Linguagem e Técnicas de Programação

Prof(a). Karhyne S P Assis
Prof. Fábio F Assis

1. Introdução

1.1.Teoria: Algoritmos

- 1.1.1. Conceitos básicos
- 1.1.2. Programas
- 1.1.3. Representação lógica
- 1.1.4. Regras
- 1.1.5. Fases
- 1.1.6. Tipos de Algoritmos

1.2.Laboratório: Introdução à linguagem C

- 1.2.1. Histórico
- 1.2.2. Regras da Linguagem

2. Introdução à programação

2.1.Teoria: Tipos de informações e entrada e saída

- 2.1.1. Tipos de Dados
- 2.1.2. Variáveis, constantes e operadores
- 2.1.3. Entrada e saída de informações
- 2.1.4. Programação sequencial

2.2. Laboratório: Linguagem C; Tipos de informações e entrada e saída

- 2.2.1. Tipos de Dados
- 2.2.2. Variáveis, constantes e operadores
- 2.2.3. Entrada e saída de informações

COMPUTAÇÃO

Objetivos

Compreender os conceitos básicos de lógica de programação

Objetivos específicos: Aprender conceitos básicos de computação, aprender e praticar a escrita de algoritmos, para resolução de problemas básicos.

COMPUTADOR

Função dos computadores: Os computadores são projetados para executar e entender apenas algumas tarefas bem simples e curtas, tais como:

Somar dois números, ou seja, o computador não efetua operações com três ou mais números de uma única vez;

Copiar um conjunto de dados de uma parte da memória para outra;

Verificar o resultado de uma operação se é zero.

COMPUTAÇÃO

A computação pode ser definida como a busca de uma solução para um problema a partir de entradas (*inputs*) e tem seus resultados (*outputs*), após o processamento por meio de um algoritmo.

Um computador é uma máquina, conjunto de partes eletrônicas, eletromecânicas e sistemas (Hardware e Software) capaz de sistematicamente coletar, manipular e fornecer os resultados.

SISTEMA DE COMPUTAÇÃO

O sistema de computação pode ser entendido como um conjunto de componentes integrados para funcionar como se fossem um único elemento, tendo como objetivo realizar manipulações com dados, ou seja, por meio dos dados de entrada realizar um processamento de modo a obter uma informação (dado de saída).

O primeiro computador tinha os componentes necessários para realizar as etapas de processamento de dados: Processador e Sistema de Entrada/Saída (E/S), ou do inglês: *Input*/*Output* (I/O).

LISTA DE SÍMBOLOS:

∧ Conector “e”

∨ Conector “ou”

⊻ Conector “ou” exclusivo

⟹ Conector “Se...então...”

⟺ Conector “Se e somente se”

¬ Negação de uma proposição, também pode ser representado pelo sinal ~

TABELA VERDADE:

O uso da tabela verdade, aplicações da tabela verdade

REGRAS BÁSICAS DAS OPERAÇÕES LÓGICAS:

Operador	Símbolo	Nome da Proposição	Regra
E	$\wedge$	Conjunção	Será verdadeiro caso todas as proposições sejam verdadeiras
Ou	$\vee$	Disjunção	Será verdadeiro caso pelo menos uma das proposições seja verdadeira
Não	$\sim ou \neg$	Negação	Negação da proposição
Se... Então...	$\rightarrow$	Condicional	Só será falsa quando a proposição antecedente for verdadeira e a consequente for falsa
Se e somente se	$\leftrightarrow$	Bicondicional	Será verdadeira quando ambas as proposições forem verdadeiras ou ambas falsas

Tabela 1. Exemplo de regras de operações lógicas

REGRAS BÁSICAS DAS OPERAÇÕES LÓGICAS:

Estrutura Lógica	***É verdade quando***	***É falso quando***
p ∧ **q**	**p** e **q** são, ambos, verdade	um dos dois for falso
p V q	um dos dois for verdade	**p** e **q**, ambos, são falsos
~p	**p** é falso	**p** é verdade
p → q	Nos demais casos	**p** é verdade e **q** é falso
p↔q	**p** e **q** tiverem valores lógicos iguais	**p** e **q** tiverem valores lógicos diferentes

Tabela 2. Exemplo de regras de operações lógicas

TABELA VERDADE

Construção da tabela verdade

Exemplo 1: tabela verdade de p

p
V
F

Exemplo 2: tabela verdade de q

q
V
F

TABELA VERDADE

Construção da tabela verdade

Exemplo 1: tabela verdade de p

p
V
F

Exemplo 2: tabela verdade de q

q
V
F

Exemplo 3: tabela verdade de $\neg p$

p	$\neg p$
V	F
F	V

Exemplo 4: tabela verdade de $\neg q$

q	$\neg \boldsymbol{q}$
V	F
F	V

TABELA VERDADE

Construção da tabela verdade

Exemplo 5: tabela verdade de $p \wedge q$

Primeira parte

p	q
V	V
V	F
F	V
F	F

TABELA VERDADE

Construção da tabela verdade

Exemplo 5: tabela verdade de $p \wedge q$

Primeira parte

p	q
V	V
V	F
F	V
F	F

Por fim, aplicamos o conectivo na sentença

p	q	$p \wedge q$
V	V	V
V	F	F
F	V	F
F	F	F

TABELA VERDADE

Construção da tabela verdade

Exemplo 6: tabela verdade de $p \vee q$

Primeira parte

p	q
V	V
V	F
F	V
F	F

TABELA VERDADE

Construção da tabela verdade

Exemplo 6: tabela verdade de $p \vee q$

Primeira parte

p	q
V	V
V	F
F	V
F	F

Por fim, aplicamos o conectivo na sentença

p	q	$p \vee q$
V	V	V
V	F	V
F	V	V
F	F	F

TABELA VERDADE

Construção da tabela verdade

p	q
V	V
V	F
F	V
F	F

p	q	r
V	V	V
V	V	F
V	F	V
V	F	F
F	V	V
F	V	F
F	F	V
F	F	F

p	q	r	s
V	V	V	V
V	V	V	F
V	V	F	V
V	V	F	F
V	F	V	V
V	F	V	F
V	F	F	V
V	F	F	F
F	V	V	V
F	V	V	F
F	V	F	V
F	V	F	F
F	F	V	V
F	F	V	F
F	F	F	V
F	F	F	F

Tabela Verdade - Exercícios

1 – Construa a tabela verdade de $(\neg p \wedge q)$:

Resolução: uma solução é montar a tabela verdade e separar a sentença em partes

Primeira parte

p	q
V	V
V	F
F	V
F	F

Tabela Verdade - Exercícios

1 – Construa a tabela verdade de $(\neg p \wedge q)$:

Resolução: uma solução é montar a tabela verdade e separar a sentença em partes

Primeira parte

p	q
V	V
V	F
F	V
F	F

Segunda parte $\neg p$

p	q	$\neg p$
V	V	F
V	F	F
F	V	V
F	F	V

Tabela Verdade - Exercícios

1 – Construa a tabela verdade de $(\neg p \wedge q)$:

Resolução: uma solução é montar a tabela verdade e separar a sentença em partes

Primeira parte

p	q
V	V
V	F
F	V
F	F

Segunda parte $\neg p$

p	q	$\neg p$
V	V	F
V	F	F
F	V	V
F	F	V

Por fim, aplicamos o conectivo $\wedge$ na sentença

p	q	$\neg p$	q	$(\neg p \wedge q)$
V	V	F	V	F
V	F	F	F	F
F	V	V	V	V
F	F	V	F	F

Tabela Verdade - Exercícios

2 – Construa a tabela verdade de $(\neg p \vee q)$:

Resolução: uma solução é montar a tabela verdade e separar a sentença em partes

Primeira parte

p	q
V	V
V	F
F	V
F	F

Tabela Verdade - Exercícios

2 – Construa a tabela verdade de $(\neg p \vee q)$:

Resolução: uma solução é montar a tabela verdade e separar a sentença em partes

Primeira parte

p	q
V	V
V	F
F	V
F	F

Segunda parte $\neg p$

p	q	$\neg p$
V	V	F
V	F	F
F	V	V
F	F	V

Tabela Verdade - Exercícios

2 – Construa a tabela verdade de $(\neg p \vee q)$:

Resolução: uma solução é montar a tabela verdade e separar a sentença em partes

Primeira parte

p	q
V	V
V	F
F	V
F	F

Segunda parte $\neg p$

p	q	$\neg p$
V	V	F
V	F	F
F	V	V
F	F	V

Por fim, aplicamos o conectivo V na sentença

p	q	$\neg p$	q	$(\neg p \vee q)$
V	V	F	V	V
V	F	F	F	F
F	V	V	V	V
F	F	V	F	V

Tabela Verdade - Exercícios

3 – Construa a tabela verdade de $(\neg p \wedge \neg q)$:

Resolução:

Tabela Verdade - Exercícios

3 – Construa a tabela verdade de $(\neg p \wedge \neg q)$:

Resolução: uma solução é montar a tabela verdade e separar a sentença em partes

Primeira parte

p	q
V	V
V	F
F	V
F	F

Tabela Verdade - Exercícios

3 – Construa a tabela verdade de $(\neg p \wedge \neg q)$:

Resolução: uma solução é montar a tabela verdade e separar a sentença em partes

Primeira parte

p	q
V	V
V	F
F	V
F	F

Segunda parte $\neg p$

p	q	$\neg p$
V	V	F
V	F	F
F	V	V
F	F	V

Tabela Verdade - Exercícios

3 – Construa a tabela verdade de $(\neg p \wedge \neg q)$:

Resolução: uma solução é montar a tabela verdade e separar a sentença em partes

Primeira parte

p	q
V	V
V	F
F	V
F	F

Segunda parte $\neg p$

p	q	$\neg p$
V	V	F
V	F	F
F	V	V
F	F	V

Terceira parte

p	q	$\neg p$	$\neg q$
V	V	F	F
V	F	F	V
F	V	V	F
F	F	V	V

Tabela Verdade - Exercícios

3 – Construa a tabela verdade de $(\neg p \wedge \neg q)$:

Resolução: uma solução é montar a tabela verdade e separar a sentença em partes

Primeira parte

p	q
V	V
V	F
F	V
F	F

Segunda parte $\neg p$

p	q	$\neg p$
V	V	F
V	F	F
F	V	V
F	F	V

Terceira parte $\neg q$

p	q	$\neg p$	$\neg q$
V	V	F	F
V	F	F	V
F	V	V	F
F	F	V	V

Por fim, aplicamos o conectivo $\wedge$ na sentença

P	q	$\neg p$	$\neg q$	$(\neg p \wedge \neg q)$
V	V	F	F	F
V	F	F	V	F
F	V	V	F	F
F	F	V	V	V

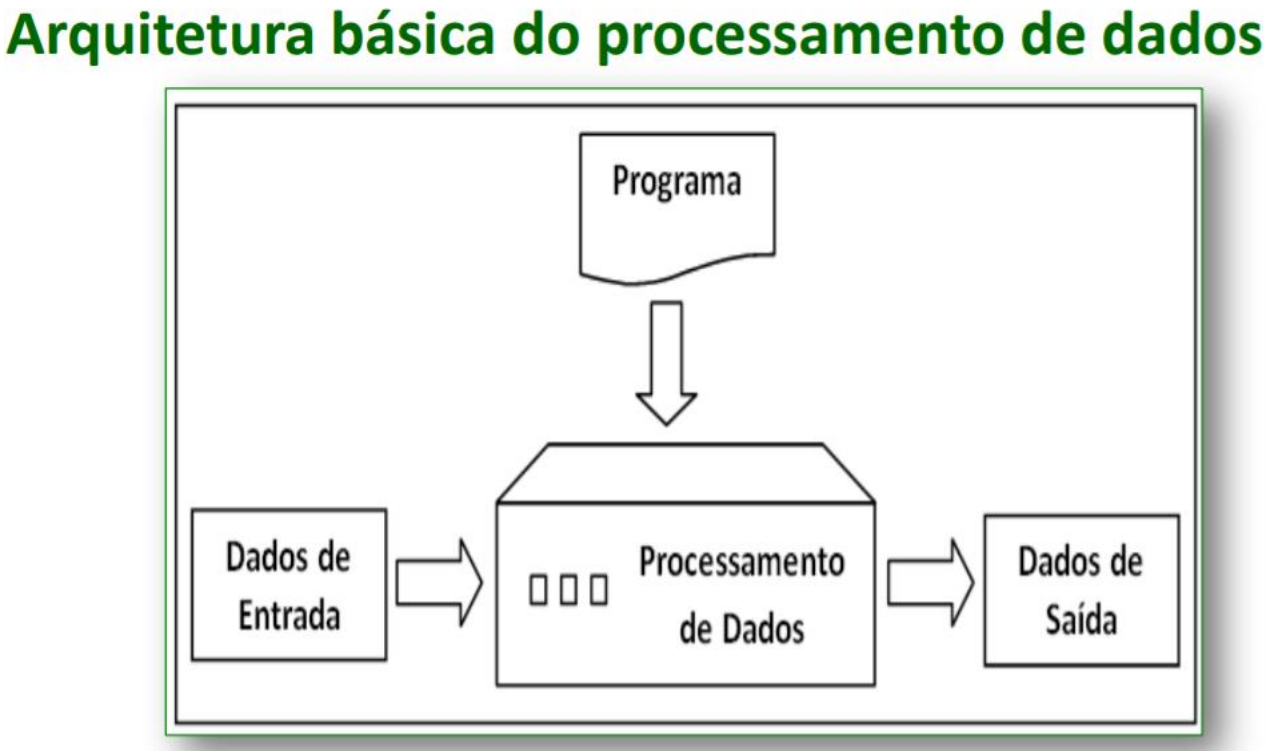


Figura 1. Exemplo de processamento de dados

LÓGICA

A lógica consiste simplesmente na organização e explicação de um pensamento. Podemos identificar a utilização da lógica no nosso dia-a-dia.

LÓGICA DE PROGRAMAÇÃO

Lógica de programação é a organização coesa de uma sequência de instruções voltadas à resolução de um problema, ou à criação de um software ou aplicação.

ALGORITMO

Um algoritmo é um conjunto de regras e operações bem definidas e ordenadas de forma lógica destinadas à solução de um problema, em um número finito de etapas;

Os algoritmos são utilizados para manipular dados nas estruturas de várias formas, como por exemplo: inserir, excluir, procurar, modificar ou ordenar dados.

O principal componente de um programa de computador é a sua lógica, ou seja, o que o computador deve fazer para resolver um problema.

ALGORITMO

Os algoritmos, portanto, podem ser divididos em partes, que são elas:

Início;

Entrada: dados que devem ser lidos, ou fornecidos pelo usuário;

Processamento: são executados os cálculos, os dados de entrada são manipulados a fim de se gerar uma resposta (um resultado);

Saída: são os resultados de seu processamento;

Fim.

ALGORITMO – Bloco de Execução

Bloco de execução é um conjunto de ações que possui uma função bem definida.

O início de um bloco é marcado pela palavra início.

O fim de um bloco é marcado pela palavra fim.

```
Início
        - variáveis
        - comandos
fim
```

ALGORITMO – Bloco de Execução

A sintaxe da definição do bloco de execução de um algoritmo é:

Algoritmo <nomeDoAlgoritmo>

início

- variáveis

- comandos

fimAlgoritmo.

Algoritmo nome do algoritmo;

Var
declaração de variáveis;

Início
corpo do algoritmo;

Fim.

ALGORITMO – Bloco de Execução

Os algoritmos, portanto, podem ser divididos em partes, que são elas:

Início;

Entrada: dados que devem ser lidos, ou fornecidos pelo usuário;

Processamento: são executados os cálculos, os dados de entrada são manipulados a fim de se gerar uma resposta (um resultado);

Saída: são os resultados de seu processamento;

Fim.

ALGORITMO – Exemplo: Algoritmo troca lâmpada

Passos para realizar a troca de uma lâmpada

1 - Pegar uma escada.

2 - Posicionar a escada embaixo da lâmpada.

3 - Buscar uma lâmpada nova.

4 - Subir na escada.

5 - Retirar a lâmpada velha.

6 - Colocar a lâmpada nova.

ALGORITMO – Exemplo: Algoritmo troca lâmpada

Passos para realizar a troca de uma lâmpada

1 - Pegar uma escada.

2 - Posicionar a escada embaixo da lâmpada.

3 - Buscar uma lâmpada nova.

4 - Subir na escada.

5 - Retirar a lâmpada velha.

6 - Colocar a lâmpada nova.

Devemos procurar desenvolver algoritmos que resolvam o problema com menor esforço e maior objetividade.

EXEMPLOS DE PROBLEMAS

Criar um sistema de informação que permita gerir as entradas/saídas (cargas/descargas) de caminhões em uma empresa.

Desenvolver um programa para controle da movimentação de um drone.

Desenvolver um programa para calcular operações financeiras para um banco.

ALGORITMO

Passos para a construção do algoritmo (programa)

Identificar o problema (objetivo).

Identificar os resultados desejados (saída de dados) .

Identificar os dados a serem fornecidos para o algoritmo (entrada de dados).

Determinar o que deve ser feito (processamento).

VARIÁVEIS

Variável é tudo que está sujeito a variações. Todo dado a ser armazenado na memória de um computador deve ser previamente identificado segundo seu tipo.

Identificamos uma variável em um programa de computador pelo seu nome, também chamado de identificador.

A declaração de variáveis deve seguir as regras de cada linguagem de programação.

Figura 2. Exemplo de variável

MEMÓRIA

A memória é usada para armazenar os dados processados pelo computador. Os dados guardados por um dispositivo de memória são sempre sequências de bits, isto é, dígitos binários 0-1 escritos em base 2.

Podemos inserir em cada endereço da memória um único valor (seja ele numérico, lógico ou caractere).

TIPOS DE DADOS

– Caractere: incluem um caractere ou um conjunto deles, incluindo letras, símbolos e números.

– Inteiro: um número inteiro é aquele que não possui parte fracionária, podendo representar quantidades positivas, negativas ou zero.

– Real: número real em computação pode representar números inteiros ou com parte fracionária positivos e negativos, além do zero, também chamado de número de ponto flutuante.

– Lógico: um dado lógico armazena 1 bit para representar o valor 1 (verdadeiro) ou o valor 0 (falso).

LINGUAGEM

Uma linguagem (ou pseudo linguagem) possui duas características:

Sintaxe - como escrever os comandos e seus componentes;

Semântica - o significado de cada comando e conceito.

Conceitos Básicos

PROGRAMA

Programas (*Softwares*) são construídos, por meio de linguagens de programação.

LINGUAGEM DE PROGRAMAÇÃO

É um conjunto limitado de símbolos e comandos padronizado, utilizados para desenvolvimento de programas.

Por meio da linguagem de programação se estabelece uma comunicação com o computador (*hardware*).

LINGUAGEM DE PROGRAMAÇÃO

Baixo nível: uma linguagem de programação de baixo nível, esta mais próxima hardware, ou seja, o computador a interpreta com menor esforço, porém para nós humanos é mais difícil sua compreensão.

Neste caso, o programador precisa entender além da linguagem em si, mas também da arquitetura do dispositivo que ele irá trabalhar. Um exemplo de linguagem de baixo nível é a linguagem ASSEMBLY.

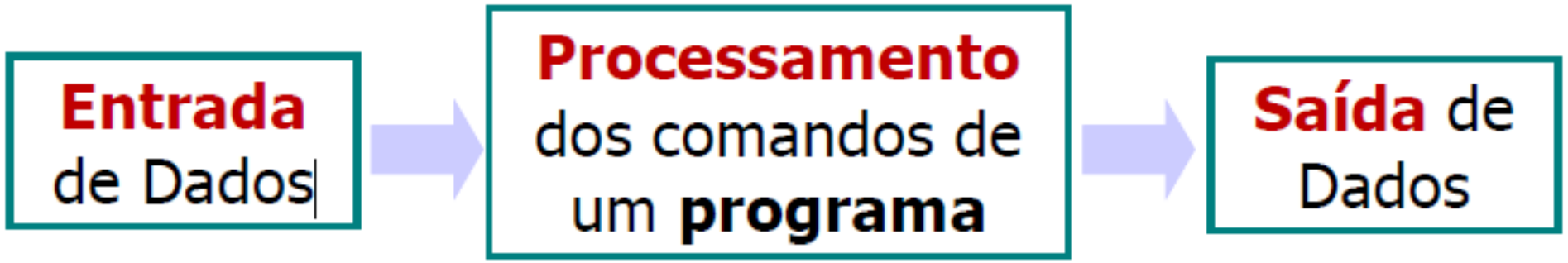
LINGUAGEM DE PROGRAMAÇÃO

Alto nível: uma linguagem de programação de alto nível, esta mais próxima do entendimento da humanos, ou seja, este tipo de linguagem apresenta uma sintaxe mais próxima da linguagem natural humana. Exemplos é a linguagem de programação: C, C++, C#, Java, Python.

Os programas escritos em linguagens de alto nível são convertidos para a linguagem de máquina através de um programa compilador, ou de um interpretador.

PROGRAMA

Podemos definir programa, como uma sequência de instruções (comandos) que, a partir de dados inseridos, obtêm um resultado que será disponibilizado por algum dispositivo de saída



EXPRESSÕES

Aritméticas (+ - * / ^ sqrt mod)
Lógicas (.NÃO. .E. .OU.)
Relacionais (> < >= <= = <>)
Literais (concatenação/cópia)

OPERADORES RELACIONAIS

Operador	Comparação	Precedência (ordem)
<	Menor	1ª
<=	Menor ou igual	1ª
>	Maior	1ª
>=	Maior ou igual	1ª
=	Igual	2ª
<>	Diferente	2ª

PRIORIDADE ENTRE OPERADORES

Operação	***Precedência (ordem)***
Operadores aritméticos/literais	1ª
Operadores relacionais	2ª
Operadores lógicos	3ª

PRIORIDADE ENTRE OPERADORES

Operadores Aritméticos +, -, *, /, \ (divisão inteira), MOD(resto da divisão inteira) ^ (potenciação)

Operadores Relacionais =, <=, >=, <, >

Operadores Lógicos ou, nao, e, xou

Entrada de Dados leia

Saída de Dados escreva

****Caso utilize a biblioteca *math.h* em C, comando *include* <*math.h*>, ou *Math* Java, comando *import java.lang.Math;* podemos trabalhar com funções prontas para cálculo com potência, raiz quadrada, outros.

PRIORIDADE ENTRE OPERADORES

Quando várias operações ocorrem em uma expressão, cada parte é avaliada e resolvida em uma ordem predeterminada chamada precedência do operador.

Quando as expressões contêm operadores de mais de uma categoria, os operadores aritméticos são avaliados primeiro, os operadores de comparação são avaliados em seguida, e os operadores lógicos são avaliados por último.

Operadores de comparação todos têm precedência; ou seja, eles são avaliados na ordem da esquerda para a direita em que aparecem.

PRIORIDADE ENTRE OPERADORES

Quando a multiplicação e a divisão ocorrem em conjunto em uma expressão, cada operação é avaliada conforme ela ocorre da esquerda para a direita.

Quando a adição e a subtração ocorrem juntas em uma expressão, cada operação é avaliada em ordem de aparência da esquerda para a direita.

PRIORIDADE ENTRE OPERADORES

Operadores aritméticos e lógicos são avaliados na seguinte ordem de precedência:

Aritmética	Comparação	Lógica
Exponenciação (^)	Igualdade (=)	Not
Negação (-)	Inequality (< >)	And
Multiplicação e divisão (*, /)	Menor que (<)	Or
Divisão de inteiro (\)	Maior que (>)	Xor
Aritmética de módulo (Mod)	Menor que ou igual a (< =)	Eqv
Adição e subtração (+, -)	Maior ou igual a (> =)	Imp
Concatenação de cadeia (&) de caracteres	Like, é	

https://docs.microsoft.com/pt-br/office/vba/language/reference/user-interface-help/operator-precedence

ESTRUTURA SEQUÊNCIAL

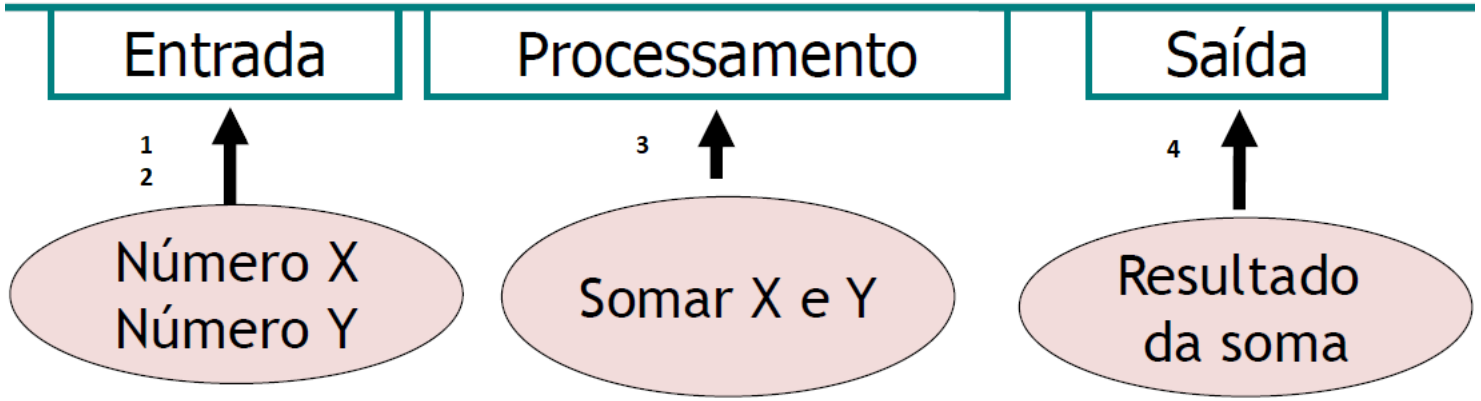
Uma estrutura sequencial é um conjunto de comandos que serão executados em uma sequência linear, de cima para baixo.

Os comandos serão executados na mesma ordem em que foram escritos. Até o momento todos os exemplos e exercícios foram resolvidos com estruturas sequenciais.

EXEMPLO DE ALGORITMO

Algoritmo (Programa) que soma dois números e exibe o resultado na tela

```
1. Leia o número X
2. Leia o número Y
3. Some X e Y
4. Mostre o resultado da soma
```



Exemplo:

Tabela de tipos de dados em C

Tabela de tipos inteiros [editar | editar código-fonte]

Convém ver a tabela de tipos inteiros.

Tipo	Num de bits	Formato para leitura com scanf	Intervalo Inicio	Fim
char	8	%c	-128	127
unsigned char	8	%c	0	255
signed char	8	%c	-128	127
int	16	%d	-32.768	32.767
unsigned int	16	%u	0	65.535
signed int	16	%i	-32.768	32.767
short int	16	%hi	-32.768	32.767
unsigned short int	16	%hu	0	65.535
signed short int	16	%hi	-32.768	32.767
long int	32	%li	-2.147.483.648	2.147.483.647
signed long int	32	%li	-2.147.483.648	2.147.483.647
unsigned long int	32	%lu	0	4.294.967.295

Fonte: https://pt.wikibooks.org/wiki/Programar_em_C/Tipos_de_dados

Exemplo 1

EXEMPLO DE ALGORITMO

Programa que exibe a mensagem Olá Mundo!

```
programa
{
        funcao inicio ()
        {
                escreva("Olá Mundo!\n")
        }
}
```

```
#include <stdio.h>
int main(void)
{
    printf("Olá Mundo!\n");

    return 0;
}
```

```
print("Olá Mundo!")
```

** Pesquisar em livros ou na Internet sobre a Sintaxe Básica da Linguagem de Programação da disciplina.
** A barra invertida (\) é chamada de **caractere de escape**. Ela indica que printf deve fazer algo fora do comum

EXEMPLO DE ALGORITMO 1.2

Programa que exibe a mensagem Olá Mundo!

```c
1  /* Figura 2.1: fig02_01.c
2     Primeiro programa em C */
3  #include <stdio.h>
4
5  /* função main inicia execução do programa */
6  int main( void )
7  {
8     printf( "Bem-vindo a C!\n" );
9
10    return 0; /* indica que o programa terminou com sucesso */
11 } /* fim da função main */
```

```
Bem-vindo a C!
```

#**include <stdio.h>** é uma diretiva do **pré-processador C**. As linhas que começam com # são verificadas pelo pré-processador antes de o programa ser compilado. A linha 3 diz ao pré-processador que inclua no programa o conteúdo do **cabeçalho-padrão de entrada/saída** (**<stdio.h>**). Esse cabeçalho contém informações usadas pelo compilador quando a compilação solicita as funções da biblioteca-padrão de entrada/saída, como printf

int main(void) faz parte de todo programa em C. Os parênteses depois de main indicam que main é um bloco de construção de programa chamado de **função**. O void entre parênteses significa que main não

```c
/* Figura 2.1: fig02_01.c
   Primeiro programa em C */
#include <stdio.h>

/* função main inicia execução do programa */
int main( void )
{
   printf( "Bem-vindo a C!\n" );

   return 0; /* indica que o programa terminou com sucesso */
} /* fim da função main */
```

```
Bem-vindo a C!
```

A função **scanf** lê o dado de entrada-padrão, que normalmente é o teclado. Esse scanf tem dois argumentos, "%d" e &inteiro1.

- "%d" (a **string de controle de formato**) indica o tipo de dado que deve ser digitado pelo usuário. O **especificador de conversão %d** indica que os dados devem ser um inteiro (a letra d significa 'inteiro decimal').
- O segundo argumento de scanf começa com um **(&)** — chamado de **operador de endereço** em C —, seguido pelo nome da variável. O (&), quando combinado com o nome da variável, informa à scanf o local (ou endereço) na memória em que a variável inteiro1 está armazenada.

Exemplo 2

EXEMPLO DE ALGORITMO

Programa que exibe soma dois números e exibe o resultado na tela

```
1.  programa
2.  {
3.          funcao inicio()
4.          {
5.                  inteiro a, b, soma
6.
7.                  escreva("Digite o valor de a: ")
8.                  leia(a)
9.                  escreva("Digite o valor de b: ")
10.                 leia(b)
11.
12.                 soma = a + b            // Soma os dois valores
13.
14.
15.                 escreva("\nA soma dos números é igual a: ", soma) // Exibe o resultado
16.         }
17. }
```

Declaração de Variáveis

Entrada de Dados

Entrada de Dados

Processamento

Saída de Dados

Exemplo 2 Algoritmo

EXEMPLO DE ALGORITMO

Programa que exibe soma dois números e exibe o resultado na tela

```
1.  programa
2.  {
3.          funcao inicio()
4.          {
5.                  inteiro a, b, soma
6.
7.                  escreva("Digite o valor de a: ")
8.                  leia(a)
9.                  escreva("Digite o valor de b: ")
10.                 leia(b)
11.
12.                 soma = a + b            // Soma os dois valores
13.
14.
15.                 escreva("\nA soma dos números é igual a: ", soma) // Exibe o resultado
16.         }
17. }
```

Declaração de Variáveis

Entrada de Dados

Entrada de Dados

Processamento

Saída de Dados

Tabela de execução do algoritmo passo a passo.

Linha	a	b	soma	tela
7				Digite o valor de a:
8	12			
9				Digite o valor de b:
10		3		
12			15	
15				Exibe texto e o resultado na tela

Exemplo 3

EXEMPLO DE ALGORITMO

Programa que calcula a soma de dois números e exibe o resultado na tela em linguagem de programação C. Podemos utilizar um ambiente de desenvolvimento integrado online, por exemplo https://ideone.com, Para executar os códigos em linguagem C.

Portugol Webstudio — Sem título

```
1  programa {
2      funcao inicio() {
3
4
5          inteiro a, b, soma
6          escreva("Digite o valor de a: ")
7          leia(a)
8
9          escreva("Digite o valor de b: ")
10         leia(b)
11
12         soma = a + b
13         escreva("\nA soma = ", soma)
14     }
15 }
```

</> source code

```
1   #include <stdio.h>
2
3   int main(void)
4   {
5       int a, b, soma;
6       printf("Digite o valor de A: \n");
7       scanf("%d", &a);
8
9       printf("Digite o valor de B: \n");
10      scanf("%d", &b);
11
12      soma = a+b;
13      printf("soma = %d", soma);
14      return 0;
15  }
```

```
// programa em C

#include <stdio.h>
int main(void)
{
    int a, b, soma;
    printf("Digite o primeiro número: ");
    scanf("%d", &a);

    printf("\nDigite o segundo número: ");
    scanf("%d", &b);

    soma = a + b;

    printf("\nSoma = : %d", soma);

    return 0;
}
```

Exemplo 4

EXEMPLO DE ALGORITMO

Programa que multiplica dois números e exibe o resultado na tela.

```
programa
{
    funcao inicio()
    {
        inteiro a, b, mult
        escreva("Digite o primeiro número: ")
        leia(a)

        escreva("\nDigite o segundo número: ")
        leia(b)

        mult = a * b // Multiplica os dois valores

        // Exibe o resultado da multiplicação
        escreva("\nA multiplicação dos números é : ", mult)
    }
}
```

```
// programa em C
#include <stdio.h>
int main(void)
{
    int a, b, mult;
    printf("Digite o primeiro número: ");
    scanf("%d", &a);

    printf("\nDigite o segundo número: ");
    scanf("%d", &b);

    mult = a * b;

    printf("\nA multiplicação dos números é: %d", mult);

    return 0;
}
```

Exemplo 5

Faça um programa que exibe na tela o número digitado pelo usuário.

Solução:

Exemplo 5

Faça um programa que exibe na tela o número digitado pelo usuário.

Solução:

```
programa
{
    funcao inicio()
    {
        inteiro numero
        escreva("Digite um número: ")
        leia(numero)

        escreva("\nO número digitado foi: ", numero)
    }
}
```

```
// programa em C

#include <stdio.h>
int main(void)
{
    int numero;
    printf("Digite um número: ");
    scanf("%d", &numero);

    printf("\nO número digitado foi: %d",
    numero);

    return 0;
}
```

Exemplo 6

EXEMPLO DE ALGORITMO

Algoritmo (Programa) que soma dois números exibe o resultado na tela

```
programa
{
	funcao inicio()
	{
		inteiro numeroX=0, numeroY=0, soma=0
		escreva("\nDIGITE o primeiro NÚMERO \n")
		leia(numeroX)

		escreva("\nDIGITE o segundo NÚMERO \n")
		leia(numeroY)

		soma = numeroX + numeroY

		escreva("\nSOMA = ", soma,"\n")
	}
}
```

Exemplo 7

EXEMPLO DE ALGORITMO

Algoritmo (Programa) que soma dois números exibe o resultado na tela

```
1  programa
2  {
3      funcao inicio()
4      {
5          inteiro a=1, b=3, soma
6
7          soma = a + b
8
9          escreva("\n\nSOMA = ", soma)
10     }
11 }
```

Exemplo 8

EXEMPLO DE ALGORITMO

Algoritmo (Programa) que soma dois números exibe o resultado na tela

```
1  programa
2  {
3
4      funcao inicio()
5      {
6          inteiro a = 2, b = 3
7          //a operação de soma foi efetuada dentro da função escreva
8
9          escreva("SOMA = ", a+b,"\n")
10     }
11 }
```

Exemplo 9

EXEMPLO DE COMPARAÇÃO

Programa que recebe do usuário um nome digitado e exibe o resultado na tela.

```
programa
{
    funcao inicio()
    {
        cadeia nome
        escreva("Digite seu nome: ")
        leia(nome)

        // Exibe o resultado
        escreva("\nNome: ", nome)
    }
}
```

```
// programa em C

#include <stdio.h>
int main(void)
{
        char nome[10];
        scanf("%s", nome);

        //gets(nome);

        printf("nome: %s", nome);

    return 0;
}
```

Exemplo 10

EXEMPLO DE ALGORITMO

Programa que multiplica dois números e exibe o resultado na tela.

```
programa
{
    funcao inicio()
    {
        inteiro a, b, soma, mult
        escreva("Digite o primeiro número: ")
        leia(a)

        escreva("\nDigite o segundo número: ")
        leia(b)

        mult = a * b // Multiplica os dois valores

        // Exibe o resultado da multiplicação
        escreva("\nA multiplicação dos números é : ", mult)
    }
}
```

```
// programa em C

#include <stdio.h>
int main(void)
{
    int a, b, soma, mult;
    printf("Digite o primeiro número: ");
    scanf("%d", &a);

    printf("\nDigite o segundo número: ");
    scanf("%d", &b);

    mult = a * b;

    printf("\nA multiplicação dos números é: %d", mult);

    return 0;
}
```

AULA 01

ATIVIDADE

1 – Faça um programa que calcula a soma de dois números e exibe o resultado na tela.

2 – Faça um programa que calcula o produto de dois números digitados pelo usuário.

3 – Faça um programa que calcula a área de um retângulo, ou seja, o usuário digita o valor do lado X e o valor do lado Y do retângulo (tipo real), depois faça o cálculo da área exiba o resultado na tela.

4 – Leia uma temperatura em graus Celsius e apresente-a convertida em graus Fahrenheit. A fórmula para conversão é: $F = C*(9.0/5.0)+32.0$, sendo F a temperatura em Fahrenheit e C a temperatura em Celsius.

5 – Leia uma temperatura em graus Fahrenheit e apresente-a convertida em graus Celsius. A fórmula para conversão é: $C = 5.0 * (F - 32.0)/9.0$, sendo C a temperatura em Celsius e F a temperatura em Fahrenheit.