função print(), conforme abaixo:

```
exemplo2.py — C:\Python\Exemplos — Atom
File Edit View Selection Find Packages Help
exemplo2.py
1 print(20+35)
```

Fonte: Elaborado pelo autor (2023).

Segue abaixo a tela de execução

```
C:\Users\LEONI\AppData\Local\Programs\Python\Python36\python.exe
55
Process returned 0 <0x0>      execution time : 0.135 s
Pressione qualquer tecla para continuar. . .
```

Fonte: Elaborado pelo autor (2023).

Também podemos concatenar um texto com um cálculo matemático, conforme abaixo:

```
exemplo3.py — C:\Python\Exemplos — Atom
File Edit View Selection Find Packages Help
exemplo3.py
1 print('O resultado da soma é:', 20+35)
```

Fonte: Elaborado pelo autor (2023).

Note que esteticamente é muito melhor fazermos isso, pois o usuário comprehende melhor as informações que estão sendo mostradas na tela. Segue abaixo a tela de execução:

```
C:\Users\LEONI\AppData\Local\Programs\Python\Python36\python.exe
0 resultado da soma é: 55
Process returned 0 <0x0>      execution time : 0.161 s
Pressione qualquer tecla para continuar. . .
```

Fonte: Elaborado pelo autor (2023).



ATIVIDADE DE FIXAÇÃO

1. Escreva sobre a linguagem Python.
2. O que significa dizer que uma linguagem é de baixo nível?
3. O que significa dizer que uma linguagem é de médio nível?

4. O que significa dizer que uma linguagem é de alto nível?
 5. O Python é uma linguagem de baixo, médio ou alto nível?
 6. O que é uma IDE?
 7. Qual a diferença entre o IDLE (PYTHON SHELL) e o interpretador do Python via Prompt de Comando?
8. Porque usaremos o Atom para ser o interpretador do Python?
 9. O que significa dizer que o que delimita o bloco no Python é a indentação?
 10. Desenvolva um programa que faça aparecer na tela uma frase de uma letra de música.
 11. Quais são as formas de se fazer comentário no código fonte do Python?
 12. Execute os dois comandos: `print('Olá\mundo!')` e `print('Olá mundo!')`. Explique a diferença entre eles.

Variáveis em Python



Fonte:<https://youtu.be/GwsN0jTZ-yE>

Uma variável serve para armazenar um valor na memória, para que esse valor possa ser acessado/utilizado pelo software quando for necessário. O nosso cérebro possui um comportamento semelhante, por exemplo, o cérebro tem a capacidade de armazenar vários tipos de informações, tais como: número de CPF, cheiro da comida da mãe, voz de pessoas conhecidas e muito mais. Cada informação é catalogada com um tipo.

As variáveis são armazenadas na memória. O processo de criar um vínculo, ou seja, associar uma área de memória, chamamos de liberação.

Existem quatro categorias de alocação: estática, dinâmica da pilha, dinâmica no heap explícita e dinâmica no heap implícita. As variáveis estáticas são aquelas em que a vinculação ocorre antes do início da execução.

Quando a vinculação é criada a partir de uma sentença de declaração, a chamada variável dinâmica de pilha. E advinha onde a variável está alocada! Sim, na pilha (stack memory). É alocada e liberada em tempo de execução.

Exemplo 1:

Por exemplo, ao escrever `cpf = 32485977805`, declaramos uma variável com o nome de `cpf` e atribuindo a ela o valor de `32485977805`.

```
exemplo_variavel1.py — C:\Python\Exemplos — Atom
File Edit View Selection Find Packages Help
    exemplo_variavel1.py •
1 cpf = 32485977805
```

Fonte: Elaborado pelo autor (2023).

Ao utilizar o comando `print`, poderemos visualizar na tela o valor armazenado na variável, conforme demonstrado na linha 3 abaixo:

```
exemplo_variavel1.py — C:\Python\Exemplos — Atom
File Edit View Selection Find Packages Help
    exemplo_variavel1.py •
1 cpf = 32485977805
2
3 print (cpf) ←
```

Fonte: Elaborado pelo autor (2023).

Segue a tela de execução abaixo:

```
C:\Users\Professor ESSA\AppData\Local\Programs\Python\Python37\python.exe
32485977805
Process returned 0 (0x0)      execution time : 0.063 s
Pressione qualquer tecla para continuar. . .
```

Fonte: Elaborado pelo autor (2023).

Podemos concatenar textos com variáveis para mostrar algo para o usuário, conforme a linha 3 do exemplo abaixo:

```
exemplo_variavel1.py — C:\Python\Exemplos — Atom
File Edit View Selection Find Packages Help
    exemplo_variavel1.py •
1 cpf = 32485977805
2
3 print ('O meu cpf é:',cpf) ←
```

Fonte: Elaborado pelo autor (2023).

Tela de execução:

C:\Users\Professor ESSA\AppData\Local\Programs\Python\Python37\python.exe
0 meu cpf é: 32485977805
Process returned 0 <0x0> execution time : 0.110 s
Pressione qualquer tecla para continuar. . .

Fonte: Elaborado pelo autor (2023).

Dicas para criação de variáveis

As variáveis não podem começar com números.

Devemos começar a variável com uma letra ou underscore (_)

O Python é case sensitive, ou seja, se você criou uma variável com letras minúsculas deverá sempre escrevê-la da mesma forma, do contrário ele não reconhecerá a variável. Portanto, você sempre deverá se atentar se está escrevendo o nome das variáveis exatamente como foram declaradas.

É uma boa prática evitar criar nomes de variáveis muito longos.

Procure também criar variáveis com nomes autoexplicativos, isso facilitará o entendimento do código por outras pessoas.

Palavras chaves do Python				
False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	
break	except	in	raise	

Fonte: Elaborado pelo autor (2023).

Além disso, também evite dar nomes de funções para as variáveis.

Exemplo1

No exemplo abaixo, foram declaradas três variáveis, chamadas: cpf, nome e altura.

```
exemplo_variavel1.py — C:\Python\Exemplos — Atom
File Edit View Selection Find Packages Help
exemplo_variavel1.py
1 cpf = 32485977805
2
3 nome = 'Fulano'
4
5 altura = 1.75
```

Fonte: Elaborado pelo autor (2023).

Automaticamente o Python atribui a elas um tipo, que dependerá do valor que estaremos atribuindo a elas, por exemplo, na variável cpf estamos atribuindo a ela um número inteiro (int), já em nome estamos atribuindo caracteres (string) e na variável altura estamos atribuindo um valor real (float).

Note abaixo nas linhas 5,6 e 7 que é possível utilizarmos o comando type para mostrar o tipo da variável, no caso também utilizamos o comando print para que essas informações sejam mostradas em tela.

```
exemplo_variavel1.py — C:\Python\Exemplos — Atom
File Edit View Selection Find Packages Help
exemplo_variavel1.py
1 cpf = 32485977805
2 nome = 'Fulano'
3 altura = 1.75
4
5 print (type(nome))
6 print (type(cpf))
7 print (type(altura))
```

Fonte: Elaborado pelo autor (2023).

Segue abaixo a tela de execução, note que a variável do tipo nome é do tipo string (str), cpf é do tipo inteiro (int) e altura é do tipo real (float):

```
C:\Users\Professor ESSA\AppData\Local\Programs\Python\Python37\python.exe
<class 'str'>
<class 'int'>
<class 'float'>
```

Fonte: Elaborado pelo autor (2023).

Podemos melhorar a forma de visualizar essas informações concatenando textos no comando print, conforme as linhas 5,6 e 7 da tela abaixo:

```
exemplo_variavel1.py — C:\Python\Exemplos — Atom
File Edit View Selection Find Packages Help
    exemplo_variavel1.py •
1 cpf = 32485977805
2 nome = 'Fulano'
3 altura = 1.75
4
5 print ('O tipo da variável nome é:',type(nome))
6 print ('O tipo da variável cpf é:',type(cpf))
7 print ('O tipo da variável altura é:',type(altura))
8
```

Fonte: Elaborado pelo autor (2023).

Segue a tela de execução:

```
C:\Users\Professor ESSA\AppData\Local\Programs\Python\Python37\python.exe
0 tipo da variável nome é: <class 'str'>
0 tipo da variável cpf é: <class 'int'>
0 tipo da variável altura é: <class 'float'>
```

```
exemplo_variavel2.py — C:\Python\Exemplos — Atom
File Edit View Selection Find Packages Help
    exemplo_variavel2.py •
1
2 num1 = 10
3 num2 = 20
4
5 print('A soma entre',num1,'e',num2,'é:',num1+num2)
6
```

Fonte: Elaborado pelo autor (2023).

```
C:\Users\Professor ESSA\AppData\Local\Programs\Python\Python37\python.exe
A soma entre 10 e 20 é: 30
```

Fonte: Elaborado pelo autor (2023).

Strings

As strings são entendidas pelo Python como uma sequência de caracteres.

Para criar variáveis do tipo string (caracteres) basta quando for atribuir um valor para a variável utilizar aspas simples ou aspas duplas, conforme a tela abaixo:

```
exemplo_string1.py — C:\Python\Exemplos — Atom
File Edit View Selection Find Packages Help
exemplo_string1.py
1
2 nome = 'Fulano' ←
3 sobrenome = "Bezerra" ←
```

Fonte: Elaborado pelo autor (2023).

Note que ao utilizar os comandos print e type conseguimos confirmar que as variáveis “nome” e “sobrenome” são do tipo string

```
5 print(type(nome))
6 print(type(sobrenome))
```

Fonte: Elaborado pelo autor (2023).

Segue a tela de execução:

```
C:\Users\Professor ESSA\AppData\Local\Programs\Python\Python37\python.exe
<class 'str'>
<class 'str'>
```

Fonte: Elaborado pelo autor (2023).

Note que com o operador ‘+’ é possível concatenar as variáveis que são do tipo string.

Segue abaixo um exemplo, na linha 5 o operador ‘+’ utilizado no comando print, fez com que as variáveis fossem concatenadas, porém sem espaço entre elas. Para que seja possível concatenar com um espaço entre as variáveis é necessário utilizar um espaço em branco na concatenação, assim como demonstrado na linha 6.

```
exemplo_string1.py — C:\Python\Exemplos — Atom
File Edit View Selection Find Packages Help
exemplo_string1.py
1
2 nome = 'Fulano'
3 sobrenome = "Bezerra"
4
5 print (nome+sobrenome) ←
6 print (nome+ ' '+sobrenome) ←
```

Fonte: Elaborado pelo autor (2023).

Segue abaixo a tela de execução do exemplo:

```
C:\Users\Professor ESSA\AppData\Local\Programs\Python\Python37\python.exe
FulanoBezerra
Fulano Bezerra
```

Fonte: Elaborado pelo autor (2023).

Podemos também utilizar o operador '*' nas variáveis do tipo string

```
exemplo_string2.py — C:\Python\Exemplos — Atom
File Edit View Selection Find Packages Help
exemplo_string2.py •
1
2 nome = 'Fulano'
3 print (nome*3) ←
4
```

Fonte: Elaborado pelo autor (2023).

Segue abaixo a tela de execução, note que o nome foi exibido 3 vezes por conta que utilizamos o operador '*', que no caso o Python entendeu no exemplo acima (na linha 3) que era para escrever o nome três vezes.

```
C:\Users\Professor ESSA\AppData\Local\Programs\Python\Python37\python.exe
FulanoFulanoFulano
```

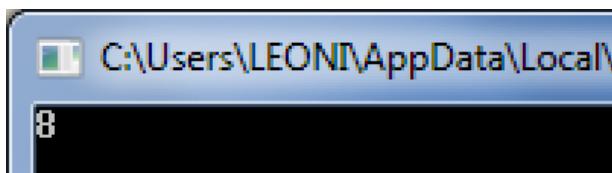
Fonte: Elaborado pelo autor (2023).

Como mencionado anteriormente as variáveis do tipo string são um conjunto de caracteres, portanto ao utilizar a função `len(nome_da_variável)`, poderemos verificar a quantidade de caracteres armazenados numa variável do tipo string.

```
exemplo2Strings.py — C:\Python\Exemplos — Atom
File Edit View Selection Find Packages Help
exemplo2Strings.py •
1 nome = "Beltrano" ←
2 print(len(nome))
3
4 ↑
```

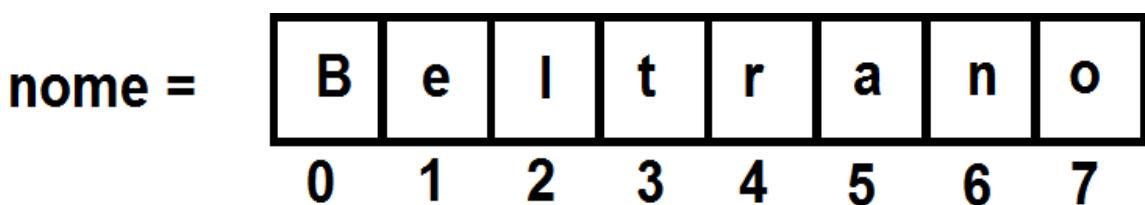
Fonte: Elaborado pelo autor (2023).

Segue abaixo a tela de execução:



Fonte: Elaborado pelo autor (2023).

Note que o nome “Beltrano” ocupa 8 posições da variável nome. Conforme abaixo, observe que o índice (número da posição) começa em zero:



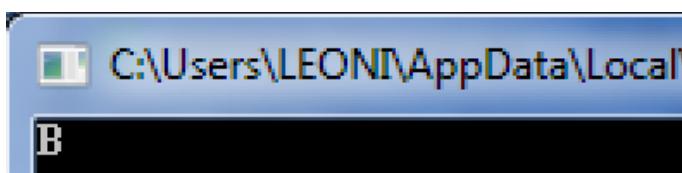
Fonte: Elaborado pelo autor (2023).

Por exemplo, se quisermos exibir na tela o caractere que está no índice zero, basta fazer conforme o exemplo abaixo na linha 2.

```
exemplo2Strings.py — C:\Python\Exemplos — Atom
File Edit View Selection Find Packages Help
exemplo2Strings.py
1 nome = "Beltrano"
2 print (nome[0])
3
4
```

Fonte: Elaborado pelo autor (2023).

Segue abaixo a tela de execução, note que apareceu o primeiro caractere do nome armazenado na variável nome:



Fonte: Elaborado pelo autor (2023).

No exemplo abaixo, foi exibido todos os 8 caracteres armazenados na variável nome através dos índices de cada posição. Ver linhas 2 até 9.

```
exemplo2Strings.py — C:\Python\Exemplos — Atom
File Edit View Selection Find Packages Help
exemplo2Strings.py •

1 nome = "Beltrano"
2 print (nome[0])
3 print (nome[1])
4 print (nome[2])
5 print (nome[3])
6 print (nome[4])
7 print (nome[5])
8 print (nome[6])
9 print (nome[7])
```

Fonte: Elaborado pelo autor (2023).

Segue abaixo o resultado de execução do código:

```
C:\Users\LEONI\AppData\Local\Temp\Temporary Internet Files\Content.IE5\HJLWZPQ\exemplo2Strings.py
B
e
l
t
r
a
n
o
```

Fonte: Elaborado pelo autor (2023).

Para acessar o último caractere de uma variável do tipo string, não é necessário saber o seu tamanho, basta escrever: nome_variável [-1], conforme o exemplo abaixo na linha 2:

```
exemplo2Strings.py — C:\Python\Exemplos
File Edit View Selection Find Packages
exemplo2Strings.py •

1 nome = "Beltrano"
2 print (nome[-1])
3
4
```

Fonte: Elaborado pelo autor (2023).

Segue abaixo a tela de execução, note que foi exibida a letra “o” pois é a última letra dos caracteres armazenados na variável nome, que está armazenando o nome “Beltrano”.

```
C:\Users\LEONI\AppData\Local\\
```

```
o
```

Fonte: Elaborado pelo autor (2023).

Fatiando a variável

No exemplo abaixo (linha 2), note que é possível mostrar uma sequência determinada de caracteres de uma variável, no caso serão exibidos os caracteres do índice 0 até o índice 2:

```
exemplo2Strings.py — C:\Python\Exemplos
```

```
File Edit View Selection Find Packages
```

```
exemplo2Strings.py
```

```
1 nome = "Beltrano"
2 print (nome[0:3])
```

Fonte: Elaborado pelo autor (2023).

Segue abaixo a tela de execução, note que foi exibido os três primeiros caracteres armazenados na variável nome:

```
C:\Users\LEONI\AppData\Local\\
```

```
Be1
```

Fonte: Elaborado pelo autor (2023).

Abaixo, veja que é possível especificarmos para mostrar os últimos caracteres, no exemplo, ao utilizar print (nome_variável [-2:]) o Python irá mostrar os dois últimos caracteres armazenados na variável.

```
exemplo2Strings.py — C:\Python\Exemplos  
File Edit View Selection Find Packages  
exemplo2Strings.py  
1 nome = "Beltrano"  
2 print (nome[-2:])
```

Fonte: Elaborado pelo autor (2023).

Segue abaixo a tela de execução:

```
C:\Users\LEONI\AppData\Local\P  
no
```

Fonte: Elaborado pelo autor (2023).

Para ler um texto a partir do usuário

Para lermos um texto de um usuário para armazenar numa variável utilizamos a função `input()`, conforme a tela abaixo (linha 1):

```
exemplo_strings.py  
1 nome = input()  
2 print (nome)
```

Fonte: Elaborado pelo autor (2023).

Segue a tela de execução do código:

```
C:\Users\joaofl\AppData\Local\P  
Fulano  
Fulano
```

Fonte: Elaborado pelo autor (2023).

Podemos melhorar a tela adicionando na função `input` uma frase explicando ao usuário o que ele deve digitar, conforme abaixo (linha 1):

```
exemplo_strings.py
1 nome = input('Escreva o seu nome:')
2 print ('Olá '+nome+'!')
```

Fonte: Elaborado pelo autor (2023).

Segue abaixo a tela de execução:

```
C:\Users\joao\AppData\Local\Programs\Python\Python310\python.exe exemplo_strings.py
Escreva o seu nome:Adamastor
Olá Adamastor!
```

Fonte: Elaborado pelo autor (2023).

OBS: A função `input()` lerá as informações do usuário e irá armazená-las como strings.

Números

Como visto anteriormente, ao atribuirmos um número para uma variável, o Python automaticamente irá definir a variável como do tipo inteiro (int) ou real (float). Note que nas linhas 3 e 4 foi atribuído um número inteiro para a variável num1 e atribuído um número real (float) para a variável num2.

```
exemplo1_numeros.py
1
2
3 num1 = 2
4 num2 = 5.5
5
6 print(type(num1))
7 print(type(num2))
```

Fonte: Elaborado pelo autor (2023).

Ao utilizar o comando `print (type(num1))` e `print (type(num2))` é possível checar o tipo da variável, conforme a tela de execução abaixo:

C:\Users\joaof\AppData\Local\Programs\Python\Python37\python.exe
<class 'int'>
<class 'float'>

Fonte: Elaborado pelo autor (2023).

Note abaixo que foi atribuído para a variável num3 a soma de num1+num2 cujo resultado será 7.5. Portanto, o Python entenderá que a variável num3 deverá ser do tipo real (float) pois ela possui casas decimais após a vírgula ou ponto.

Na linha 7 será apresentado o valor da variável num3, e na linha 8 será apresentado qual é o tipo dela, ou seja, se é do tipo inteiro ou real.

 exemplo1_numeros.py
1
2
3 num1 = 2
4 num2 = 5.5
5 num3 = num1 + num2
6
7 print (num3)
8 print (type(num3))
9

Fonte: Elaborado pelo autor (2023).

Segue abaixo a tela de execução:

 C:\Users\joaof\AppData\Loc
7.5
<class 'float'>

Fonte: Elaborado pelo autor (2023).

Exemplo de potenciação

Para fazer a potenciação de um número, basta utilizar o sinal **, no exemplo abaixo (linha 6) a variável num1 cujo valor é 2 terá uma potência de 4.

```
exemplo1_numeros.py
1
2
3 num1 = 2
4 num2 = 5.5
5
6 potenciacao = num1 ** 4
7 print (potenciacao)
```

Fonte: Elaborado pelo autor (2023).

Segue abaixo a tela de execução com o resultado da potenciação:

```
C:\Users\joaoft\Appl
16
```

Fonte: Elaborado pelo autor (2023).

Priorização de cálculos matemáticos

Para forçar a priorização de uma operação matemática em relação a outra, devemos utilizar os parênteses, conforme a linha 3, note que forçamos para que primeiro seja feita a soma dos números para que depois seja feita a divisão.

```
exemplo1_numeros.py
1 print(1+1/2)
2
3 print ((1+1)/2)
4
5
```

Fonte: Elaborado pelo autor (2023).

Segue abaixo a tela de execução:

```
C:\Users\joao\AppData
1.5
1.0
```

Fonte: Elaborado pelo autor (2023).

Lendo números a partir do usuário

Números inteiros

Para lermos um número através do usuário para armazenar numa variável utilizamos a função `int()` e depois a função `input()`, conforme a tela abaixo (linha 1 e 2):

```
exemplo2_numeros.py
1 num1 = int(input('Digite um valor para num1: '))
2 num2 = int(input('Digite um valor para num2: '))
3
4 soma = num1+num2
5
6 print ('A soma de num1 + num2 =',soma)
```

Fonte: Elaborado pelo autor (2023).

Conforme dito anteriormente a função `input()` permite somente a leitura de texto, se quisermos ler números inteiros teremos que utilizar a função `int()` para converter em número o texto recebido pela função `input()`.

Abaixo segue a tela de execução:

```
C:\Users\joao\AppData\Local\Programs\Pyt
Digite um valor para num1: 25
Digite um valor para num2: 17
A soma de num1 + num2 = 42
```

Fonte: Elaborado pelo autor (2023).

Números reais (float)

Para lermos um número através do usuário para armazenar numa variável utilizamos a função float() e depois a função input(), conforme a tela abaixo (linha 1 e 2):

```
exemplo2_numeros.py
1 num1 = float(input('Digite um valor para num1: '))
2 num2 = float(input('Digite um valor para num2: '))
3
4 soma = num1+num2
5
6 print ('A soma de num1 + num2 =',soma)
```

Fonte: Elaborado pelo autor (2023).

Segue abaixo a tela de execução:

```
C:\Users\joaof\AppData\Local\Programs\Python
Digite um valor para num1: 15.5
Digite um valor para num2: 37.2
A soma de num1 + num2 = 52.7
```

Fonte: Elaborado pelo autor (2023).

Variáveis do tipo lógico

Para definir uma variável como do tipo lógico (booleano) é necessário atribuir para ela True (verdadeiro) ou False (Falso), conforme o exemplo abaixo (linha 1).

Note que na linha 2, foi dado o comando print(type(nome_da_variavel)) para mostrar o tipo da variável.

```
exemplo_booleano.py — C:\Users\joaof\AppData\Local\Programs\Python\Python37\idle.pyw(1)
File Edit View Selection Find Packages Help
exemplo_booleano.py
1 booleano = True
2 print (type(booleano))
3
```

Fonte: Elaborado pelo autor (2023).

Segue abaixo a tela de execução:

```
C:\Users\joaoof\AppData\Local\Programs\Python\Python310\python.exe -c "print(type(bool))"
```

```
<class 'bool'>
```

Fonte: Elaborado pelo autor (2023).



ATIVIDADE DE FIXAÇÃO

1. O que são variáveis?
2. Qual a função no Python que é responsável por informar o tipo da variável?
3. O que deve ser feito para ler uma variável do tipo inteiro? Dê um exemplo.
4. O que deve ser feito para ler uma variável do tipo real (float)? Dê um exemplo.
5. Faça um programa que calcule o valor de desconto para um determinado produto, faça um Print Screen do código fonte do seu programa e também do programa em execução.
6. Crie um programa para converter uma temperatura da escala Celsius para a escala Fahrenheit. A fórmula para conversão é $Fahrenheit = ((Celsius * 9) / 5) + 32$
7. Elabore um programa que calcule quantas notas de 50,10 e 1 são necessárias para se pagar uma conta cujo valor é fornecido.
8. Elabore um programa que permita a entrada de um número inteiro e diga se ele é par ou ímpar.
9. Elabore um programa que permita a entrada de dois valores, x e y , troque seus valores entre si e então exiba os novos resultados.



Fonte: Elaborado pelo autor (2023).

TEMA 10

Listas, tuplas e dicionários

Habilidades:

- Codificar programas utilizando técnica de programação estruturada.
- Depurar e versionar programas, utilizando ambiente de desenvolvimento integrado.



Fonte: <https://youtu.be/bHn91RxiTjY?list=PLyqOvdQmGdTSEPnO0DKgHIkXb8x3cyglD>

As listas e tuplas são variáveis que podem armazenar um conjunto de informações, que podem ser do tipo texto, números inteiros, reais ou lógicos. Sem eles teríamos que criar inúmeras variáveis para armazenar os dados, por exemplo, podemos criar uma lista para armazenar os nomes dos alunos, se tivéssemos que criar uma variável para cada aluno seria uma tarefa extremamente trabalhosa e desnecessária.

Tuplas

Para definir uma tupla, basta atribuir um conjunto de informações para uma variável, porém essas informações devem estar entre parênteses.

```
exemplo_listas_tuplas.py
1 estacoes = ('primavera', 'verao', 'outono', 'inverno')
2
3 print(estacoes)
4 print(estacoes[3])
5
```

Fonte: Elaborado pelo autor (2023).

Segue abaixo a tela de execução:

```
C:\Users\joao\AppData\Local\Programs\Python\Python37\python.exe
('primavera', 'verao', 'outono', 'inverno')
inverno
```

Fonte: Elaborado pelo autor (2023).

Abaixo utilizamos o type() para mostrar o tipo da variável estacoes (ver linha 3):

```
exemplo_listas_tuplas.py
1 estacoes = ('primavera', 'verao', 'outono', 'inverno')
2
3 print(type(estacoes))
4
```

Fonte: Elaborado pelo autor (2023).

Abaixo, segue a tela de execução, note que trata-se de uma “Tupla”:

```
C:\Users\joao\AppData\Local\Programs\Python\Python37
<class 'tuple'>
```

Fonte: Elaborado pelo autor (2023).

Listas

Assim como a Tupla uma Lista também pode armazenar um conjunto de informações, a diferença principal entre uma Lista e um Tupla é que numa Tupla sabemos que as informações são imutáveis (não mudam), já numa Lista as informações são mutáveis (podem mudar).

As Listas também possuem índices onde ficam armazenados os elementos (informações). Exemplo:

```
exemplo_listas_tuplas.py
1 alunos = ['Rafaela', 'Pedro', 'Bianca', 'Jessica']
2
3 print(type(alunos))
```

Fonte: Elaborado pelo autor (2023).

Segue abaixo a tela de execução:

```
C:\Users\joao\AppData\Local\Progr
<class 'list'>
```

Fonte: Elaborado pelo autor (2023).

Abaixo utilizamos a função len() para contar o tamanho dos elementos das variáveis estacoes e alunos.

```
exemplo_listas_tuplas.py
1 estacoes = ('primavera', 'verao', 'outono', 'inverno')
2 alunos = ['Rafaela', 'Pedro', 'Bianca', 'Jessica', 'Bruno']
3
4 print(len(estacoes))
5 print(len(alunos))
```

Fonte: Elaborado pelo autor (2023).

Segue abaixo a tela de execução:

```
C:\Users\joao\AppData\Local
4
5
```

Fonte: Elaborado pelo autor (2023).

Note que é possível visualizarmos os elementos armazenados em cada índice, veja as linhas 4 até 8:

```
exemplo_listas_tuplas.py
1 estacoes = ('primavera', 'verao', 'outono', 'inverno')
2 alunos = ['Rafaela', 'Pedro', 'Bianca', 'Jessica', 'Bruno']
3
4 print (estacoes[0])
5 print (estacoes[1])
6
7 print (alunos[2])
8 print (alunos[3])
```

Fonte: Elaborado pelo autor (2023).

Segue abaixo a tela de execução:

```
C:\Users\joao\AppData\Local\Programs\Python\Python37\python.exe
primavera
verao
Bianca
Jessica
```

Fonte: Elaborado pelo autor (2023).

Alterando um elemento de uma lista

```
exemplo_listas_tuplas.py
1 estacoes = ('primavera','verao','outono','inverno')
2 alunos = ['Rafaela','Pedro','Bianca','Jessica','Bruno']
3
4 alunos[0]= 'Thiago'
5
6 print (alunos)
```

Fonte: Elaborado pelo autor (2023).

Segue abaixo a tela de execução:

```
C:\Users\joao\AppData\Local\Programs\Python\Python37\python.exe
['Thiago', 'Pedro', 'Bianca', 'Jessica', 'Bruno']
```

Fonte: Elaborado pelo autor (2023).

Colocando um novo elemento na lista

```
exemplo_listas_tuplas.py
1 estacoes = ('primavera','verao','outono','inverno')
2 alunos = ['Thiago','Pedro','Bianca','Jessica','Bruno']
3
4 alunos.append('Helena') -----^
5
6 print(alunos)
```

Fonte: Elaborado pelo autor (2023).

Segue abaixo a tela de execução:

```
C:\Users\joaof\AppData\Local\Programs\Python\Python37\python.exe
['Thiago', 'Pedro', 'Bianca', 'Jessica', 'Bruno', 'Helena']
```

Fonte: Elaborado pelo autor (2023).

Inserir um elemento a partir de um determinado índice

```
exemplo_listas_tuplas.py
1 estacoes = ('primavera', 'verao', 'outono', 'inverno')
2 alunos = ['Thiago', 'Pedro', 'Bianca', 'Jessica', 'Bruno', 'Helena']
3
4 alunos.insert(1, 'Daniela')
5
6 print(alunos)
```

Fonte: Elaborado pelo autor (2023).

Segue abaixo a tela de execução:

```
C:\Users\joaof\AppData\Local\Programs\Python\Python37\python.exe
['Thiago', 'Daniela', 'Pedro', 'Bianca', 'Jessica', 'Bruno', 'Helena']
```

Fonte: Elaborado pelo autor (2023). Fonte: Elaborado pelo autor (2023).

Ordenação de uma lista

```
exemplo_listas_tuplas.py
1 alunos = ['Thiago', 'Pedro', 'Bianca', 'Jessica', 'Bruno', 'Helena']
2 alunos.sort()
3 print(alunos)
```

Fonte: Elaborado pelo autor (2023).

Segue abaixo a tela de execução:

```
C:\Users\joaof\AppData\Local\Programs\Python\Python37\python.exe
['Bianca', 'Bruno', 'Helena', 'Jessica', 'Pedro', 'Thiago']
```

Fonte: Elaborado pelo autor (2023).

Para remover um dado de uma lista

```
exemplo_listas_tuplas.py
```

```
1 alunos = ['Thiago', 'Pedro', 'Bianca', 'Jessica', 'Bruno', 'Helena']
2 alunos.pop(0)
3 print(alunos)
```

```
C:\Users\joaof\AppData\Local\Programs\Python\Python37\python.exe
['Pedro', 'Bianca', 'Jessica', 'Bruno', 'Helena']
```

Fonte: *Imagens elaboradas pelo autor (2023)*.

Outra maneira de excluir um dado

```
exemplo_listas_tuplas.py
```

```
1 alunos = ['Thiago', 'Pedro', 'Bianca', 'Jessica', 'Bruno', 'Helena']
2 alunos.remove('Thiago')
3 print(alunos)
```

```
C:\Users\joaof\AppData\Local\Programs\Python\Python37\python.exe
['Pedro', 'Bianca', 'Jessica', 'Bruno', 'Helena']
```

Fonte: *Elaborado pelo autor (2023)*.

Armazenando números inteiros e reais

```
exemplo_listas_tuplas.py
```

```
1 notasAlunos = [5.5, 6.8, 7, 10, 5.7]
2 print(notasAlunos)
3
4 idadeAlunos = [17, 18, 19, 19, 20]
5 print(idadeAlunos)
```

Fonte: *Elaborado pelo autor (2023)*.

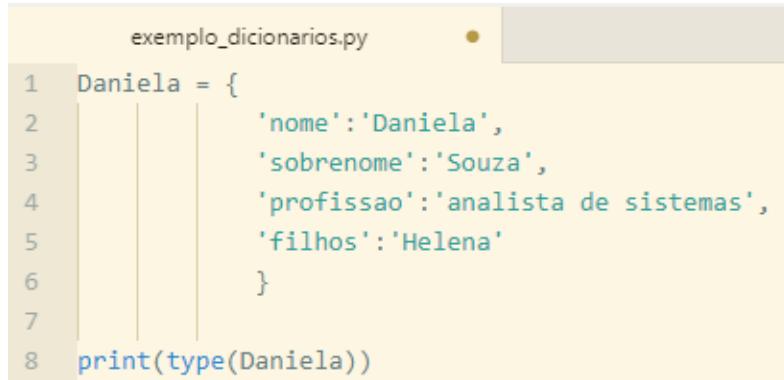
Segue abaixo a tela de execução:

```
C:\Users\joaof\AppData\Local\
[5.5, 6.8, 7, 10, 5.7]
[17, 18, 19, 19, 20]
```

Fonte: *Elaborado pelo autor (2023)*.

Dicionários

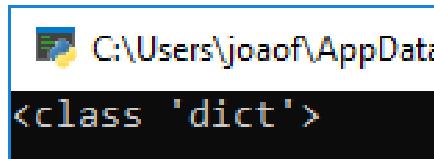
Os dicionários assim como as Tuplas e Listas também podem armazenar um conjunto de valores, a diferença é que esses valores não possuem uma sequência ordenada, ou seja, o acesso aos valores não ocorre por meio de índices e sim por meio de keys (chaves).



```
exemplo_dicionarios.py
1 Daniela = {
2     'nome': 'Daniela',
3     'sobrenome': 'Souza',
4     'profissao': 'analista de sistemas',
5     'filhos': 'Helena'
6 }
7
8 print(type(Daniela))
```

Fonte: Elaborado pelo autor (2023).

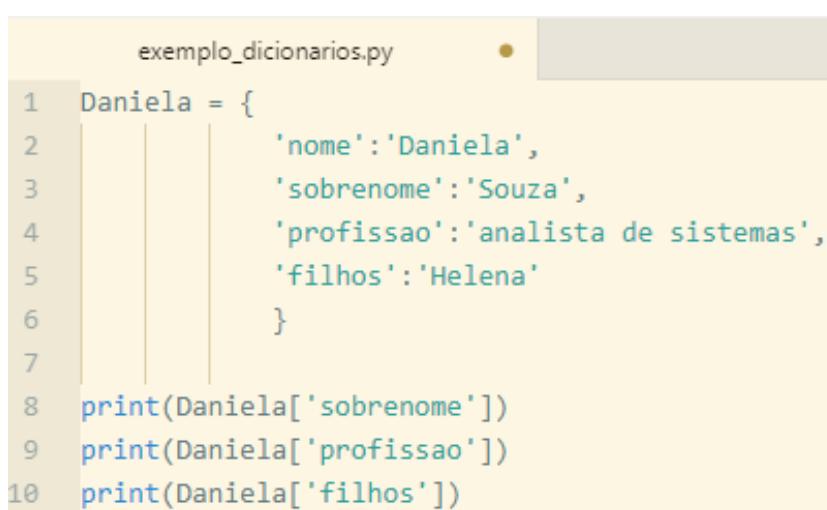
Na imagem acima (linha 8), foi novamente utilizada a função type() para checarmos se a variável é reconhecida como do tipo dicionário, note abaixo que sim, é do tipo dicionário:



```
C:\Users\joao\AppData
<class 'dict'>
```

Fonte: Elaborado pelo autor (2023).

Na tela abaixo, se quisermos visualizar as informações armazenadas teremos que utilizá-las em vez de índices para acessar os dados.



```
exemplo_dicionarios.py
1 Daniela = {
2     'nome': 'Daniela',
3     'sobrenome': 'Souza',
4     'profissao': 'analista de sistemas',
5     'filhos': 'Helena'
6 }
7
8 print(Daniela['sobrenome'])
9 print(Daniela['profissao'])
10 print(Daniela['filhos'])
```

Fonte: Elaborado pelo autor (2023).

Segue abaixo a tela de execução:

```
C:\Users\joao\AppData\Local\Temp\nbtemp1234567890\nbtemp1234567890.py
Souza
analista de sistemas
Helena
```

Fonte: Elaborado pelo autor (2023).

Verificando a quantidade de chaves armazenadas no dicionário

Para checarmos a quantidade de chaves de um dicionário podemos utilizar a função len(), conforme abaixo:

```
exemplo_dicionarios.py
Daniela = {
    'nome': 'Daniela',
    'sobrenome': 'Souza',
    'profissao': 'analista de sistemas',
    'filhos': 'Helena'
}
print(len(Daniela))
```

Fonte: Elaborado pelo autor (2023).

Abaixo segue a tela de execução:

```
C:\Users\joao\AppData\Local\Temp\nbtemp1234567890\nbtemp1234567890.py
4
```

Fonte: Elaborado pelo autor (2023).

Deletando uma chave: Para deletar uma chave devemos utilizar o comando del Nome_dicionario [nome_chave]:

```
exemplo_dicionarios.py
1 Daniela = {
2     'nome': 'Daniela',
3     'sobrenome': 'Souza',
4     'profissao': 'analista de sistemas',
5     'filhos': 'Helena'
6 }
7
8
9 del Daniela ['filhos']
10 print(Daniela)
```

Fonte: Elaborado pelo autor (2023).

Segue abaixo a tela de execução:

```
C:\Users\joaof\AppData\Local\Programs\Python\Python37\python.exe
{'nome': 'Daniela', 'sobrenome': 'Souza', 'profissao': 'analista de sistemas'}
```

Fonte: Elaborado pelo autor (2023).

Alterando os dados de uma chave

Para alterar um dado de uma chave, basta referenciar qual chave deseja-se alterar e atribuir a ela o valor desejado, conforme a linha 9 da tela abaixo:

```
exemplo_dicionarios.py
1 Daniela = {
2     'nome': 'Daniela',
3     'sobrenome': 'Souza',
4     'profissao': 'analista de sistemas',
5     'filhos': 'Helena'
6 }
7
8
9 Daniela ['profissao'] = 'analista de testes'
10 print (Daniela['profissao'])
```

Fonte: Elaborado pelo autor (2023).

```
C:\Users\joaof\AppData\Local\Programs\Python\Python37\python.exe
analista de testes
```

Fonte: Elaborado pelo autor (2023).

Verificando se existe uma determinada chave no dicionário

Basta fazer 'nome_chave' in nome_dicionario, conforme tela abaixo (linha 9)

```
exemplo_dicionarios.py
1 Daniela = {
2     'nome':'Daniela',
3     'sobrenome':'Souza',
4     'profissao':'analista de sistemas',
5     'filhos':'Helena'
6 }
7
8
9 print('sobrenome' in Daniela)
```

Fonte: Elaborado pelo autor (2023).

Segue a tela de execução abaixo:

```
C:\Users\joaof\AppData\l
True
```

Fonte: Elaborado pelo autor (2023).

Lendo os valores armazenados num dicionário

Para ler um valor armazenado num dicionário, basta utilizar um laço de repetição for, conforme abaixo (linhas 9,10 e 11).

```
exemplo_dicionarios.py
1 Daniela = {
2     'nome':'Daniela',
3     'sobrenome':'Souza',
4     'profissao':'analista de sistemas',
5     'filhos':'Helena'
6 }
7
8
9 for x in Daniela:
10     print (x)
11
```

Fonte: Elaborado pelo autor (2023).

Segue abaixo a tela de execução:

```
C:\Users\joaof\AppData\Local\F
nome
sobrenome
profissao
filhos
```

Fonte: Elaborado pelo autor (2023).

Para melhorar a apresentação dos dados em tela, podemos utilizar como exemplo o laço de repetição abaixo, linhas 9 e 10;

```
exemplo_dicionarios.py
1 Daniela = {
2     'nome': 'Daniela',
3     'sobrenome': 'Souza',
4     'profissao': 'analista de sistemas',
5     'filhos': 'Helena'
6 }
7
8
9 for x in Daniela:
10     print(x+': '+Daniela[x])
11 
```

Fonte: Elaborado pelo autor (2023).

Segue abaixo a tela de execução do programa:

```
C:\Users\joaof\AppData\Local\Programs\
nome: Daniela
sobrenome: Souza
profissao: analista de sistemas
filhos: Helena
```

Fonte: Elaborado pelo autor (2023).

Conforme abaixo (linhas 8 e 9) podemos utilizar a função get() que pode trazer uma informação para o usuário, ela possui uma condição que se não aplicada pode, por exemplo, mostrar uma mensagem de erro para o usuário.

```
exemplo_dicionarios.py
```

```
1 Daniela = {  
2     'nome': 'Daniela',  
3     'sobrenome': 'Souza',  
4     'profissao': 'analista de sistemas',  
5     'filhos': 'Helena'  
6 }  
7  
8 print(Daniela.get('idade', 'Esta informação não consta no cadastro'))  
9 print(Daniela.get('profissao', 'Esta informação não consta no cadastro'))  
10
```

Fonte: Elaborado pelo autor (2023).

Segue abaixo a tela de execução:

```
C:\Users\joaof\AppData\Local\Programs\Python\Python37\  
Esta informação não consta no cadastro  
analista de sistemas
```

Fonte: Elaborado pelo autor (2023).

Limpando todas as chaves do dicionário

Ao utilizar a função `clear()`, serão removidas todas as chaves e consequentemente serão deletados todos os dados do dicionário, veja as linhas 8 e 9:

```
exemplo_dicionarios.py
```

```
1 Daniela = {  
2     'nome': 'Daniela',  
3     'sobrenome': 'Souza',  
4     'profissao': 'analista de sistemas',  
5     'filhos': 'Helena'  
6 }  
7  
8 Daniela.clear()  
9 print(Daniela)
```

Fonte: Elaborado pelo autor (2023).

```
C:\Users\joaof\AppData\Local\  
{}
```

Fonte: Elaborado pelo autor (2023).



ATIVIDADE DE FIXAÇÃO

- 1.** Qual a diferença entre uma tupla e uma lista?
- 2.** Qual a diferença entre um vetor comparado com as duplas e listas?
- 3.** Desenvolva um programa que calcule e exiba a diferença entre o maior e o menor elemento de uma lista denominada VALORES. Os valores da lista devem ser lidos.
- 4.** Desenvolva um programa que faça a soma de todos os elementos de índice par de uma lista.
- 5.** Desenvolva um programa que possua duas listas A e B com 5 elementos cada e então troque seus elementos, de forma que a lista A ficará com os elementos da lista B e vice-versa.
- 6.** Desenvolva um programa que utilizará a Cifra de César para criptografar uma mensagem digitada pelo usuário, após mostrar em tela a mensagem criptografada.
- 7.** Desenvolva um programa que armazenará números reais numa lista. A seguir, encontre o menor elemento da lista e a sua posição (índice) dentro da lista, mostre esses valores em tela.
- 8.** Crie uma tupla com os signos do zodíaco e exiba os valores dessa tupla em tela.
- 9.** O que é um dicionário? Qual a principal diferença de um dicionário se comparado com as tuplas e listas?
- 10.** Crie um dicionário para um determinado animal de estimação. Exiba em tela as informações armazenadas nesse dicionário.

TEMA 11

Condicionais

Habilidades:

- Codificar programas utilizando técnica de programação estruturada.
- Depurar e versionar programas, utilizando ambiente de desenvolvimento integrado.



Condicional simples

Condicionais são condições impostas para que algo aconteça, ou seja, espera-se que uma condição seja verdadeira ou falsa, e para cada uma das situações o programa fará alguma coisa especificada pelo programador. As condicionais são partes fundamentais nos programas, pois devemos tentar prever as situações que podem ocorrer, principalmente quando recebemos algum dado através do usuário.

No exemplo abaixo, é utilizado a condição if, note que a condição if possui uma condição que se for verdadeira executará alguma ação:

```
exemplo_condicionais.py
1 idade = 20
2
3 if idade == 20:
4     print ('Maior de idade')
```

Fonte: Elaborado pelo autor (2023).

Note abaixo que a mensagem “Maior de idade” apareceu porque a condição idade == 20 foi correspondida:

C:\Users\joao\AppData\
Maior de idade

Fonte: Elaborado pelo autor (2023).

Abaixo, segue um exemplo recebendo a idade através do usuário com a função input()

 exemplo_condicionais.py
1 idade = int(input('Digite a sua idade: '))
2
3 if idade <= 18:
4 print ('Você é menor de idade')

Fonte: Elaborado pelo autor (2023).

Segue abaixo a tela de execução:

 C:\Users\joao\AppData\Local\Programs\
Digite a sua idade: 15
Você é menor de idade

Fonte: Elaborado pelo autor (2023).

Condicional composta

Utilizamos o comando else para prever uma ação para quando a condição inicial for falsa, no exemplo abaixo o if verifica se a idade é menor que 18, se for escreverá em tela “Você é menor de idade”, caso contrário (else) escreverá: “Você é maior de idade”.

 Selecionar C:\Users\joao\AppData\Local\
Digite a sua idade: 35
Você é maior de idade

Fonte: Elaborado pelo autor (2023).

Segue abaixo a tela de execução:

```
exemplo_condicionais.py
1 idade = int(input('Digite a sua idade: '))
2
3 if idade < 18:
4     print ('Você é menor de idade')
5 else:
6     print ('Você é maior de idade')
```

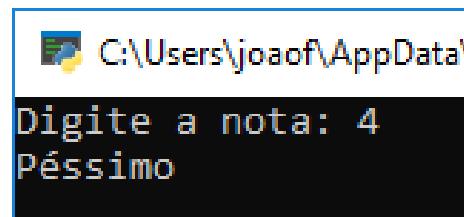
Fonte: Elaborado pelo autor (2023).

No exemplo abaixo foi utilizado a condição elif que permite que verifiquemos mais uma possibilidade:

```
exemplo_condicionais.py
1 nota = int(input('Digite a nota: '))
2
3 if nota == 10:
4     print('Excelente!')
5 elif nota <= 5:
6     print('Péssimo')
7 else:
8     print('Bom')
```

Fonte: Elaborado pelo autor (2023).

Segue abaixo a tela de execução:



Fonte: Elaborado pelo autor (2023).

Veja que combinando vários if é possível tentar prever várias possibilidades e tomar uma ação para cada uma delas. No exemplo a seguir, também está sendo utilizado o conectivo lógico “and”, este conectivo lógico exige que todas as condições sejam verdadeiras para que a ação ocorra.

```
exemplo_condicionais.py
```

```
1 idade = int(input('Digite a sua idade: '))
2 sexo = input('Digite o sexo M ou F: ').lower()
3
4 if idade < 18 and sexo == 'm':
5     print ('homem menor de idade')
6 elif idade >= 18 and sexo == 'm':
7     print ('homem maior de idade')
8 elif idade < 18 and sexo == 'f':
9     print ('mulher menor de idade')
10 elif idade >= 18 and sexo == 'f':
11     print ('mulher maior de idade')
12 else:
13     print('Erro na entrada de dados')
14
15
```

Fonte: Elaborado pelo autor (2023).

Segue abaixo a tela de execução:

```
C:\Users\joao\AppData\Local\Pro
Digite a sua idade: 35
Digite o sexo M ou F: M
homem maior de idade
```

Fonte: Elaborado pelo autor (2023).

No exemplo abaixo estamos utilizando o conectivo lógico “or”, este conectivo lógico exige que ao menos uma condição seja verdadeira para que seja tomada a ação programada:

```
exemplo_condicionais.py
```

```
1 nota = float(input('Digite a nota do aluno: '))
2 faltas = int(input('Digite a quantidade de faltas do aluno: '))
3
4 if nota < 5 or faltas > 4:
5     print('Aluno reprovado!')
6 else:
7     print('Aluno aprovado!')
```

Fonte: Elaborado pelo autor (2023).

```
C:\Users\joao\AppData\Local\Programs\Python\Python37\python.exe
Digite a nota do aluno: 10
Digite a quantidade de faltas do aluno: 6
Aluno reprovado!
```

Fonte: Elaborado pelo autor (2023).



ATIVIDADE DE FIXAÇÃO

1. Qual a diferença entre uma condicional simples e uma condicional composta?
2. Desenvolva um programa utilizando uma condicional simples, o algoritmo deve verificar se o valor digitado é menor ou igual a 35. Se a condição for verdadeira, deve aparecer uma mensagem: "O valor digitado é menor ou igual a 35!".
3. Desenvolva um programa de sua preferência que tenha uma condicional composta.
4. Desenvolva um programa que calcule e exiba a soma dos números pares contidos entre 1 e um número fornecido pelo usuário.
5. A contribuição para o INSS é calculada a partir da tabela a seguir:

TABELA VIGENTE - 2018	
Salário de contribuição (R\$)	Aliquota para fins de recolhimento ao INSS (%)
Até R\$ 1.693,72	8%
De R\$ 1.693,72 até R\$ 2.822,90	9%
De R\$ 2.822,91 até R\$ 5.645,80	11%

Fonte: Elaborado pelo autor (2023).

6. Desenvolva um programa para ler o valor de um salário bruto para que seja calculado o valor da contribuição ao INSS e o salário líquido restante.
7. Desenvolva um programa que leia um número inteiro e verifique se este é ou não um número primo (número primo é divisível por um e por ele mesmo).
8. Um número palíndromo é aquele que se lido da esquerda para a direita e da direita para a esquerda possui o mesmo valor (por exemplo: 34543). Desenvolva um programa que leia um número inteiro e verifique se ele é um palíndromo.

TEMA 12

Laços de repetição

Habilidades:

- Codificar programas, utilizando técnica de programação estruturada.



Fonte: <https://youtu.be/iFYWrDMfVNo>

Os laços de repetição são muito utilizados nos sistemas para fazer algum tipo de consistência de dados do usuário, ocorre que por engano ou distração o usuário pode entrar com informações inválidas que se não forem consistidas podem prejudicar a base de dados dos sistemas.

Também podemos utilizá-los para ler as informações de vetores e matrizes, e são utilizados em algoritmos de ordenação, ou seja, há uma infinidade de utilidades para os laços de repetição.

While (Enquanto)

Segue abaixo um exemplo do laço de repetição while(ver linha 3), note que enquanto a condição for verdadeira os comandos do laço serão executados. Note que no caso abaixo, cada execução é acrescentada +1 na variável x.

```
lacos_repeticao.py
1 x = 0
2
3 while x < 5:
4     nome = input ('Qual é o seu nome? ')
5     x=x+1
```

Fonte: Elaborado pelo autor (2023).

Segue abaixo a tela de execução:

```
C:\Users\joao\AppData\Local\Programs\Python\Python37\python.exe
Qual é o seu nome? Rafael
Qual é o seu nome? Bruna
Qual é o seu nome? Daniela
Qual é o seu nome? Ícaro
Qual é o seu nome? Fernando
```

Fonte: Elaborado pelo autor (2023).

No exemplo abaixo serão lidos nomes digitados pelo usuário e todo nome digitado é armazenado na variável pessoas que é uma Lista. Os nomes são adicionados através da função append().

```
lacos_repeticao.py
1 x = 0
2 pessoas = []
3
4 while x < 5:
5     nome = input ('Qual é o seu nome? ')
6     pessoas.append(nome)
7     x=x+1
8
9 print(pessoas)
```

Fonte: Elaborado pelo autor (2023).

Segue abaixo a tela de execução:

```
C:\Users\joao\AppData\Local\Programs\Python\Python37\python.exe
Qual é o seu nome? Fulano
Qual é o seu nome? Beltrano
Qual é o seu nome? Larissa
Qual é o seu nome? Maisa
Qual é o seu nome? Manoela
['Fulano', 'Beltrano', 'Larissa', 'Maisa', 'Manoela']
```

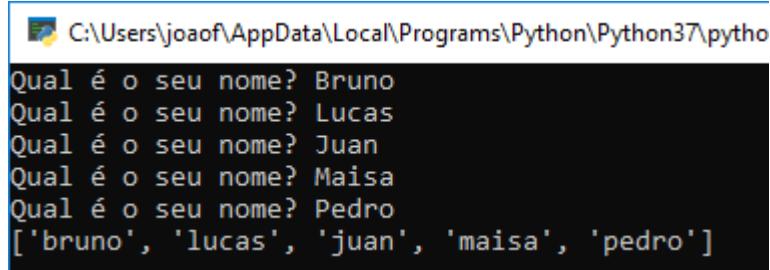
Fonte: Elaborado pelo autor (2023).

No laço de repetição o looping será eterno até que o usuário digite o nome “Pedro”

```
lacos_repeticao.py
1
2 pessoas = []
3
4 while 'pedro' not in pessoas:
5     nome = input ('Qual é o seu nome? ').lower()
6     pessoas.append(nome)
7
8 print(pessoas)
```

Fonte: Elaborado pelo autor (2023).

Segue abaixo a tela de execução:



```
C:\Users\joaof\AppData\Local\Programs\Python\Python37\python
Qual é o seu nome? Bruno
Qual é o seu nome? Lucas
Qual é o seu nome? Juan
Qual é o seu nome? Maisa
Qual é o seu nome? Pedro
['bruno', 'lucas', 'juan', 'maisa', 'pedro']
```

Fonte: Elaborado pelo autor (2023).

For (Para)

O laço de repetição FOR no exemplo abaixo, faz com que a variável “i” percorra todos os elementos da Lista até que não tenha mais elementos.

```
lacos_repeticao2.py
1 compras = ['bananas', 'queijo', 'suco', 'peixe']
2
3 for i in compras:
4     print(i)
5
```

Fonte: Elaborado pelo autor (2023).

Segue abaixo a tela de execução:

```
C:\Users\joao\AppData\Local\  
bananas  
queijo  
sucos  
peixe
```

Fonte: Elaborado pelo autor (2023).

Lembrem-se que uma string armazena caractere por caractere de um texto, ou seja, cada caractere necessita estar num índice. Portanto, se fizermos o laço de repetição utilizando o for conforme as linhas 3 e 4, iremos ler caractere por caractere da palavra que estiver armazenada na variável.

```
lacos_repeticao2.py  
1 nome = 'Fulano'  
2  
3 for i in nome:  
4     print(i)
```

Fonte: Elaborado pelo autor (2023).

Segue abaixo a tela de execução:

```
lacos_repeticao2.py  
1 faturamento = [100, 250, 220, 500, 2000]  
2 total = 0  
3  
4 for i in faturamento:  
5     total = total + i  
6  
7 print(total)
```

Fonte: Elaborado pelo autor (2023).

Podemos utilizar o “for”, para contar todos os números uma lista, conforme abaixo:

```
C:\Users\joao\AppData\  
F  
u  
l  
a  
n  
o
```

Fonte: Elaborado pelo autor (2023).

Segue abaixo a tela de execução:

```
Selecionar C:\User...
3070
```

Fonte: Elaborado pelo autor (2023).

Também podemos utilizar o “for” para exibir as chaves e os dados de um dicionário:

```
lacos_repeticao2.py
1 cores = {'branco': 'white', 'amarelo': 'yellow', 'preto': 'black'}
2
3 for i in cores:
4     print(i, ':', cores[i])
```

Fonte: Elaborado pelo autor (2023).

Segue abaixo a tela de execução:

```
C:\Users\joaof\AppData\Local\
branco : white
amarelo : yellow
preto : black
```

Fonte: Elaborado pelo autor (2023).

Podemos mostrar números ordenados conforme abaixo:

```
lacos_repeticao2.py
1 for i in range(0,10):
2     #este Looping mostrará os números ordenados de 0 até 9
3     print(i)
```

Fonte: Elaborado pelo autor (2023).

Segue abaixo a tela de execução:

```
C:\Users\|  
0  
1  
2  
3  
4  
5  
6  
7  
8  
9
```

Fonte: Elaborado pelo autor (2023).



ATIVIDADE DE FIXAÇÃO

1. Considere o algoritmo abaixo:

```
Início  
Ler os valores de A e B  
C <- 0  
Enquanto     A > B     Faça  
    Subtraia B de A, coloque o resultado em A e some 1 em C  
Fim Enquanto  
Mostre os valores finais de C e de A  
Fim
```

Fonte: Elaborado pelo autor (2023).

Adapte o algoritmo acima, e desenvolva um programa que execute as instruções para os seguintes pares de números: 10 e 2, 6 e 2, 15 e 3. Qual será o valor final de C? E o valor final de A?

2. Desenvolva um programa que escreva na tela os números de um número inicial a um número final. Os números inicial e final devem ser informados pelo usuário;
3. Desenvolva um programa que gera e escreve os números ímpares entre 100 e 200;
4. Desenvolva um programa que leia 10 números inteiros e, ao final, apresente a soma de todos os números lidos;
5. Desenvolva um programa que calcule a média dos números digitados pelo usuário, se

eles forem pares. Termine a leitura se o usuário digitar zero (0);

6. Desenvolva um programa que leia valores inteiros e encontre o maior e o menor deles. Termine a leitura se o usuário digitar zero (0);

7. Desenvolva um programa que leia o sexo de uma pessoa. O sexo deverá ser com o tipo de dado caractere e o programa deverá aceitar apenas os valores “M” ou “F”.

8. Desenvolva um programa que calcule a soma de todos os números primos existentes entre 1 e 100;

9. Desenvolva um programa para calcular o IMC (Índice de Massa Corporal), o algoritmo deverá ler a altura e o peso do usuário e depois calcular o IMC com base na fórmula: $IMC = \frac{\text{peso}}{(\text{altura})^2}$

Após o cálculo deve ser informado o valor do IMC do usuário e a classificação com base na tabela abaixo:

Classificação	IMC
Muito abaixo do peso	16 a 16,9
Abaixo do peso	17 a 18,4
Peso normal	18,5 a 24,9
Acima do peso	25 a 29,9
Obesidade Grau I	30 a 34,9
Obesidade Grau II	35 a 40
Obesidade Grau III	> 40

Fonte: Elaborado pelo autor (2023).

Deve-se utilizar um laço de repetição para consistir se o usuário digitou um valor de peso e altura maiores do que 0. Enquanto ele digitar um valor menor ou igual a zero deverá aparecer uma mensagem de erro de: “Peso e altura devem ser maiores que 0.”

10. Utilizando o laço de repetição PARA, desenvolva um programa que faça uma tabuada, para isso leia um número de 1 a 10, se o usuário digitar um valor fora desse intervalo deve aparecer uma mensagem de erro: “Digite um valor entre 1 e 10.” A tela esperada deve conter os valores da tabuada para o número digitado pelo usuário.

TEMA 13:

Validação de dados

Habilidades:

Codificar programas, utilizando técnica de programação estruturada.



Disponível em :<<https://tinyurl.com/mpzr776k>>. Acesso em 20 jul. 2023.

Os menus são importantes para gerar interação com o usuário, um sistema por mais básico que seja terá algum tipo de menu. Os menus além de necessitarem ser funcionais, também devem possuir um layout intuitivo, amigável e bonito. Os menus do sistema são a embalagem do nosso produto, não adianta ter um sistema muito bom, porém com uma tela ruim para a interação com o usuário.

Já a validação de dados é algo extremamente necessária, visto que o sistema deve prever a possibilidade de falhas humanas durante a interação com o usuário através de leitura de dados em tela. Para validar as informações digitadas pelos usuários, é necessário muitas vezes utilizar laços de repetição, como no exemplo

abaixo onde foi utilizado “while”. No exemplo abaixo, é utilizado o comando try onde podemos fazer uma condição que se não atendida fará outra ação, isso é possível através do comando except (ver linha 19).

```
exemplo_validacao.py
1 repetir = 's'
2 fatura = []
3 total = 0
4 valid_preco = False
5
6 while repetir == 's':
7     produto = input('Digite o nome do produto: ')
8
9     while valid_preco == False:
10        preco = input('Digite o preço do produto: ')
11        try:
12            preco = float(preco)
13
14            if preco <= 0:
15                print('O preço precisa ser maior que zero')
16                print(" ")
17            else:
18                valid_preco = True
19        except:
20            print("Formato de preço inválido. Use apenas números e separe os centavos com '. '")
21            print(" ")
22        fatura.append([produto,preco])
23        total += preco
24        valid_preco = False
25    repetir = input('Deseja comprar mais algum produto? (S ou N) ').lower()
26    print(" ")
```

Fonte: Elaborado pelo autor (2023).

Na linha 29 temos o contador for para toda vez que a variável 1 incrementar um valor, a rotina imprime a mensagem de qual é o valor da fatura.

Segue abaixo a tela de execução:

```
28
29 for i in fatura:
30     print(i[0],'=>',i[1])
31
32 print(" ")
33 print('O total da fatura é:',total)
34
35
```

The screenshot shows a terminal window with the following text output:

```
C:\Users\Professor ESSA\AppData\Local\Programs\Python\Python37\python.exe
Digite o nome do produto: Celular ASUS
Digite o preço do produto: blablablablablablabla
Formato de preço inválido. Use apenas números e separe os centavos com ', '
Digite o preço do produto: 999,99
Formato de preço inválido. Use apenas números e separe os centavos com ', '
Digite o preço do produto: 999.99
Deseja comprar mais algum produto? (S ou N) n
Celular ASUS => 999.99
O total da fatura é: 999.99
```

Fonte: Elaborado pelo autor (2023).



ATIVIDADE DE FIXAÇÃO

1. Qual a importância em validar os dados digitados pelo usuário?
2. Crie um programa que faça a validação para um campo de e-mail.
3. Crie um programa que faça a validação para um campo nome.
4. Crie um programa que faça a validação do CPF.
5. Crie um programa que faça a validação do RG.
6. Crie um programa que faça uma validação de data.
7. Crie um programa que faça validação de hora.
8. Crie um programa que faça validação de campo vazio.

TEMA 14

Funções e Módulos

Habilidades:

- Codificar programas utilizando técnica de programação estruturada.
- Depurar e versionar programas, utilizando ambiente de desenvolvimento integrado.



Fonte: Vídeo sobre funções em Python.

As funções servem para guardarmos um bloco de código que poderá ser reutilizado quando precisarmos. É comum nos códigos estruturados fazer uso de funções. E, para “alimentar” essas funções, precisamos passar os dados a elas. Existem dois métodos de passagem de parâmetros geralmente usados: por valor e por referência.

Quando um parâmetro é passado por valor, uma cópia do valor contido é passada à função, e este valor inicia uma variável local. Quando um parâmetro é passado por referência, o endereço de memória em que a variável está é passado para a função. A função tem acesso real ao dado passado, e toda a alteração que ocorrer dentro da função irá impactar a variável.

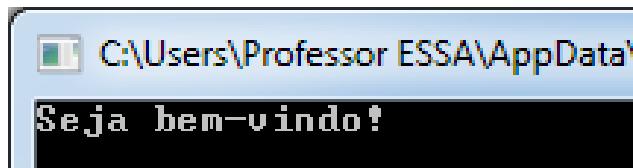
As funções também servem para organizar melhor o código fonte. Podemos ter funções que somente imprimirão em tela algum valor e teremos as funções que retornarão valores.

Abaixo segue um exemplo de uma função que faz uma saudação para o usuário.

```
exemplo_funcoes.py
1 def mensagem():
2     print('Seja bem-vindo!')
3
4 mensagem()
5
```

Fonte: Elaborado pelo autor (2023).

Segue abaixo a tela de execução:



Fonte: Elaborado pelo autor (2023).

Note abaixo que criamos a função mensagem2(), ela receberá como parâmetro uma string com o nome do usuário:

```
exemplo_funcoes.py
1 def mensagem():
2     print('Seja bem-vindo!')
3
4 def mensagem2(nome):
5     print('Seja bem-vindo ' + nome + '!')
6
7 mensagem2('Fulano')
```

Fonte: Elaborado pelo autor (2023).

Segue abaixo a tela de execução:



Fonte: Elaborado pelo autor (2023).

Abaixo criamos uma função que calcula a média, note que essa função foi criada para receber 3 parâmetros e que ela retornará a média.

```
exemplo_funcoes.py
```

```
1
2 def media_aluno(nota1,nota2,nota3):
3     media = ((nota1+nota2+(nota3*2))/4)
4     return (media)
5
6 #Lendo as notas do aluno
7 n1 = float(input('Digite a primeira nota: '))
8 n2 = float(input('Digite a segunda nota: '))
9 n3 = float(input('Digite a nota do trabalho: '))
10
11 print('')
12
13 #Chamando a função media_aluno passando os valores
14 # e atribuindo para a variável media_final o resultado
15
16 media_final = media_aluno(n1,n2,n3) ←
17
18 print('A sua media final do aluno é:',media_final)
```

Fonte: Elaborado pelo autor (2023).

Segue abaixo a tela de execução:

```
C:\Users\Professor ESSA\AppData\Local\Programs\Python\Python310\python.exe C:/Users/Professor ESSA/Downloads/exemplo_funcoes.py
Digite a primeira nota: 10
Digite a segunda nota: 5
Digite a nota do trabalho: 4
A sua media final do aluno é: 5.75
```

Fonte: Elaborado pelo autor (2023).

As funções são módulos de códigos que irão realizar as tarefas repetitivas dos programas. Sempre que surgir a necessidade, poderemos chamá-las a fim de economizar tempo, visto que escrever uma linha de código para toda tarefa repetida é algo cansativo. As funções podem ser chamadas em qualquer parte do programa e normalmente retornam algum valor.

Módulos



Fonte: <https://youtu.be/qoHCXc-sL6U>

Os módulos são arquivos com a extensão “.py”, ou seja, os arquivos que criamos até o momento são módulos. Os módulos podem ser conectados, eles abrem novas possibilidades no Python. No exemplo abaixo extrairemos do módulo “exemplo_funcoes.py” a parte que faz o cálculo da média.

```
exemplo_funcoes.py
```

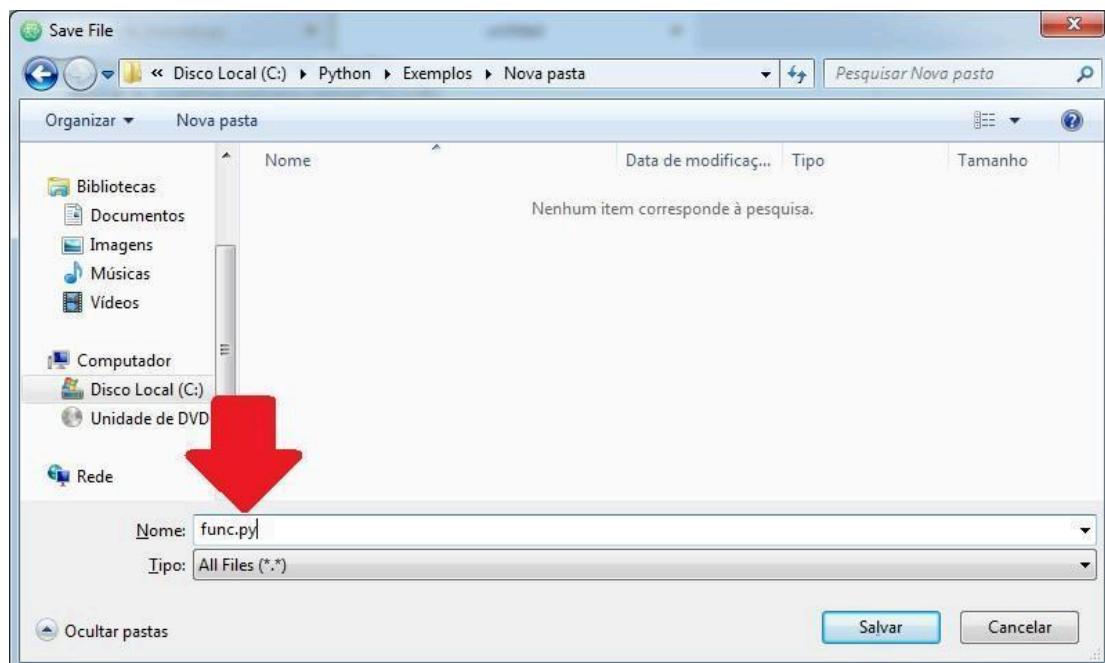
```
1 def media_aluno(nota1,nota2,nota3):
2     media = ((nota1+nota2+(nota3*2))/4)
3     return (media)
4
5
6 #Lendo as notas do aluno
7 n1 = float(input('Digite a primeira nota: '))
8 n2 = float(input('Digite a segunda nota: '))
9 n3 = float(input('Digite a nota do trabalho: '))
10
11 print('')
12
13 #Chamando a função media_aluno passando os valores
14 # e atribuindo para a variável media_final o resultado
15
16 media_final = media_aluno(n1,n2,n3)
17
18 print('A sua media final do aluno é:',media_final)
19
20
```

Fonte: Elaborado pelo autor (2023).

Para linkar os módulos são necessários que ambos estejam na mesma pasta, na tela abaixo estamos salvando o nosso módulo “func.py” que faz o cálculo da média. Note que o módulo “func” possui as informações extraídas do módulo anterior:

```
exemplo_funcoes.py • func.py
1 def media_aluno(nota1,nota2,nota3):
2     media = ((nota1+nota2+(nota3*2))/4)
3     return (media)
4
5
```

Fonte: Elaborado pelo autor (2023).

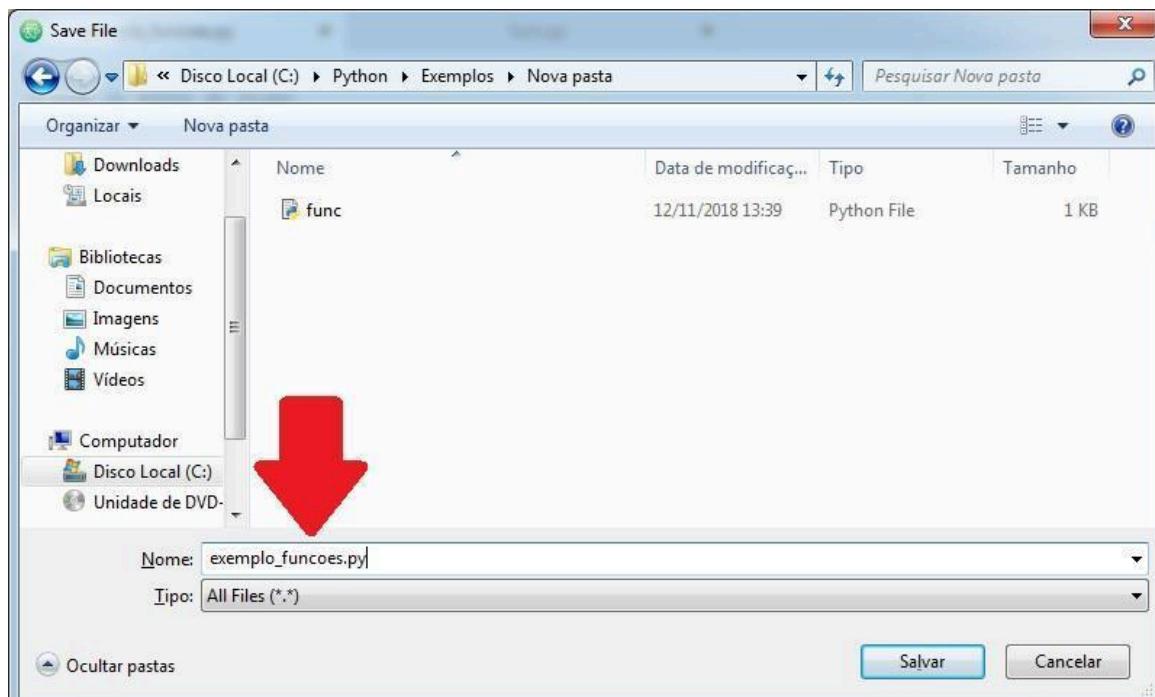


Fonte: Elaborado pelo autor (2023).



```
exemplo_funcoes.py          func.py
1
2 #Lendo as notas do aluno
3 n1 = float(input('Digite a primeira nota: '))
4 n2 = float(input('Digite a segunda nota: '))
5 n3 = float(input('Digite a nota do trabalho: '))
6
7 print('')
8
9 #Chamando a função media_aluno passando os valores
10 # e atribuindo para a variável media_final o resultado
11
12 media_final = media_aluno(n1,n2,n3)
13
14 print('A sua media final do aluno é:',media_final)
15
```

O módulo “exemplo_funcoes.py” também será salvo na mesma pasta do módulo “func.py”



Fonte: Elaborado pelo autor (2023).

```

exemplo_funcoes.py          •      func.py
1 import func ←
2 #Lendo as notas do aluno
3 n1 = float(input('Digite a primeira nota: '))
4 n2 = float(input('Digite a segunda nota: '))
5 n3 = float(input('Digite a nota do trabalho: '))
6
7 print('')
8
9 #Chamando a função media_aluno passando os valores
10 # e atribuindo para a variável media_final o resultado
11
12 media_final = func.media_aluno(n1,n2,n3)
13
14 print('A sua media final do aluno é:',media_final)
15

```

Para que o módulo exemplo_funcoes.py possa linkar o módulo “func.py” basta usarmos o comando import nome_módulo, conforme abaixo:

Fonte: Elaborado pelo autor (2023).

Segue abaixo a tela de execução, note que para o usuário não percebe a diferença se um programa está chamando outros módulos para determinadas tarefas.

```

C:\Users\Professor ESSA\AppData\Local\Programs\Python\...
Digite a primeira nota: 10
Digite a segunda nota: 9
Digite a nota do trabalho: 5
A sua media final do aluno é: 7.25

```

Fonte: Elaborado pelo autor (2023).

Módulos do próprio Python

O módulo math é um módulo matemático, note abaixo que o math.ceil() faz o arredondamento para cima de um número real:

```

exemplo_modulos_builtin.py
1 import math
2
3 print(math.ceil (3.2))

```

Fonte: Elaborado pelo autor (2023).

Segue abaixo a tela de execução:

```
C:\Users\Professor ESSA\AppData\Local\Temp\4
```

Fonte: Elaborado pelo autor (2023).

Já o módulo math.floor, faz o contrário, faz o arredondamento para baixo.

```
exemplo_modulos_builtin.py
1 import math
2
3 print(math.floor (3.7))
```

Fonte: Elaborado pelo autor (2023).

Segue abaixo a tela de execução:

```
C:\Users\Professor ESSA\AppData\Local\Temp\3
```

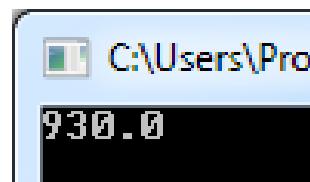
Fonte: Elaborado pelo autor (2023).

O módulo math.fsum() faz a soma de números de listas e tuplas.

```
exemplo_modulos_builtin.py
1 import math
2
3 vendas = [200,230,500]
4
5 print(math.fsum(vendas))
```

Fonte: Elaborado pelo autor (2023).

Segue abaixo a tela de execução:



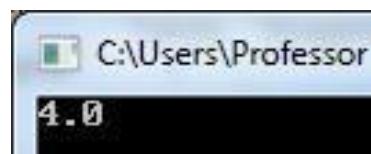
Fonte: Elaborado pelo autor (2023).

O módulo math.sqrt() faz o cálculo da raiz quadrada de um número.

```
exemplo_modulos_builtin.py
1 import math
2
3 print(math.sqrt(16))
```

Fonte: Elaborado pelo autor (2023).

Segue abaixo a tela de execução:



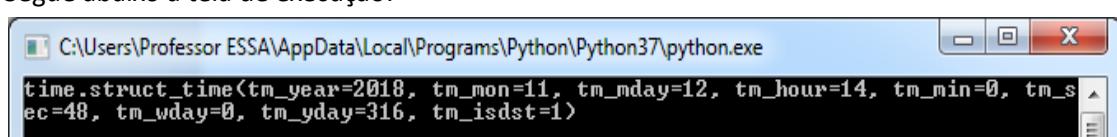
Fonte: Elaborado pelo autor (2023).

Há também módulos de tempo, por exemplo, o módulo time.localtime() trará a hora local.

```
exemplo_modulos_builtin.py
1 import time
2
3 print(time.localtime())
```

Fonte: Elaborado pelo autor (2023).

Segue abaixo a tela de execução:



Note que através do módulo time.localtime() podemos extrair mais informações como minutos, horas ou segundos, veja as linhas 3 e 5.

```
exemplo_modulos_builtin.py
1 import time
2
3 hora = time.localtime().tm_hour
4
5 minuto = time.localtime().tm_min
6
7 print('Transacao realizada às ' + str(hora) + 'h e ' + str(minuto) + ' minutos.')
8
```

Fonte: Elaborado pelo autor (2023).

Segue abaixo a tela de execução:

```
C:\Users\Professor ESSA\AppData\Local\Programs\Python\Transacao realizada às 14h e 7 minutos.
```

Fonte: Elaborado pelo autor (2023).



RESUMO

Verificamos que os módulos são programas (arquivos .py) criados por nós ou que já existem no próprio Python, utilizá-los é uma boa prática para diminuir o tamanho do código fonte do programa e para deixar o código mais organizado. Uma das vantagens em utilizarmos módulos é que eles podem ser reaproveitados por vários programas, portanto ganhamos tempo ao utilizá-los.



ATIVIDADE DE FIXAÇÃO

1. O que são funções?
2. Qual a diferença entre uma função e um procedimento?
3. Com base no exemplo 2, desenvolva um programa que irá ler dois valores inteiros. Crie uma função para cada uma das seguintes operações matemáticas: soma, subtração, multiplicação e divisão.

No final, o resultado das operações deverá ser exibido em tela.

4. Desenvolva um programa que possuirá uma função chamada de pagamento, ela deverá receber as horas trabalhadas e o valor das horas. No final o valor a ser pago para o trabalhador deverá ser retornado pela função e depois exibido em tela.

5. Desenvolva um programa que possuirá uma função que receba o nome do usuário e a idade dele, a função deverá retornar quantos dias de vida o usuário já viveu.

6. Construa uma função que receba uma data no formato DD/MM/AAAA e devolva uma string no formato D de mesPorExtenso de AAAA. Opcionalmente, valide a data e retorne NULL caso a data seja inválida.

7. O que é um módulo?

8. Quais as vantagens em utilizarmos os módulos?

9. Crie um módulo que faça o cálculo da média aritmética entre dois números, este módulo deve ser importado (chamado) através de outro programa.

10. Crie um módulo que faça o cálculo área do triângulo, este módulo deve ser importado (chamado) através de outro programa.

11. Crie um módulo que faça o cálculo do IMC (Índice de Massa Corporal), o programa que for fazer a chamada desse módulo deverá classificar através do valor retornado se a pessoa está ou não acima do peso. $IMC = \text{peso} / \text{altura}^2$



ATIVIDADE DE FIXAÇÃO



O modo de resolução é livre, mas podem ser utilizadas equações ou frases em português:

1. Descreva como descobrir a moeda falsa em um grupo de cinco moedas, fazendo uso de uma balança analítica (sabe-se que a moeda falsa é mais leve que as outras), com o menor número de pesagens possível. Lembre-se de que sua descrição deve resolver o problema para qualquer situação.

Dica: É possível resolver com apenas duas pesagens.

2. Idem ao anterior, porém só se sabe que a moeda falsa tem massa diferente. Para descobrir se ela é mais leve ou mais pesada que as outras, é necessário mudar alguma coisa?

3. Idem ao exercício 1, porém com 9 moedas.

4. Têm-se três garrafas, com formatos diferentes, uma cheia até a boca, com capacidade de oito litros e as outras duas vazias com capacidades de cinco e três litros respectivamente. Deseja-se separar o conteúdo da primeira garrafa em duas quantidades iguais. Elabore uma rotina que consiga realizar a tarefa, sem que se possa fazer medidas.

5. Um caramujo está na parede de um poço a cinco metros de sua borda. Tentando sair do poço, ele sobe três metros durante o dia, porém desce escorregando dois metros durante a noite. Quantos dias levará para o caramujo conseguir sair do poço?

6. Um tijolo “pesa” um quilo mais meio tijolo. Quantos quilos “pesa” um tijolo e meio?

7. Você está em uma margem de um rio, com três animais: uma galinha, um cachorro e uma raposa. Somente pode atravessar com um animal por vez e nunca deixar a raposa e o cachorro sozinhos nem a raposa e a galinha. Descreva uma forma de conseguir atravessar os três animais, obedecendo a essas condições.

8. Você dispõe de uma balança precisa e dez sacos cheios de moedas idênticas na aparência, das quais todas as moedas de um dos sacos são falsas e de massa 1 g menor que as verdadeiras. Qual o menor número de pesagens necessárias para se descobrir o saco de moedas falsas?

Fluxogramas

9. Crie um algoritmo e um fluxograma para converter uma temperatura da escala Celsius para a escala Fahrenheit. A fórmula para conversão é $\text{Fahrenheit} = ((\text{Celsius} * 9) / 5) + 32$

10. Elabore um fluxograma que calcule quantas notas de 50,10 e 1 são necessárias para se pagar uma conta cujo valor é fornecido.

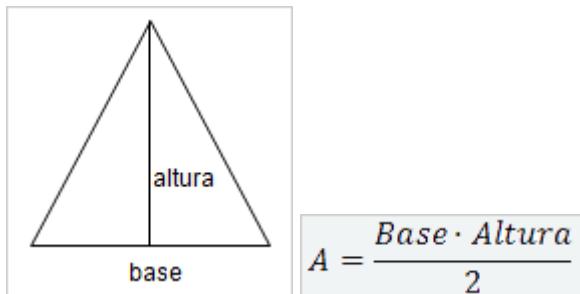
11. Elabore um fluxograma que permita a entrada de um número inteiro e diga se ele é par ou ímpar.

12. Elabore um fluxograma que permita a entrada de dois valores, x e y , troque seus valores entre si e então exiba os novos resultados.

Algoritmos diversos

13. Desenvolva um programa que armazene o valor 10 em uma variável A e o valor 20 em uma variável B. A seguir (utilizando apenas atribuições entre variáveis) troque os seus conteúdos fazendo com que o valor que está em A passe para B e vice-versa. Ao final, escrever os valores que ficaram armazenados nas variáveis.

14. Desenvolva um programa para ler as dimensões de um retângulo (base e altura), calcular e escrever a área do retângulo. Sabendo que para calcular a área devemos usar a fórmula a seguir:



Fonte: Elaborado pelo autor (2023).

15. Desenvolva um programa que leia a idade de uma pessoa expressa em anos, meses e dias e escreva a idade dessa pessoa expressa apenas em dias. Considerar ano com 365 dias e mês com 30 dias.

16. Desenvolva um programa que calcule e exiba a soma dos números pares contidos entre zero e um número par fornecido pelo usuário.

17. A contribuição para o INSS é calculada a partir da tabela a seguir:

TABELA VIGENTE - 2018	
Salário de contribuição (R\$)	Aliquota para fins de recolhimento ao INSS (%)
Até R\$ 1.693,72	8%
De R\$ 1.693,72 até R\$ 2.822,90	9%
De R\$ 2.822,91 até R\$ 5.645,80	11%

Fonte: *Elaborado pelo autor (2023)*.

- Desenvolva um programa para ler o valor de um salário bruto para que seja calculado o valor da contribuição ao INSS e o salário líquido restante.

18. O desconto do IRRF (Imposto de Renda Retido na Fonte), também denominado “Mordida do Leão”, é calculado sobre o salário líquido após a dedução da contribuição ao INSS, de acordo com a seguinte tabela:

Base de cálculo mensal (R\$)	Aliquota (%)	Parcela a deduzir do IR (%)
Até 1.903,98	isento	isento
De 1.903,99 até 2.826,65	7,5%	142,80
De 2.826,66 até 3.751,05	15%	354,80
De 3.751,06 até 4.664,68	22,5%	636,13
Acima de 4.664,68	27,5%	869,36

Fonte: *Elaborado pelo autor (2023)*.

- Desenvolva um programa que irá ler o valor do salário bruto e após a dedução da contribuição ao INSS, calcule o desconto do IRRF.

19. Desenvolva um programa que permita ao usuário escolher entre a conversão de medida de centímetros em polegadas, de polegadas em centímetros, de quilômetros em milhas e de milhas em quilômetros. As fórmulas as fórmulas para conversão são:

- Centímetros/polegadas: valor em centímetros x 0,3937
- Polegadas/centímetros: valor em centímetros x 2,54
- Quilômetros/milhas: valor em quilômetros x 0,6214
- Milhas/quilômetros: valor em milhas x 1,6093

20. Desenvolva um programa que leia um número inteiro e verifique se este é ou não um número primo (número primo é divisível por um e por ele mesmo).

21. Um número palíndromo é aquele que se lido da esquerda para a direita e da direita para a esquerda possui o mesmo valor (por exemplo: 34543). Desenvolva um programa que leia um número inteiro e verifique se ele é um palíndromo.

Estruturas de repetição

9. Considere o algoritmo abaixo:

```
Início
    Ler os valores de A e B
    C <- 0
    Enquanto    A > B    Faça
        Subtraia B de A, coloque o resultado em A e some 1 em C
    Fim Enquanto
    Mostre os valores finais de C e de A
Fim
```

Fonte: Elaborado pelo autor (2023).

- Adapte o algoritmo acima, e desenvolva um programa que execute as instruções para os seguintes pares de números: 10 e 2, 6 e 2, 15 e 3. Qual será o valor final de C? E o valor final de A?
22. Desenvolva um programa que escreva na tela os números de um número inicial a um número final. Os números inicial e final devem ser informados pelo usuário;
23. Desenvolva um programa que gera e escreve os números ímpares entre 100 e 200;
24. Desenvolva um programa que leia 10 números inteiros e, ao final, apresente a soma de todos os números lidos;
25. Desenvolva um programa que calcule a média dos números digitados pelo usuário, se eles forem pares. Termine a leitura se o usuário digitar zero (0);
26. Desenvolva um programa que leia valores inteiros e encontre o maior e o menor deles. Termine a leitura se o usuário digitar zero (0);
27. Desenvolva um programa que leia o sexo de uma pessoa. O sexo deverá ser com o tipo de dado caractere e o programa deverá aceitar apenas os valores “M” ou “F”.
28. Desenvolva um programa que calcule a soma de todos os números primos existentes entre 1 e 100;

Vetores e matrizes

1. Desenvolva um programa que calcule e exiba a diferença entre o maior e o menor elemento de um vetor denominado VALORES (com 10 elementos). Os valores do vetor devem ser lidos.

2. Desenvolva um programa que faça a soma de todos os elementos de índice par de um vetor de tamanho 10.

3. Desenvolva um programa que leia dois vetores A e B de tamanho 5 e então troque seus elementos, de forma que o vetor A ficará com os elementos do vetor B e vice-versa.

4. Desenvolva um programa que utilizará a Cifra de César para criptografar uma mensagem digitada pelo usuário, após mostrar em tela a mensagem criptografada.

5. Desenvolva um programa que leia um vetor N[20]. A seguir, encontre o menor elemento do vetor N e a sua posição dentro do vetor, mostre esses valores em tela.

6. Desenvolva um programa para armazenar valores inteiros em uma matriz (5,6). A seguir, calcular a média dos valores pares contidos na matriz e escrever seu conteúdo.

7. Desenvolva um programa para ler uma matriz (7,4) contendo valores inteiros (supor que os valores são distintos). Após, encontrar o menor valor contido na matriz e sua posição.

8. Desenvolva um programa que leia uma matriz M (5,5) e calcula as somas:
 - Da linha 4 de M.
 - Da coluna 2 de M.
 - Da diagonal principal.
 - Da diagonal secundária.
 - De todos os elementos da matriz.
 - Escreva estas somas e a matriz.



REFERÊNCIAS

CASTRO, Bruno; VIEIRA, Rui. Python: Introdução à Resolução de Problemas com Algoritmos. Rio de Janeiro: Editora Ciência Moderna, 2012.

FORBELLONE, André Luiz Villar; EBERSPÄCHER, Henri Frederico. Lógica de Programação: A Construção de Algoritmos e Estruturas de Dados. 4. ed. São Paulo: Pearson Prentice Hall, 2019.

GAMA, Eric. Python Fluente: Programação Clara, Concisa e Eficaz. São Paulo: Novatec, 2016.

GOMES, Cláudio; FERNANDES, Antônio. Lógica de Programação e Estrutura de Dados: Algoritmos em Java. Rio de Janeiro: LTC, 2015.

LEMOS, Sandro; WIEDEMANN, Nelson. Python: Escreva seus primeiros programas. São Paulo: Digerati Books, 2017.

LUTZ, Mark. Aprendendo Python: Do Iniciante ao Profissional. São Paulo: Novatec, 2014.

MANZANO, José Augusto N. G.; OLIVEIRA, Jayr Figueiredo de. Algoritmos: Lógica para Desenvolvimento de Programação de Computadores. 28. ed. São Paulo: Érica, 2018.

PRATA, Bruno. Lógica de Programação: A Construção de Algoritmos e Estruturas de Dados. 1. ed. São Paulo: Casa do Código, 2017.

VIEIRA, Sérgio. Python para Desenvolvedores. 2^a edição. São Paulo: Novatec, 2010.

ZIVIANI, Nívio. Projeto de Algoritmos com Implementações em Pascal e C. 4. ed. São Paulo: Cengage Learning, 2015.



Viva sua profissão!