Tópico 1

Lógica, Algoritmos, pseudocódigos e fluxogramas.

Você já percebeu que a lógica está presente em todas as áreas da ciência, tais como a medicina, física, matemática e psicologia?

Algoritmos

Os algoritmos são mais antigos do que se imagina, a ideia já era pauta em debate no século VXII, claro que na imaginação de filósofos e matemáticos chamados na época de loucos como, Gottfried Von Leibniz, que já previa a existência de máquinas de calcular universais e estranhas linguagens simbólicas. A matemática clássica é em grande parte o estudo de determinados algoritmos.

Um algoritmo é uma sequência de lógica e finita de instruções que devem ser seguidas para a resolução de um problema ou execução de uma tarefa. Os algoritmos são amplamente utilizados nas disciplinas ligadas à área de ciências exatas, tais como matemática, física, química e computação, além de ter aplicação ampla em outras áreas.

Algoritmos aplicados a soluções computacionais

Tipos de algoritmos

Existem diversos tipos de algoritmos descritos na literatura. Para nossos estudos vamos utilizar os dois tipos mais clássicos, que são pseudocódigos e fluxograma:

- Pseudocódigo: Utilizam a linguagem estruturada que pode ser chamada de português estruturado, para demonstrar de forma didática como programas de computador podem solucionar problemas e tarefas de nosso cotidiano de forma otimizada.

- Fluxograma: Formatos universais para demostrar através de formas geométricas como programas de computador podem solucionar problemas e tarefas de nosso cotidiano, de maneira a representar os passos de todo o processo.

Pseudocódigos

Para representar como o pseudocódigo funciona é dado um exemplo do Algoritmo 1 para demostrar o processo de entrada e saída dos dados.

Algoritmo 1 – Ler nome

```
1.  ALGORITMO
2.          ler_nome
3.  VAR
4.          nome:  LITERAL
5.  INICIO
6.          LER (nome)
7.          MOSTRAR ("Meu nome é ", nome)
8.  FIM
```

No Algoritmo 1 a ideia é representar através de linguagem estruturada um programa de computador que leia e guarde o nome de uma determinada pessoa e em seguida mostre na tela.

Este processo é explicado pela seguinte estrutura:

Identificação do algoritmo: que no exemplo recebe o nome de ler_nome.

Declaração da variável: que receberá na etapa 3 um valor do tipo LITERAL (texto, número, caracteres, etc.) identificado por nome.

Corpo do pseudocódigo: o onde ocorre o processo de resolução do problema, no qual o valor armazenado na variável nome é lido e processado para ser mostrado na tela. Se fosse digitado a palavra "Vitor" e em seguida executado o algoritmo, teríamos apresentado na tela a seguinte informação:

Meu nome é Vítor

Fluxograma

O fluxograma é um tipo de algoritmo que utiliza gráficos para representar as ações ou instruções a serem seguidas. Assim como um pseudocódigo, o fluxograma é utilizado para organizar o raciocínio lógico a ser seguido para resolução de um problema ou para definir os passos de uma tarefa.

O fluxograma, por utilizar figuras para representação das ações, é considerado um algoritmo universal, ou seja, é uma forma universal de representação também conhecido como diagrama de blocos.

Representação Fluxograma

Terminal

O início e o final do fluxograma

Processamento

É a execução de operações ou ações como cálculos aritméticos

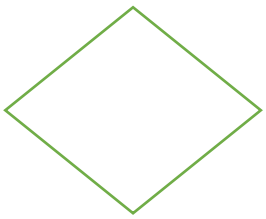
Teclado

É a entrada de dados para as variáveis por meio do teclado

Vídeo

Saída de informações (dados de mensagens), por meio do monitor de vídeo

Decisão



Representa uma ação lógica que resultará na escolha de uma das sequências de instruções, ou seja, se o teste lógico apresentar o resultado "verdade", realizará uma sequência e, se o teste lógico apresentar o resultado falso, realizará outra sequência,

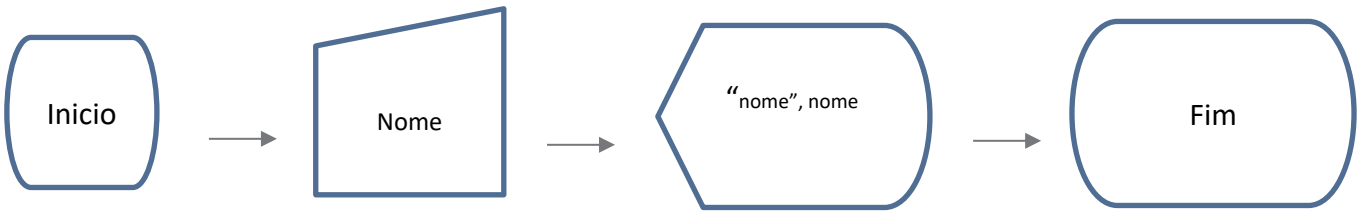
Preparação
É uma preparação para o processamento, ou seja um processamento predefinido

Conector
Utilizado para interligar partes do fluxograma ou para desviar o fluxo corrente para um determinado trecho do fluxograma

Conector e páginas
Utilizado para interligar partes do fluxograma em páginas distantes.

Seta de orientação do fluxo
Orientação a sequência de execução ou leitura, pode ser horizontal ou vertical

Fluxograma 1 – "Ler nome"

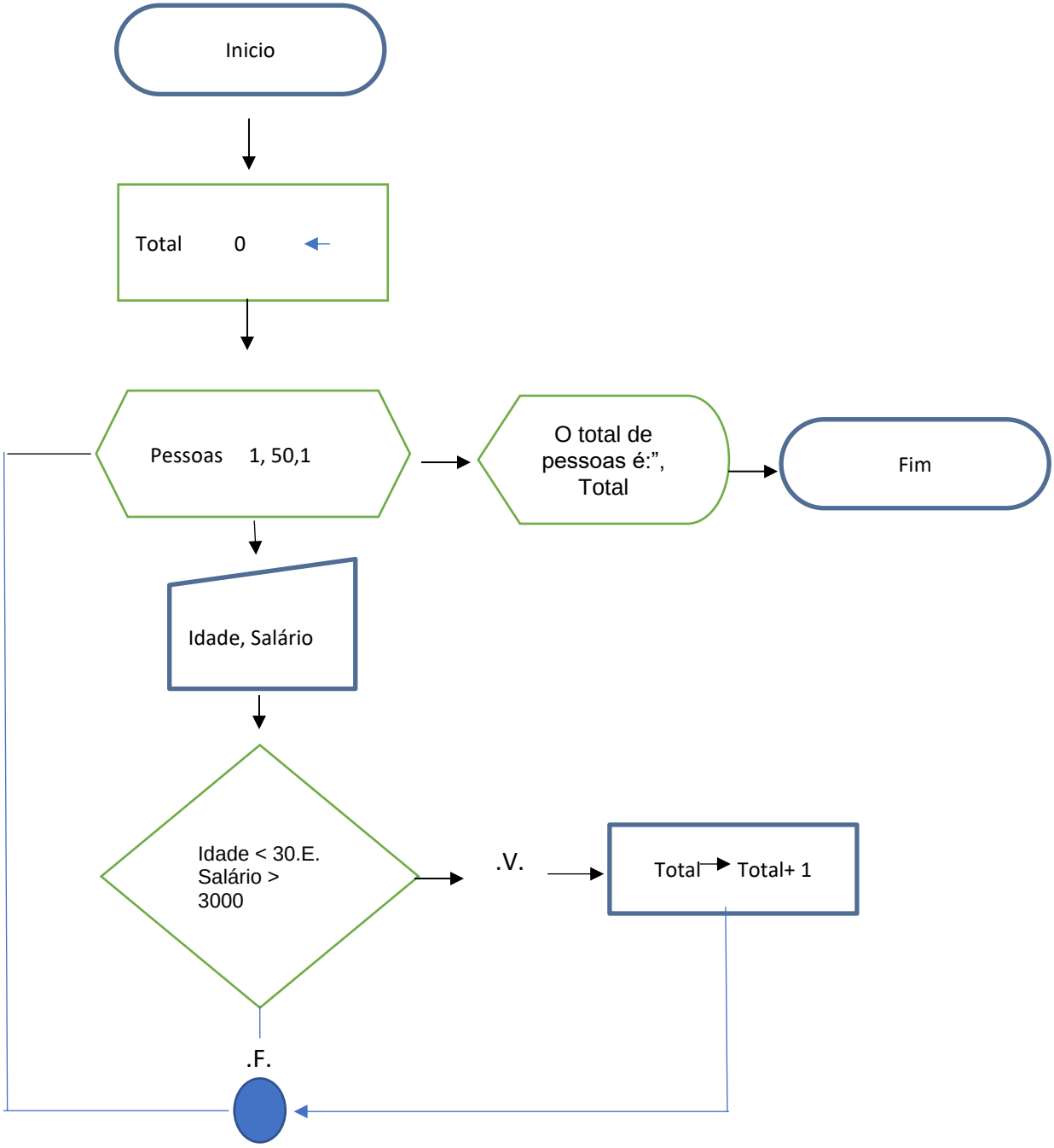


O Exemplo acima é a representação do mesmo algoritmo, "Ler nome" apresentado no subtítulo dos pseudocódigos, só que agora em forma de fluxograma, o que nada mais é que a representação gráfica do algoritmo.

Podemos perceber que a visualização gráfica torna mais fácil a percepção do objetivo a ser realizado pela tarefa a que o programa se propõe a realizar.

No fluxograma abaixo está representado um algoritmo para ler a idade e o salário de 50 pessoas e verificar quantas tem idade inferior a 30 e um salário superior a 3.000,00.

Fluxograma 2 – "Salário maior"



Linguagem de Programação

Conceitos de programação

Após o estudo realizado no tópico anterior, algumas dúvidas devem estar latentes em seu pensamento, portanto você pode estar se perguntando!

O que é linguagem de programação?

Um programa é um conjunto de instruções escrito em uma determinada linguagem que diz ao computador o que deve ser feito. Existem muitas formas e diferentes tipos de linguagens de programação, cada qual com uma finalidade específica. As linguagens de programação podem ser classificadas em níveis, chamados de baixo nível, alto nível. A designação de alto nível se dá pelo fato de que nele utilizam-se linguagens que procuram aproximar-se das linguagens naturais, usadas pelas pessoas.

Linguagens de Baixo Nível

São linguagens de programação que atuam diretamente no hardware da máquina, movimentando dados e acionando dispositivos ligados ao computador.

Linguagem de Alto Nível

Como a linguagem de baixo nível é de difícil programação, foram criados outros níveis diferentes de linguagens. Assim surgiram os programas digitados na linguagem de alto nível, constituem o que se chama de código fonte, o qual é convertido e (traduzido) para programas de baixo nível, em um processo denominado de **compilação** ou **interpretação.** Sendo traduzido o programa será executado pela máquina reproduzindo aquilo que o programador deseja.

Para fins de estudo, nesta disciplina serão tratadas as linguagens de programação de alto nível, como Pascal, C, C++ e C#, que são utilizadas pelos programadores no desenvolvimento de sistemas. Embora essa linguagem tente aproximar-se da linguagem humana esse objetivo está longe de ser alcançado. O mais perto que se conseguiu chegar foi a criação de instruções mnemônicas, em inglês, para facilitar o processo.

Mnemônico é um conjunto de técnicas utilizadas para auxiliar o processo de memorização. Consiste na elaboração de suportes como os esquemas, gráficos, símbolos, palavras ou frases relacionadas com o assunto que se pretende memorizar. Recorrer a esses suportes promove uma rápida associação e permite uma melhor assimilação do conteúdo.

Linguagem de programação

A linguagem de programação é formada por palavras, essas são agrupadas em frases para produzir um determinado significado. Dessa forma, podem-se chamar a linguagem de programação de palavra-chave, e as frases criadas com essas palavras de estrutura de programação.

Você já deve ter percebido que como na comunicação cotidiana entre as pessoas, a linguagem de programação possui uma sintaxe, definida por regras. E igual a comunicação humana, existem inúmeros tipos de linguagens. Sendo assim você deve estar se perguntando:

Por que existem tantos tipos de linguagem?

Uma linguagem é melhor que a outra?

A resposta para as duas perguntas está no objetivo para o qual foram criadas essas linguagens. Cada uma delas foi criada para um devido fim, para cumprir uma função ou resolver um tipo de problema. Uma linguagem pode ser melhor para cálculos ao mesmo tempo que outra pode ser boa para processamento de dados e assim como outra pode elaborar uma interface de fácil interação com o usuário. Sendo assim pode-se perceber que as linguagens de programação são criadas de acordo com sua adequação e aplicação, a exemplo disso, podemos citar as mais utilizadas:

Exemplos de linguagens de programação

Nome	Função	Exemplo
Pascal	Criada para ser uma ferramenta educacional, pela simplicidade de sua sintaxe, vem sendo utilizada até hoje pelos meios acadêmicos. É considerada de alto nível.	`Write ("Algoritmos e Estrutura de Dados");`
C	Tem grande aplicação no desenvolvimento de software básico e aplicações com forte interação com hardware.	O exemplo citado acima pode ser escrito em C da maneira abaixo: `Printf("Algoritmos e Estrutura de dados");`
C++	É uma evolução do C, que preserva princípios de eficiência e facilidade de programação.	O exemplo citado acima pode ser escrito em C++, da maneira abaixo: `Cout<<" Algoritmos e Estrutura de dados";`
Java	De grande aplicação no mercado, é amplamente utilizada em aplicações de processamento distribuído e para a Internet.	Reproduzindo o exemplo acima: `System.out.println("Algoritmos e Estrutura de Dados");`

PHP	O PHP (Hypertext Preprocessor) é uma linguagem interpretada, gratuita e usada originalmente apenas para o desenvolvimento de aplicações presentes e atuantes no lado do	A linha de código a seguir exemplifica uma instrução em PHP, echo é responsável por imprimir dados na tela do usuário.
		`<?php`
		`echo "Algoritmos e Estrutura de dados";`
		`?>`

	servidor, capazes de gerar conteúdo dinâmico na web. O PHP é uma linguagem totalmente voltada à Internet e os sites dinâmicos são aqueles que retornam para o cliente uma página criada em tempo real.	

Tipos de Linguagem de programação

Assim como os computadores, ao longo dos tempos foram criadas muitas linguagens de programação e estas evoluíram de acordo com o avanço dos sistemas computacionais. Algumas sobreviveram ao longo dos anos e foram reformuladas de acordo com suas particularidades e funcionalidades, sendo adaptadas para uma nova realidade de hardware e sistemas.

As primeiras linguagens eram sequenciais e eram chamadas de **"programação Linear"**. Passado algum tempo surgiram as linguagens estruturadas, ou **programação estruturada**, que permitiram o desenvolvimento de sistemas mais interativos. Mais recentemente a **programação orientada a objetos** que promoveu uma grande transformação na forma como os sistemas são concebidos e codificados, trazendo grande interatividade. Esta última, **programação orientada a objetos** será abordado com mais profundidade no decorrer do curso, mais precisamente no último módulo do curso.

Programação Linear

É aquela que cria programas que ao serem executados, obedecem de forma rigorosa, a uma sequência de passos consecutivos, com início e fim definidos. Esse princípio foi muito utilizado antigamente, nas primeiras linguagens criadas, o Assembly é um bom exemplo desse tipo de linguagem que acabou sendo a inspiração para a construção do "Basic". Para se ter ideia da dificuldade que era trabalhar com programação antigamente, abaixo está um exemplo de linguagem "Basic".

```
10  READ A1, A2, A3, A4
15  LET D = A1 * A4 - A3 * A2
20  IF D = 0 THEN 65
30  READ B1, B2
37  LET X1 = (B1*A4 - B2 * A2 ) / D
42  LET X2 = ( A1 * B2 - A3 * B1)/D
55  PRINT X1, X2
60  GØ TØ 30
65  PRINT "NØ UNIQUE SØLUTIØN"
70  DATA 1, 2,4
80  DATA 2, -7, 5
85  DATA 1, 3, 4, -7
90  END
```

A desvantagem deste tipo programação está na complexidade, programas lineares extensos são difíceis de ser desenvolvidos e até compreendidos. Programação Estruturada

Programação Estruturada

Não é novidade que a divisão de um trabalho muito extenso em pequenas tarefas, torna mais fácil a obtenção do resultado. A programação estruturada é a divisão do trabalho em tarefas que são realizadas passo a passo até que o objetivo seja atingido. Um bom programador deve atribuir funcionalidades a um programa, deve separar as tarefas que esta precisa realizar e depois ataca-las uma a uma, tornando o trabalho menos assustador.

Na programação estruturada chamamos a divisão de tarefas de modularização, que nada mais é que um processo onde um programa é dividido em partes ou módulos que executam tarefas específicas.

Como exemplo podemos observar o funcionamento de um time de futebol em campo: cada jogador tem sua tarefa dentro de campo e um objetivo a atingir, que é o gol, também há um tempo para isso ser executado são 90 minutos, se todos executarem suas funções dentro de campo, ou se todas as tarefas forem divididas adequadamente, será grande a chance de que todas sejam cumpridas no tempo determinado e o gol adversário seja atingido dentro dos noventa minutos. Porém se no meio deste processo dois jogadores do mesmo time resolverem brigar e mudar a tática empregada pelo técnico, as tarefas serão corrompidas e com certeza o tempo não será suficiente e a falha será inevitável.

As tarefas são executadas por blocos, denominados **procedimentos** e **funções,** cada bloco de códigos recebe um nome. A diferença entre eles está no

fato de que os procedimentos recebem valores, mas não retornam valor como resultado, e as funções retornam resultados de todas as operações realizadas.

Procedimentos geralmente são palavras e pequenas frases: `AtualizarDados()`

Funções podem ser executadas como varáveis porque retornam o valor: `Soma(x,y)`

A seguir, para exemplificar apresentamos um programa em C, o texto que começa com duas barras (//) é apenas comentário, que pode ser colocado para explicar o programa, não faz parte do programa:

```
1. // Exemplo de programa em C
2. // Isto é uma linha de comentário
3. void main()
4. {
5. int a;
6. // declara a variável "a"
7. a = 3 + 2;         // soma 3 com 2
8. }
```

Um programa em C é composto por um conjunto de Funções. A função pela qual o programa começa a ser executado chama-se main.

Após cada comando em C deve-se colocar um; (ponto-e-vírgula). O programa em C deve ser identificado para que possa ser lido com mais facilidade. Estudaremos mais a frente sobre procedimentos e funções.

Ambiente de Desenvolvimento de programas.

Para nossos estudos utilizaremos um ambiente de desenvolvimento de programação ou IDE – Integrated Development Environment (Ambiente de desenvolvimento integrado). Nesta disciplina podemos usar o CodeBlock, VS-Code e Dev-C.

Na informática, existem ambientes específicos onde se pode trabalhar com a programação. Esses ambientes são chamados de IDE, o que consiste em **um software que contém um conjunto de funcionalidades embutidas, cuja finalidade é prover um modo mais fácil e interativo de construir e manipular programas de computadores.**

Dentro de um IDE geralmente figuram as seguintes ferramentas:

Um editor de texto com facilidades especialmente desenhadas para a linguagem; um compilador (e um interpretador, no caso de Java e outras linguagens interpretadas); um editor gráfico, com facilidades para criação e edição da interface gráfica do programa a ser desenvolvido e por fim um Debugger, uma ferramenta especialmente feita para se tirar os bugs do código.

O Debugger possibilita um monitoramento mais elegante do funcionamento do seu programa, facilitando a detecção e remoção dos erros. Perceba que não estamos falando em erros de sintaxe, mas erros na própria lógica do programa, que fazem seu programa gerar resultados indesejados ou travar (apesar de ele compilar), e que geralmente são difíceis de se encontrar simplesmente analisando o código.

Detalhando um exemplo

A escolha do IDE depende do objetivo e da experiência do desenvolvedor, a seguir vamos analisar um exemplo de aplicação em C#, que faz um cálculo da operação potenciação:

Primeira parte do exemplo: São as bibliotecas de uso do c, onde ficam registrados todos os dados de manipulação da linguagem, sempre que se vai escrever um programa em C, é preciso chama-las existem muitas, bata saber qual utilizar e para que cada uma serve.

Em nossos estudos o importante são as duas primeiras, pois são o padrão de uso geral, o include<math.h> é para operações constantes da matemática, no nosso exemplo estamos definindo uma potência, portanto a math deve ser incluída na aplicação.

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
```

Segunda parte do exemplo: `int main(void) {`

Int – declaração de números inteiros

(Void) – indica que este método realizará uma tarefa, mas que não terá nenhum valor no retorno.

(main) – indica ao computador que este é o método principal, a parte que receberá atenção inicial a execução do programa.

Terceira parte do exemplo: `printf("Qual e o numero? \n");` É a saída de dados do programa com uma quebra de linha `\n.`

Quarta parte do exemplo:

O comando abaixo irá fazer a leitura dos dados inseridos pelo usuário e irá armazenar na variável nro.

```
scanf("%d",&nro);
```

`int nro:` é a variável de número inteira inserida pelo usuário

`float` quadrado: é a variável de número real

`pow(nro,2)` e retorna o resultado que é exibido na caixa de mensagem.

A saída usando o printf para imprimir os dados na tela.

```
printf("Quadrado = %.0f\n",quadrado);
```