



---

---

---


---

---

---


---

---



### Plano de Ensino

- Apresentação e Revisão
- **Introdução à Teoria da Computação.**
- **Conceitos Básicos de Teoria da Computação.**
  - Programas, Máquinas e Computações.
  - Modelos Computacionais.
  - Máquinas Universais.
  - Tese de Church.
  - Máquina de Turing.



---

---

---


---

---

---


---

---



### Livro-Texto

- Bibliografia Básica:
  - » LEWIS, Harry R.; PAPADIMITRIOU, Christos H. **Elementos da Teoria da Computação.** 2ª ed. Porto Alegre: Bookman, 2000.
  - » SIPSER, Michael. **Introdução à Teoria da Computação.** 2ª ed. São Paulo: Cengage Learning, 2011.



---

---

---

---

---

---

---

---

## 2. Introdução à TC – Áreas de Estudo



- Existem 3 áreas focais da Teoria da Computação: Autômatos, Computabilidade e Complexidade.
- Interligadas pela questão:

### ***Quais são as capacidades e limitações fundamentais dos computadores?***

- Em cada uma das áreas acima, a questão acima é interpretada de forma distinta e suas respostas também.
- Início dos estudos em 1930 → lógicos matemáticos iniciaram os estudos sobre o significado da computação.

---

---

---

---

---

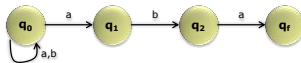
---

---

## 2. Introdução à TC – Áreas de Estudo



- Teoria dos Autômatos:
  - » Trabalha com definições e propriedades de modelos matemáticos de computação.
  - » Estes modelos possuem papel em diversas áreas aplicadas da Ciência da Computação:
    - Autômato finito → processamento de texto, compiladores e projeto de hardware.
    - Gramática livre de contexto → linguagens de programação e inteligência artificial.
  - » Os autômatos utilizam definições formais utilizadas em áreas não-teóricas da computação.



---

---

---

---

---

---

---

## 2. Introdução à TC – Áreas de Estudo



- Teoria da Computabilidade:
  - » Matemáticos descobriram que existem problemas básicos que não podem ser resolvidos por nenhum algoritmo computacional.
  - » Consequentemente, desenvolveu-se ideias de modelos teóricos de computadores para construção de computadores reais.
  - » A teoria da computabilidade classifica os problemas em solúveis e não-solúveis.

---

---

---

---

---

---

---

## 2. Introdução à TC – Áreas de Estudo



- Teoria da Complexidade:
  - » Problemas computacionais possuem diferentes níveis de complexidade: alguns são fáceis, outros difíceis.
  - » O que faz alguns problemas computacionais difíceis e outros fáceis é uma questão central da teoria da complexidade e notavelmente sem resposta.
  - » Possui relação intrínseca com a teoria da computabilidade, sendo que esta última classifica um problema em solúvel/não-solúvel e a teoria da complexidade em difícil/não-difícil.

---

---

---

---

---

---

---

## 2. Conceitos de TC – Programas



- Pode ser descrito como um conjunto estruturado de instruções que capacitam uma máquina a aplicar operações básicas e testes sobre os dados fornecidos.
- Possui uma estrutura de controle de operações e testes.
- Formas de estruturação de controle:
  - » Monolítica → desvios condicionais e incondicionais, não possuindo mecanismos explícitos de iteração, subdivisão ou recursão.
  - » Iterativa → controle de iterações de trechos de programas; não permite desvios incondicionais.
  - » Recursiva → estruturação em sub-rotinas recursivas; não permite desvios incondicionais.

---

---

---

---

---

---

---

## 2. Conceitos de TC – Programas



- Independente da estrutura de controle, duas ou mais operações ou testes podem ser compostos:
  - » Sequencial: a execução da operação ou teste subsequente somente pode ser realizada após o encerramento da execução anterior.
  - » Não-determinística: uma das operações ou testes compostos é escolhido para ser executada → também conhecida como escolha.
  - » Concorrente: as operações ou testes compostos podem ser executados em qualquer ordem, inclusive simultaneamente.

---

---

---

---

---

---

---

## 2. Conceitos de TC – Programas



- Para o estudo de programas, as operações e testes serão identificados pelos seus nomes, conforme descrito:
  - » Identificadores de operações: F, G, H, ...
  - » Identificadores de testes: T.
- Um identificador de teste produz somente um valor verdade: *verdadeiro* (*v*) ou *falso* (*f*).
- Se a operação não faz nada; ela é denominada vazia (✓).

---

---

---

---

---

---

---

## 2. Conceitos de TC – Programa Monolítico



- Programa estruturado usando desvios condicionais e incondicionais.
- Não faz uso explícito de mecanismos auxiliares de programação que permita uma melhor estruturação como iteração, subdivisão ou recursão.
- Pode ser descrito através de fluxograma (diagramas) ou na forma de textos (instruções rotuladas).

---

---

---

---

---

---

---

## 2. Conceitos de TC – Programa Monolítico



- **Definição:** um Programa Monolítico P é um par ordenado

$$P=(I, r)$$

» Onde:

I: representa o conjunto de Instruções Rotuladas o qual é finito;  
r: representa o Rótulo Inicial o qual distingue a instrução rotulada inicial em I.

- Não existem 2 instruções diferentes com o mesmo rótulo;
- Um rótulo referenciado por alguma instrução que não está associado a nenhuma instrução é dito um *Rótulo Final*.

---

---

---

---

---

---

---

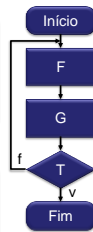
## 2. Conceitos de TC – Programa Monolítico



### Exemplo 1:

```
1: faça F vá_para 2
2: faça G vá_para 3
3: se T então vá_para 4 senão vá_para 1
```

$P_1 = (I_1, 1)$ , onde  $I_1 = \{$   
1: faça F vá\_para 2,  
2: faça G vá\_para 3,  
3: se T então vá\_para 4 senão vá\_para 1  
 $\}$



---

---

---

---

---

---

---

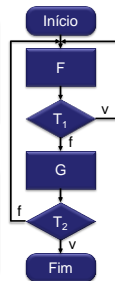
## 2. Conceitos de TC – Programa Monolítico



### Exemplo 2:

```
1: faça F vá_para 2
2: se T1 então vá_para 1 senão vá_para 3
3: faça G vá_para 4
4: se T2 então vá_para 5 senão vá_para 1
```

$P_2 = (I_2, 1)$ , onde  $I_2 = \{$   
1: faça F vá\_para 2,  
2: se T1 então vá\_para 1 senão vá\_para 3,  
3: faça G vá\_para 4,  
4: se T2 então vá\_para 5 senão vá\_para 1  
 $\}$



---

---

---

---

---

---

---

## 2. Conceitos de TC – Programa Iterativo



- Teve sua origem na tentativa de solucionar os problemas decorrentes da dificuldade do entendimento e manutenção de programas monolíticos, com grande liberdade de desvios → *quebra de lógica*.
- Substitui desvios incondicionais por estruturas de controle de ciclos ou repetições.
- Deu origem à Programação Estruturada e inspirou uma série de linguagens de programação como Pascal, C, dentre outras.

---

---

---

---

---

---

---

## 2. Conceitos de TC – Programa Iterativo



- Baseado em 4 mecanismos de composição de programas:
  - » **Sequencial** → composição de 2 programas, resultando em um terceiro. Executa-se o primeiro e em seguida o segundo programa componente.
  - » **Condicional** → composição de 2 programas, resultando em um terceiro. Executa-se somente um dos dois programas dependendo do resultado de um teste.
  - » **Enquanto** → composição de um programa, resultando em um segundo. Executa-se repetidamente o programa componente enquanto o resultado de um teste for *verdadeiro*.
  - » **Até** → análoga à composição enquanto, excetuando-se que a execução do programa componente ocorre enquanto o resultado de um teste for *falso*.

---

---

---

---

---

---

---

## 2. Conceitos de TC – Programa Iterativo



- **Definição:** um Programa Iterativo P é indutivamente definido como segue:
  - a) A operação  $\checkmark$  constitui um programa iterativo;
  - b) Cada identificador de operação constitui um programa iterativo;
  - c) Composição *Sequencial* → se V e W são programas iterativos, então a composição sequencial denotada por: **V;W** resulta em um programa iterativo cujo efeito é a execução de V e, após, a execução de W;

---

---

---

---

---

---

---

## 2. Conceitos de TC – Programa Iterativo



- d) Composição *Condicional* → se V e W são programas iterativos e T é um identificador de teste, então a composição condicional denotada por: **(se T então V senão W)** resulta em um programa iterativo cujo efeito é a execução de V se T é verdadeiro ou W se T é falso;
- e) Composição *Enquanto* → se V é um programa iterativo e se T é um identificador de teste, então a composição enquanto denotada por: **enquanto T faça (V)** resulta em um programa iterativo que testa T e executa V, repetidamente, enquanto o valor do teste for verdadeiro; caso contrário a iteração termina;
- f) Composição *Até* → se V é um programa iterativo e se T é um identificador de teste, então a composição até denotada por: **até T faça (V)** resulta em um programa iterativo que testa T e executa V, repetidamente, enquanto o resultado do teste for falso; caso contrário a iteração termina.

---

---

---

---

---

---

---

## 2. Conceitos de TC – Programa Iterativo



- Exemplo 1:

```
P1 =  
(enquanto T  
  faça (F;G))
```

---

---

---

---

---

---

---

## 2. Conceitos de TC – Programa Iterativo



- Exemplo 2:

```
P2 =  
(se T1  
  então enquanto T2  
    faça (até T3  
      faça (V;W))  
  senão ✓)
```

---

---

---

---

---

---

---

## 2. Conceitos de TC – Programa Recursivo



- É um tipo de programa encontrada na maioria das linguagens de alto nível que admite a definição de sub-rotinas recursivas.
- Recursão é uma forma indutiva de definir programas.
- Sub-rotinas permitem a estruturação hierárquica de programas, possibilitando níveis diferenciados de abstração. Identificadores de sub-rotinas são descritos por: R<sub>1</sub>, R<sub>2</sub>, ...

---

---

---

---

---

---

---

## 2. Conceitos de TC – Programa Recursivo



- **Definição:** uma Expressão de Sub-Rotina (ou simplesmente Expressão)  $E$ , é indutivamente definida como segue:
  - » A operação vazia  $\checkmark$  constitui uma expressão de sub-rotina;
  - » Cada identificador de operação constitui uma expressão de sub-rotina;
  - » Composição *Sequencial*  $\rightarrow$  se  $D_1$  e  $D_2$  são expressões de sub-rotina, então a composição sequencial denotada por:  $D_1; D_2$  resulta em uma expressão de sub-rotina cujo efeito é a execução de  $D_1$  e após, a execução de  $D_2$ ;
  - » Composição *Condicional*  $\rightarrow$  se  $D_1$  e  $D_2$  são expressões de sub-rotina e  $T$  é um identificador de teste, então a composição condicional denotada por:  $(\text{se } T \text{ então } D_1 \text{ senão } D_2)$  resulta em uma expressão de sub-rotina cujo efeito é a execução de  $D_1$  se  $T$  é verdadeiro ou  $D_2$  se  $T$  é falso.

---

---

---

---

---

---

---

## 2. Conceitos de TC – Programa Recursivo



- **Definição:** um Programa Recursivo  $P$  tem a seguinte forma:  
 $P \text{ é } E_0 \text{ onde } R_1 \text{ def } E_1, R_2 \text{ def } E_2, \dots, R_n \text{ def } E_n$ 
  - » Onde suponha  $k \in \{1, 2, \dots, n\}$ :  
 $E_0$ : Expressão Inicial a qual é uma expressão de sub-rotina;  
 $E_k$ : Expressão que define  $R_k$ , ou seja, a expressão que define a sub-rotina identificada por  $R_k$ .

---

---

---

---

---

---

---

## 2. Conceitos de TC – Programa Recursivo



- Exemplo 1:

$P_1 \text{ é } R \text{ onde}$   
 $R \text{ def } (\text{se } T \text{ então } \checkmark \text{ senão } F; G; R)$

---

---

---

---

---

---

---



## 2. Conceitos de TC – Programa Recursivo



### Exemplo 2:

```
P2 é R;S onde  
R def (F;(se T então R senão G; S)),  
S def (se T então ✓ senão F; R)
```

---

---

---

---

---

---

---



Teoria da Computação –  
Aula 02

Ciência da Computação

[clayton.valdo@anhanguera.com](mailto:clayton.valdo@anhanguera.com)



---

---

---

---

---

---

---