INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO

USP SÃO CARLOS

SISTEMAS DE INFORMAÇÃO

INTRODUÇÃO A TEORIA DA COMPUTAÇÃO

DOCUMENTAÇÃO

Simulador Universal de Máquinas De Turing

Geraldo Murilo Carrijo Viana Alves da Silva, Lucas Caetano Procópio, Iara Duarte Mainates.

São Carlos, Maio de 2021

# RESUMO

Este documento tem como objetivo apresentar os raciocínios empregados na construção da solução “Simulador Universal de Máquinas de Turing”, bem como discutir a qualidade da solução proposta e expor a eficiência da solução em termos de espaço e tempo.

**Alunos:**

Geraldo Murilo Carrijo Viana Alves da Silva, nUSP: 11849306

Lucas Caetano Procópio, nUSP: 11831338

Iara Duarte Mainates, nUSP: 11816143.

**Professor:**

João Luís Garcia Rosa.

Departamento de Ciências de Computação - ICMC - USP.

# SUMÁRIO

[**RESUMO**](#_dxuturmin9bm) **2**

[**SUMÁRIO**](#_un90f15cbfct) **2**

[**SOBRE A SOLUÇÃO**](#_7oh4fnv4wcsn) **4**

[Representação da Máquina de Turing](#_m4z716i6vo08) 4

[Representação de Estados](#_fdbcjqgzx6v4) 5

[Leitura de Transições](#_j4jov1ln0n4) 6

[Leitura de Cadeias](#_c4owvwfddr2v) 7

[Validação de Cadeias](#_ci36fc3u311g) 8

[**QUALIDADE DA SOLUÇÃO**](#_olocpc3gvzfs) **10**

[**EFICIÊNCIA DA SOLUÇÃO**](#_e4m4ttsod2c6) **10**

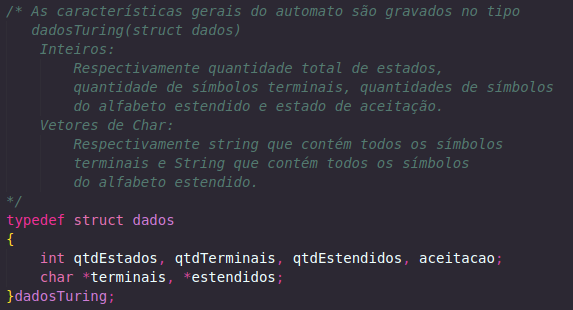
[**REFERÊNCIAS**](#_57moj4ac3qcj) **12**

# SOBRE A SOLUÇÃO

A solução proposta foi pensada utilizando os conceitos das aulas da disciplina Introdução à Teoria da Computação e programada utilizando a linguagem C, pois é a que os integrantes do grupo possuíam mais afinidade.

## Representação da Máquina de Turing

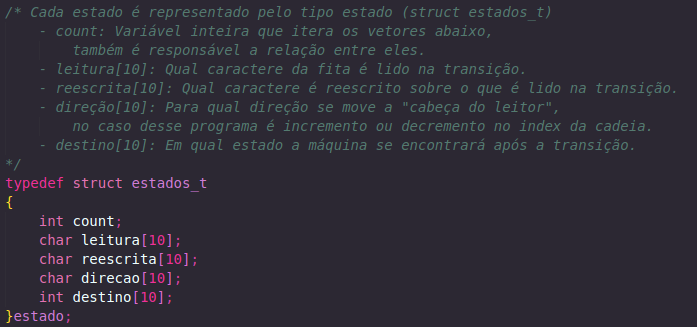
Para representar a Máquina de Turing a ser simulada foi utilizado o recurso de *struct* de dados que a identificam, chamada de “dadosTuring”. Nessa struct são armazenados os dados de quantidade de estados, quantidade de símbolos terminais, quantidade de símbolos estendidos e o estado de aceitação qn - além disso, são armazenados também os símbolos terminais e estendidos em um vetor de char.



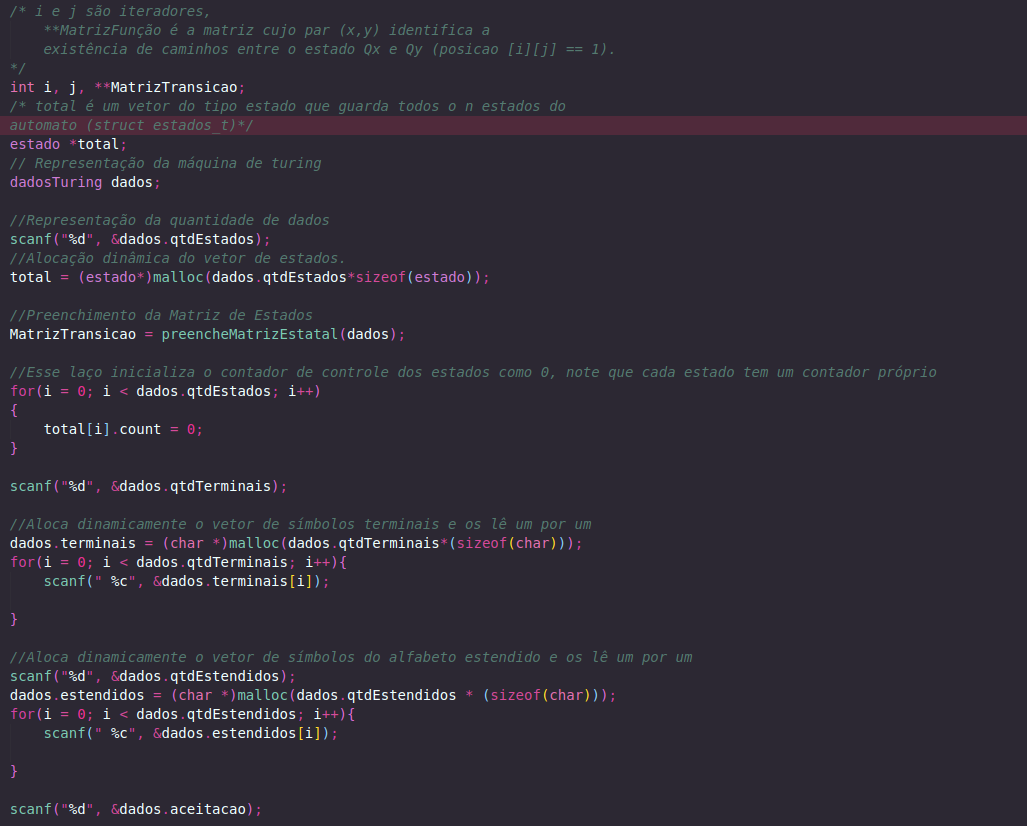
Struct de dados da máquina de Turing

## Representação de Estados

Para a representação de cada estado e suas transições também é utilizada a estrutura de *struct* que contém os dados inerentes à um estado, isto é, sua leitura, sua resposta (reescrita) a direção de movimento e o destino.



Representação dos estados da máquina de Turing

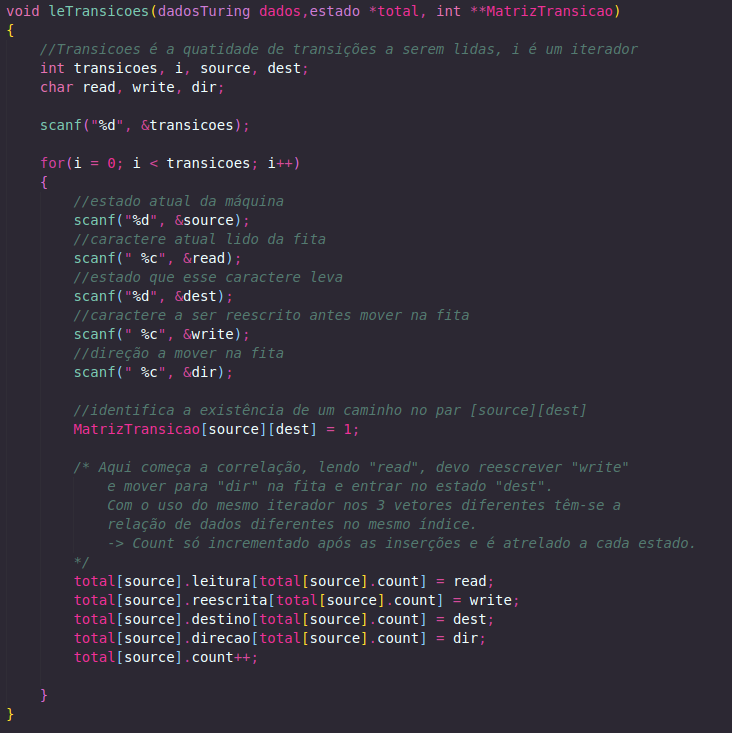
Inicialmente, o programa faz a leitura e a alocação de todos os dados até o estado de aceitação e os direciona para a struct *dadosTuring dados*

Leitura de dados da máquina de Turing

## Leitura de Transições

A leitura de cada transição é feita através chamada a função *leTransicoes*, chamadas na função main. Essa leitura é armazenada na matriz de transição como o número 1 quando a transição é válida, e seus dados são guardados na struct de estados *total*

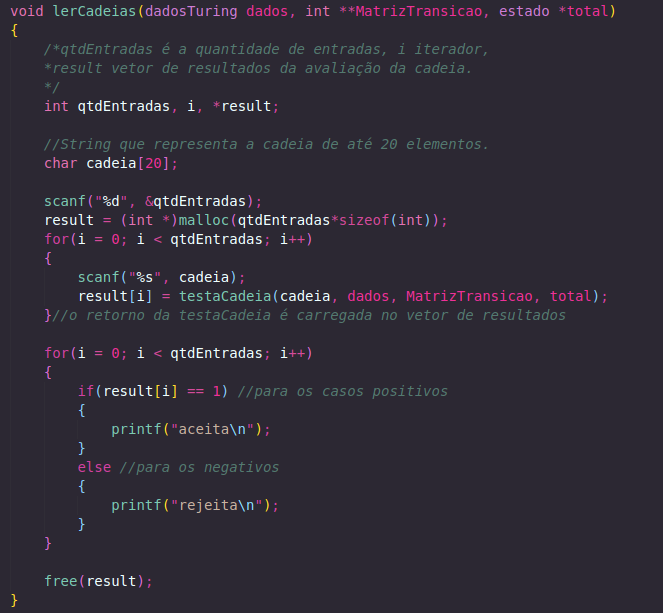




Leitura de Transições

## Leitura de Cadeias

Ao finalizar essas leituras, o programa tem todos os dados necessários para simular uma Máquina de Turing. Para isso, temos a chamada da função de *lerCadeias* para ler as sequência inseridas pelo usuário.



Leitura de Cadeias e Validação via Máquina de Turing

Dentro dessa função, o programa faz a leitura da quantidade de entradas e as lê individualmente. Ao fazer a leitura, o programa redireciona a cadeia para a função *testaCadeia* que irá fazer a validação de acordo com os princípios da Máquina de Turing e retorna 1 em caso de sucesso (cadeia válida) ou 0 em caso de cadeias inválidas. Após a leitura e validação de todas as cadeias, o programa irá imprimir na tela o resultado como “aceita” em casos positivos e “rejeita” em casos negativos.

## Validação de Cadeias

Já a função *testaCadeia* irá receber a cadeia a ser avaliada, a máquina de Turing utilizada, a matriz de transições e os estados da máquina correspondente - e para validar a cadeia irá utilizar de um loop que é executado até que a cadeia atinja o estado final ou até ou até que não seja encontrado, no vocabulário do estado, determinado símbolo terminal.

Dentro desse loop, o programa vai de 0 até a quantidade máxima de estados e para cada estado o programa verifica se o atual símbolo está contido na leitura de tal estado. Caso o símbolo atual esteja nesse estado, o programa reescreve na “fita” o valor de reescrita e verifica sua orientação de direção (R para direita e L para esquerda).

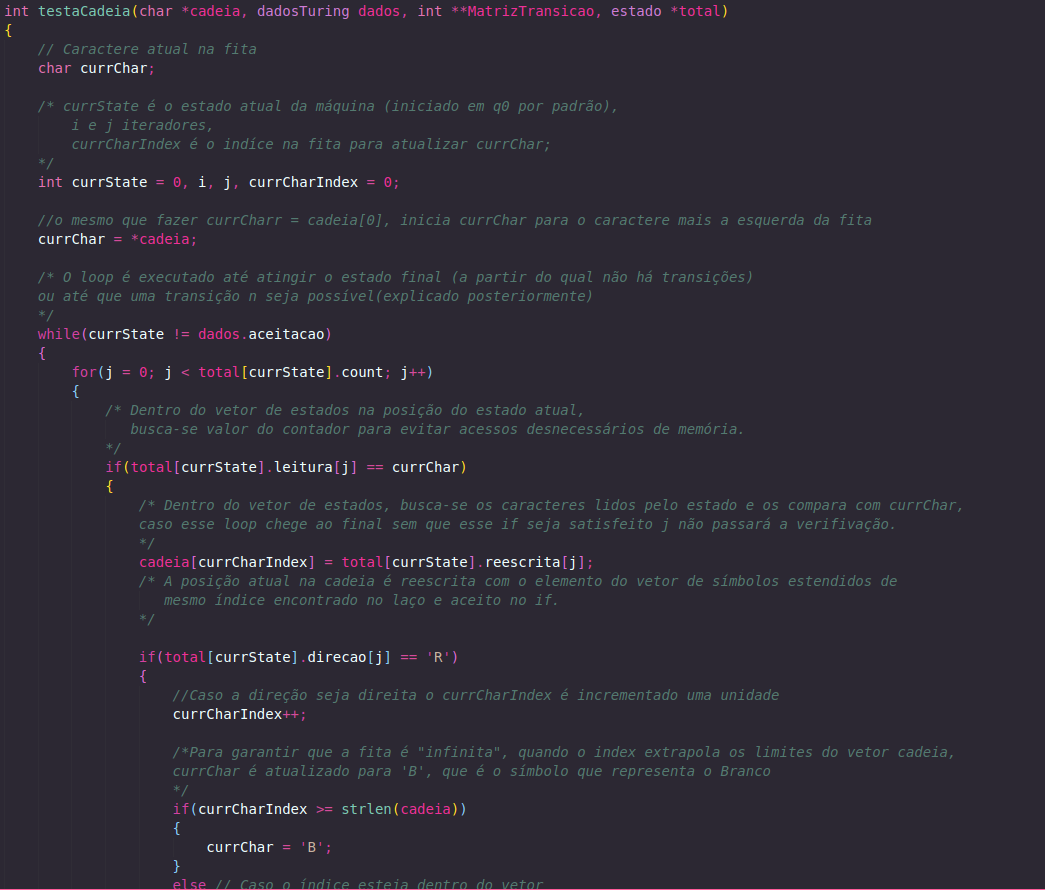
Na situação de direcionamento a direita, o index de símbolos é incrementado e o programa passará a olhar para o próximo símbolo - e para garantir que a fita será infinita, quando passamos do index limite do vetor da cadeia analisada, preenchemos o espaço atual com B, significando um branco.

Na situação de direcionamento para esquerda, o index de símbolos é decrementado e o programa irá olhar para o símbolo anterior - e para garantir que a fita será infinita, quando passamos para um index negativo, preenchemos o espaço atual com B, significando um branco.

Com a verificação, o estado atual é modificado para o índice j dentro do vetor de estados (controlado pelo loop for) e j é atualizado com o valor 200 para casos em que o if seja re-verificado - e interrompemos o loop.

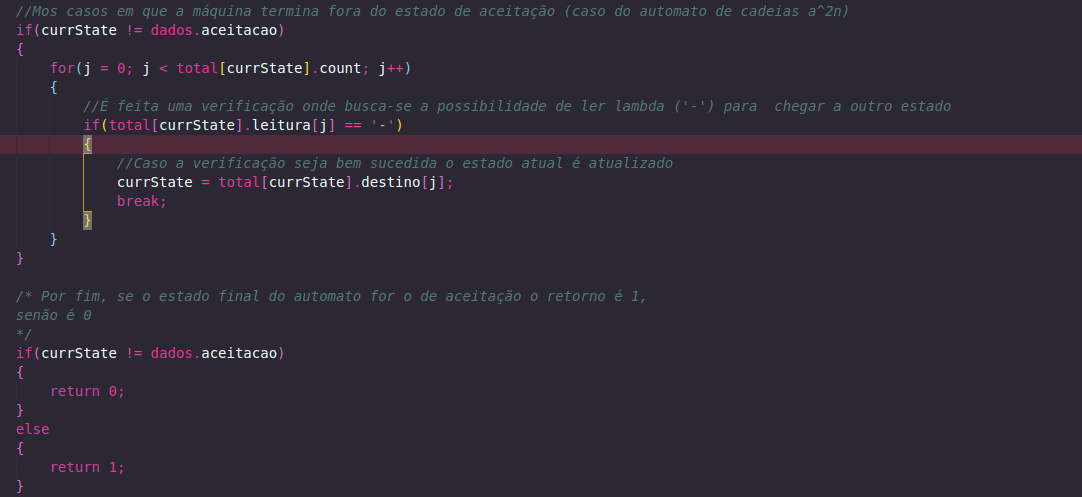
Para o tratamento do caso em que a máquina não chega ao estado de aceitação, o programa irá verificar se há a possibilidade de ler lambda (cadeia vazia) para chegar ao estado final. Caso a verificação funcione, o estado é atualizado.

Ao final, verificamos se o estado atual corresponde ao estado de aceitação - em caso positivo, a função retornará 1, e em caso negativo irá retornar 0.



Função testaCadeia, parte 1.

Função testaCadeia, parte 2.



Função testaCadeia, parte 3.

# QUALIDADE DA SOLUÇÃO

Em termos de qualidade, a solução pode ser classificada como boa pois o código faz uso de alocações e loops dinâmicos, não rodando mais do que o necessário e utilizando ciclos de acordo com o inserido pelo usuário - e não mais que isso. Além disso, a estrutura de uma Máquina de Turing têm no máximo tamanho de 36 bytes (16 bytes de inteiros relacionados aos controladores de quantidades e 20 bytes de char relacionados aos vetores de char), enquanto a estrutura de estados ocupa até 44 bytes (40 dos vetores de inteiros e 4 de inteiros).

Já para a avaliação de qualidade de código, pode-se avaliar como uma boa escrita já que o código faz separação entre funções, modularizando os códigos e as variáveis podem ser auto-explicadas.

# EFICIÊNCIA DA SOLUÇÃO

Como visto anteriormente, em termos de memória, a representação das Máquinas de Turing e seus estados ocupam um espaço reduzido e a representação dos estados em vetor de structs, armazenados sequencialmente, reduz também a quantidade de acessos - que são feitos “sequencialmente”.

Quanto ao tempo de execução, o algoritmo empregado para o cálculo do resultado da Máquina de Turing pode ser traduzido para um laço while simples que possui um for que caminha por todos os estados, assim sendo limitado apenas pelo tamanho da quantidade de estados, possuindo portanto crescimento quadrático.

# 

# REFERÊNCIAS

* **Normas ABNT para apresentação de trabalhos científicos.** Wikimedia. Disponível em: <[http://pt.wikipedia.org/wiki/Normas\_ABNT\_para\_apresentação\_de\_trabalhos\_científicos](http://en.wikipedia.org/wiki/Normas_ABNT_para_apresenta)>. Acesso em 25 de maio de 2021.
* **Rosa, J. L. G. Linguagens Formais e Autômatos**. Editora LTC, 2010.