Linguagens Formais e Autômatos

IMPLEMENTAÇÃO DE MÁQUINAS DE MEALY:
COMPOSIÇÃO DE FRACTAIS POR MEIO DE INPUTS CARTESIANOS PROCESSADOS POR UM AUTÔMATO FINITO

Alunos: Luciano Kohler e Luiza Mannes
Professor: Karina Girardi Roggia

Joinville - 2025

SUMÁRIO

1 INTRODUÇÃO....3
1.1. Introdução a Uma Máquina de Mealy....3
1.2. Objetivos do Relatório....4
2 METODOLOGIA....4
2.1. Softwares Utilizados....4
2.2. Formatação das Entradas....5
2.3. Estrutura do Projeto....6
2.4. Lógica Central do Código....7
3 RESULTADOS....7
3.1. Expressão (1+2+4)*3(2+4)*3(1+2+3+4)*....8
3.2. Expressão (12+14+32+34+21+23+41+43)*....9
3.3. Expressão (1+3)*(2+4)*....10
4 GUIA PARA USO DO SOFTWARE....11
5 REFERÊNCIAS BIBLIOGRÁFICAS....12

1 INTRODUÇÃO

1.1. Introdução a Uma Máquina de Mealy

A computação agregou valor inestimável ao cotidiano humano. Desde o advento do primeiro computador até os atuais dias, o software e o hardware se tornaram partes vitais do funcionamento da sociedade de forma que é praticamente impossível presenciar um cenário ou local livre de tecnologias. Um dos frutos contemporâneos da enorme árvore da ciência da computação, são os *autômatos finitos*, interpretações lógicas de **comportamento** que uma máquina deve seguir à partir de um **input recebido**.

Para a melhor exemplificação, conceba um autômato de exemplo. Na imagem abaixo, há a representação visual de um autômato:

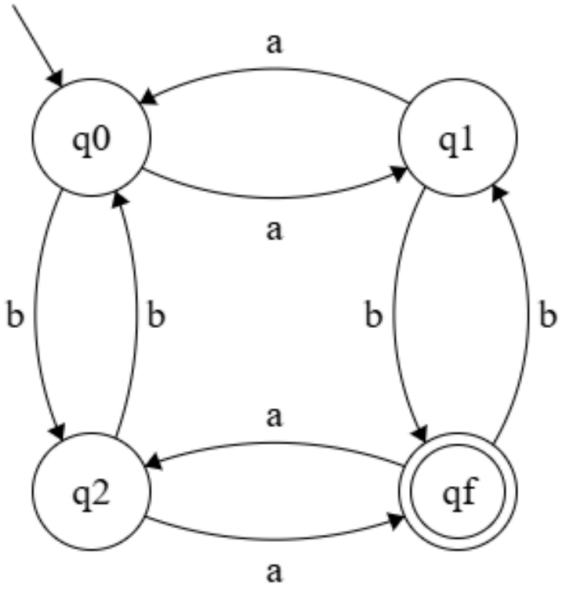


Figura 1 - Representação visual de um autômato finito A = <Q, q0,Σ, δ, qf>

Um autômato finito é definido por uma série de elementos:

- **Q:** Uma quantidade finita de **estados** representados pelas bolinhas
- **q0:** Estado inicial (representados pela seta isolada) que indicam onde as transições se iniciam
- ***Σ:*** Um alfabeto de entrada, que engloba todas os possíveis símbolos que uma palavra possa ter, gerando palavras que podem ser concebidas pelo autômato
- ***δ:*** Transições, representadas pelas setas que "movem" o input pelos estados, sempre consumindo a letra especificada no processo
- **qf:** Estado final (representados pelos estados duplamente circulados), que indica em que caso a palavra será "aceita"

O autômato da figura 1 representa a linguagem de *todas as palavras compostas por uma quantidade ímpar de "a" e "b",* aceitando palavras que seguem a regra (pois as transições irão terminar no estado final), e rejeitando os casos contrários (pois as transições terminaram em estados não finais).

Uma ramificação dos autômatos é dada pela *máquina de Mealy*, um autômato que, similar ao já visto anteriormente, possui todas as características de um autômato comum, porém suas transições, além de consumirem caracteres da palavra de entrada, elas também enviam caracteres para a palavra de saída (oriundas de um alfabeto de saída), assim, o autômato pode retornar mais dados do que somente uma mera aceitação de palavras. Veja um exemplo de máquina de Mealy abaixo:

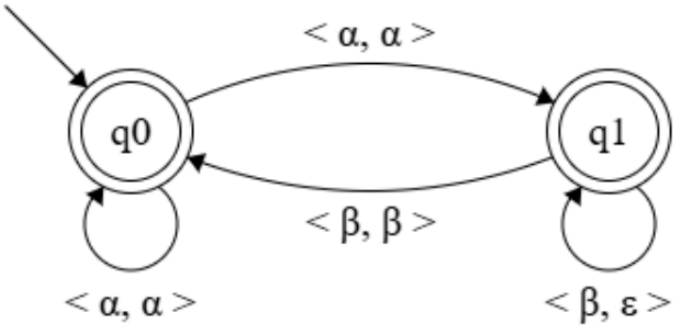


Figura 2 - Representação visual de uma máquina de Mealy que trunca espaços a mais e uma frase

Para esse exemplo, imagine que a letra α significa um símbolo qualquer, e β, um espaço em branco, o autômato acima, então, reescreve a palavra de entrada, porém removendo espaços desnecessário, então, por exemplo, “Máquina de Mealy”, passando pela máquina, se tornará “Máquina de Mealy”.

1.2. Objetivos do Relatório

Com a definição da máquina de Mealy em mente, será feito, por meio deste documento, o relatório do experimento envolvendo as máquinas em prática com o uso de setorizações cartesianas que, à partir de uma expressão regular (transformada em um Mealy) e uma palavra *w,* composta de subpalavras *l*, será “desenhado” uma imagem fractal visualmente por meio de um arquivo *ppm* preto e branco, onde pontos pretos são palavras rejeitadas pela linguagem, e os brancos, são palavras válidas, tudo implementado por um projeto feito em **C++** que recebe arquivos **.txt** como input.

O projeto está disponível via um *repositório do Github* pelo link https://github.com/LucianoKohler/Trabalho_LFA_MaquinaDeMealy.

2 METODOLOGIA

2.1. Softwares Utilizados

- Compilador **g++ V.14.2** (também funcionando para a V.15.1)
- Um arquivo **.ppm** para retornar visualmente o resultado da execução do programa
- Arquivos **.txt** para realizar o input de dados para o código

2.2. Formatação das Entradas

Para o bom funcionamento do software, é necessário que haja duas entradas padronizadas: uma **palavra** de entrada, e uma **máquina de Mealy** válida para o processamento:

- Para uma **palavra** ser válida, é necessário que:
  - Seu nome siga o padrão “wx.txt”, sendo x a quantidade de elementos que existem em uma linha
  - Seu conteúdo seja padronizadamente representado como a segmentação de um plano cartesiano em relação aos seus quadrantes, onde os números indicam a “inclinação” do setor para cada quadrante, “.” é a divisão de setor para setor, e “N” indica uma nova linha
  - Exemplo da palavra w8:
    - 222.221.212.211.122.121.112.111.N223.224.213.214.123.124.113.114.N232.231.242.241.132.131.142.141.N233.234.243.244.133.134.143.144.N322.321.312.311.422.421.412.411.N323.324.313.314.423.424.413.414.N332.331.342.341.432.431.442.441.N333.334.343.344.433.434.443.444.N
  - A palavra acima é a equivalência textual da setorização visual do plano à seguir:

222	221	212	211	122	121	112	111
223	224	213	214	123	124	113	114
232	231	242	241	132	131	142	141
233	234	243	244	133	134	143	144
322	321	312	311	422	421	412	411
323	324	313	314	423	424	413	414
332	331	342	341	432	431	442	441
333	334	343	344	433	434	443	444

Figura 3 - Representação visual da segmentação do plano cartesiano

  nota
- Para uma **máquina de Mealy** ser válida, é necessário que ela siga um padrão árduo:
  - O nome do arquivo pode ser qualquer coisa, desde que no formato **.txt**, é recomendado que siga o padrão “MMx.txt” sendo x o número da máquina (à escolha do usuário), assim seguindo padrão com as outras máquinas
  - A entrada da máquina siga o padrão:

```
q0 q1 (...) qn
q0
qf1 (...) qfj
1 2 3 4 . N
0 1 \n
q1 x1 r1 y1
q2 x2 r2 y2
(...)
qk xk rk yk
```

- Primeira linha: Contém os n estados da máquina
- *Segunda linha:* Contém o estado inicial da máquina
- *Terceira linha:* Contém todos os estados finais da máquina
- *Quarta linha:* Contém o alfabeto de entrada
- *Quinta linha:* Contém o alfabeto de saída
- *Todas as linhas subsequentes:* Irão representar uma transição
    - *qk*: Estado inicial da transição
    - *xk*: Símbolo de entrada (consumido)
    - *rk*: Estado atual (após o consumo)
    - *yk*: Símbolo de saída (escrito)

Por fim, se os dois arquivos de entrada estiverem devidamente formatados e presentes em suas respectivas pastas, basta passar os caminhos relativos dos mesmos para o programa na hora de executá-lo (especificado no tópico 4. GUIA PARA USO DO SOFTWARE)

2.3. Estrutura do Projeto

Existem quatro elementos chave no repositório:

1. A pasta **palavras**, responsável por comportar os arquivos de palavras de entrada disponibilizados pela professora orientadora. Cada arquivo segue o formato de nomeamento como sendo **wx.txt**, sendo **x** a variável que indica a quantidade de subintervalos do plano cartesiano em uma linha, w8 indica, por exemplo, que o plano foi segmentado em uma grade 8x8, w16 foi segmentado em uma grade 16x16, e assim por diante..
    - Atualmente, há quatro palavras disponíveis no projeto: **w8.txt**, **w16.txt**, **w256.txt** e **512.txt**.
    - Palavras maiores retornaram saídas mais precisas e de melhor qualidade para perceber o comportamento fractal da expressão regular
    - Nota: É possível inserir mais arquivos de palavras se elas seguirem o mesmo formato de entrada e nomeamento do projeto

2. A pasta **maquinas**, responsável por comportar as máquinas de Mealy traduzidas manualmente pelos integrantes da equipe, cada máquina tem uma expressão regular representativa que está visível nos comentários iniciais do arquivo **main.cpp**. Existem três arquivos na pasta:
    - **MM1.txt**, representando a expressão regular (1+2+4)*3(2+4)*3(1+2+3+4)*
    - **MM2.txt**, representando a expressão regular (12+14+32+34+21+23+41+43)*
    - **MM3.txt**, representando a expressão regular (1+3)*(2+4)*
    - Nota: É possível inserir mais arquivos de autômatos se eles seguirem o mesmo formato de entrada, o nomeamento padronizado, porém, é dispensável

3. A pasta **out**, responsável por armazenar todos os arquivos **.ppm** gerados pelo código principal, cada arquivo segue o padrão **wxSaida.ppm**, sendo wx a palavra usada para processamento da máquina

4. Um arquivo **main.cpp**, comportando o código responsável pelo processamento da máquina e da palavra

*Nota: Mesmo que hajam outros arquivos e pastas no repositório, os mesmos são dispensáveis e servem apenas para apresentar formalmente o projeto (como este relatório e o README.md)

2.4. Lógica Central do Código

O código presente na **main.cpp** possui 5 passos principais separados por comentários nomeados:

1. **Recepção do autômato à partir da expressão regular dada**: Onde é feito a abertura do arquivo **MMx.txt** para sua leitura posterior
2. **Tradução do texto para linguagem C++**: Onde o arquivo aberto é destrinchado e suas variáveis são separadas e alocadas em arrays dinâmicos
3. **Leitura das transições**: Um passo separado do passo 2 pela sua maior complexidade, aqui, todas as k linhas de transições são recebidas e colocadas em um **mapa** onde cada **chave** é um estado, e cada **valor** é um vetor que possui as n transições que envolvem o estado específico
4. **Leitura da palavra de entrada e formatação do arquivo de saída**: Onde o arquivo **wx.txt** é recebido e o arquivo **wxSaida.ppm** é criado e seu cabeçalho (necessário para que o arquivo funcione) é criado
5. **Processamento da palavra**: Por fim, a palavra é processada elemento por elemento, e o arquivo de saída é paralelamente escrito pelos símbolos do alfabeto de saída da máquina

3 RESULTADOS

Após a execução do programa realizada, podemos finalmente visualizar os fractais gerados por certas expressões regulares ao abrir a saída .ppm do programa.

Nos tópicos à seguir, será apresentado:

- A expressão regular analisada
- A expressão descrita em linguagem natural
- A representação visual da máquina de Mealy que descreve a expressão
    - Algo importante a se destacar é que, para qualquer estado que não possui as transições do tipo $< ., 1 >$ ou $< ., 0 >$, a saída padrão para os estados que lêem "." prematuramente é 0, mas em prol da legibilidade dos autômatos, estas arestas foram omitidas.
    - Para transições de mesma saída, mas com entradas diferentes, foi utilizado uma notação simplificada para ocupar menos espaço no diagrama:
        - $< 1, \varepsilon > < 2, \varepsilon > < 3, \varepsilon > < 4, \varepsilon > = < 1/2/3/4, \varepsilon >$
- Por fim, o output da máquina de Mealy em cima de palavras de tamanho variado

*3.1. Expressão (1+2+4)*3(2+4)*3(1+2+3+4)**

Traduzindo a expressão para linguagem natural, resultaria em algo como:
"Toda palavra onde não há ocorrências de '1' após a primeira ocorrência do número 3"

Com isso em mente, temos a máquina de Mealy:

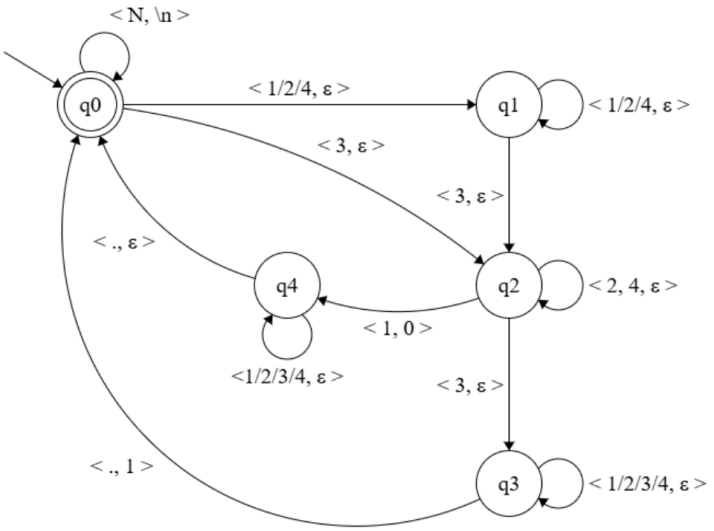
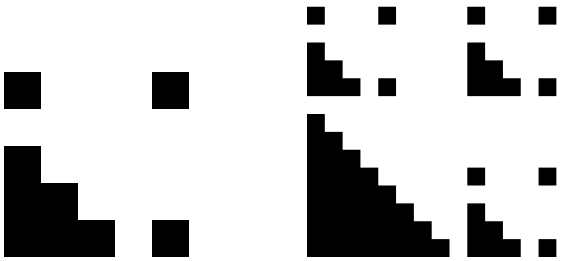


*Figura 4 - Máquina de Mealy correspondente à expressão (1+2+4)*3(2+4)*3(1+2+3+4)**

E a saída usando a w8, w16 e w512:



Figuras 5 e 6 - Saídas da w8 e w16, respectivamente

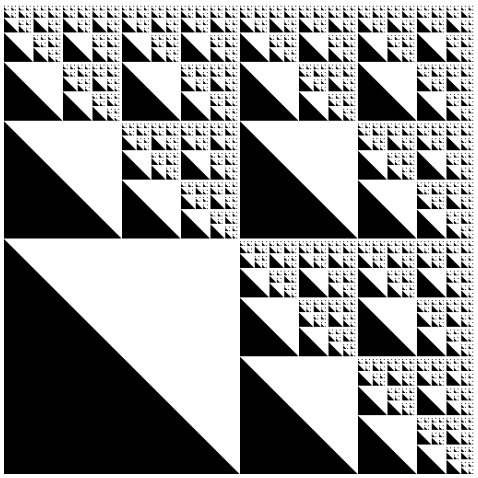


Figura 7 - Saída da w512

*3.2. Expressão (12+14+32+34+21+23+41+43)**

Traduzindo a expressão para linguagem natural, resultaria em algo como:
"Toda palavra composta por zero ou mais ocorrências dos pares 12, 14, 32, 34, 21, 23, 41, 43 "

Essa expressão possui uma característica interessante: **Todas as palavras aceitas** (que printam 1) **possuem tamanho par**, logo, para as w8 e w512, a saída será vazia, pois para representar uma grade 8x8 (no caso da w8), são necessários 3 dígitos para cada setor/bit da imagem, um valor ímpar, portanto, para verificar o fractal em um caso maior, será utilizado a **w256**, gerada fora das especificações do trabalho.

Com isso em mente, temos a máquina de Mealy:

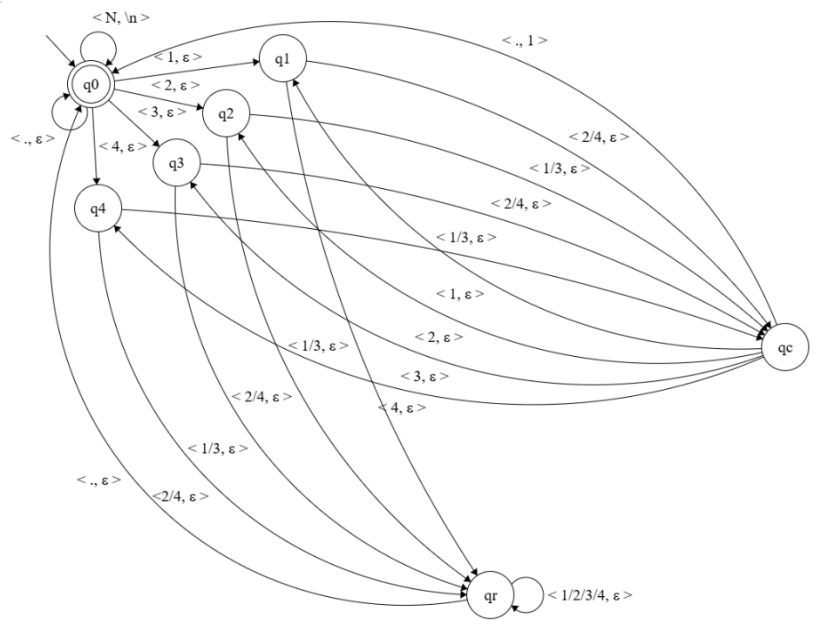
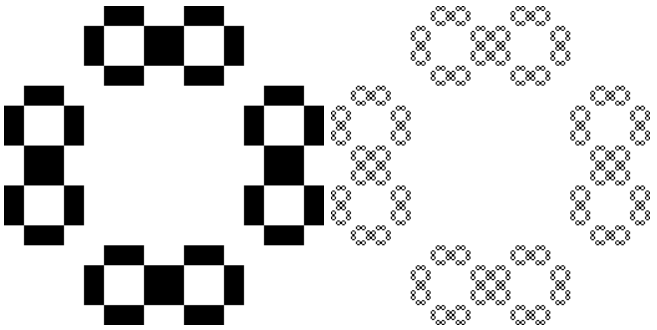


*Figura 8 - Máquina de Mealy que correspondente à expressão (12+14+32+34+21+23+41+43)**

E a saída usando a w8 e w256:



Figuras 9 e 10 - Saídas da w16 e w256, respectivamente

3.3. Expressão (1+3)(2+4)**

Traduzindo a expressão para linguagem natural, resultaria em algo como:
"Toda palavra formada por 0 ou mais ocorrências de '1' e '3' seguidas de zero ou mais ocorrências de '2' e '4'"

Com isso em mente, temos a máquina de Mealy:

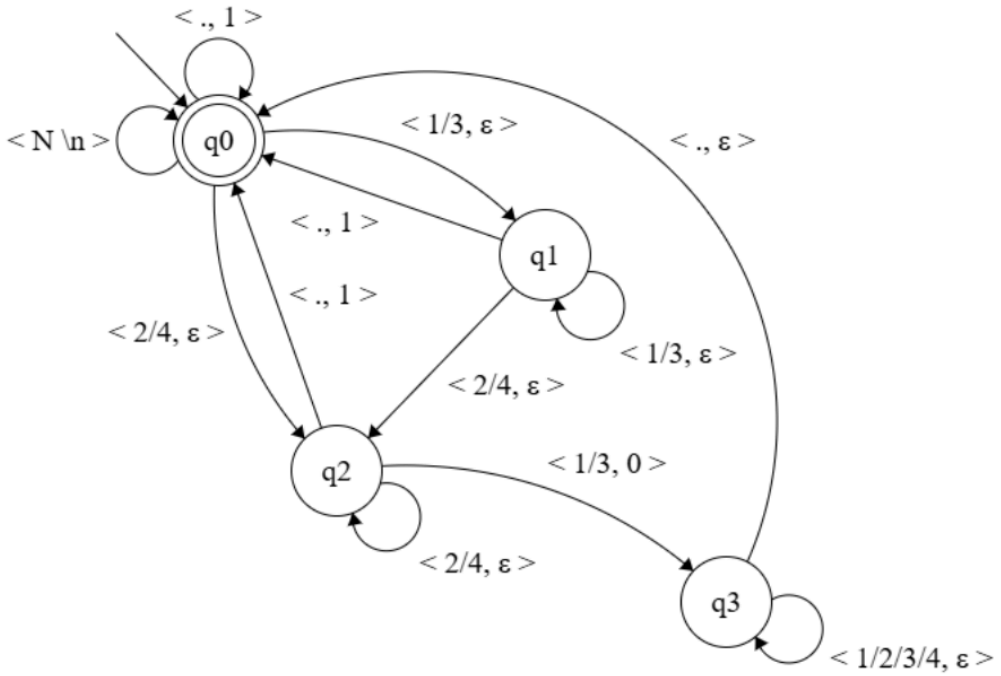
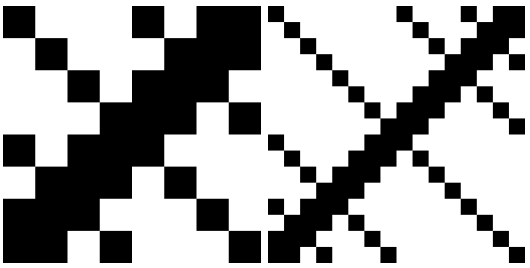


Figura 11 - Máquina de Mealy que correspondente à expressão (1+3)(2+4)**

E a saída usando a w8, w16 e w512:



Figuras 12 e 13 - Saídas da w8 e w16, respectivamente

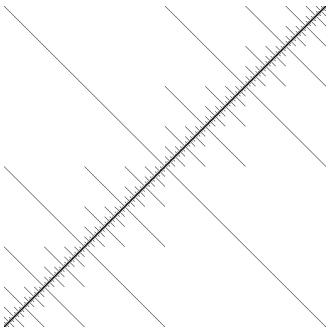


Figura 14 - Saída da w512

4 GUIA PARA USO DO SOFTWARE

Com as especificações detalhadas e os dados apresentados, é possível, agora, que o leitor interaja com o software e teste as suas próprias máquinas em busca de um novo fractal, para tal, é necessário se atentar aos seguintes passos:

- Clone ou baixe o repositório do Github contendo o projeto (https://github.com/LucianoKohler/Trabalho_LFA_MaquinaDeMealy)

- Em um terminal, no ambiente Ubuntu, entre no diretório contendo o arquivo **main.cpp** e utilize seu compilador de C++ (de preferência, o G++ de versão já especificada no tópico 2.1. Softwares Utilizados)
    - Para compilar o código rode o comando:
        - `g++ main.cpp -o main.out`

- Com o código compilado, insira dentro do projeto(se for o caso), novas máquinas e palavras, se baseando no tópico 2.2. Formatação das Entradas para formatar corretamente os arquivos de entrada
    - Nota: É totalmente aceitável colocar os arquivos de entrada na pasta raíz do projeto, desde que o caminho para os arquivos, na hora da execução do código, também remeta à pasta raíz

- Com o setup pronto, abra novamente o terminal e execute o arquivo .out gerado na hora da compilação, passando como parâmetros, respectivamente, a **máquina escolhida** e a **palavra escolhida**

- Se o código executou corretamente, uma mensagem de retorno positiva aparecerá no terminal, indicando então, que o arquivo **.ppm** foi devidamente gerado na pasta **out**.

Veja alguns exemplos usados pelos autores para gerar os fractais vistas anteriormentes:

- Exemplo de execução usando a máquina 3 com a palavra de tamanho 512:
    - `./main.out “maquinas/MM3.txt” “palavras/w512.txt"`

- Exemplo de execução utilizando uma máquina personalizada (denominada MMpersonalizada.txt) com a palavra de tamanho 256, onde ambos os arquivos estão na pasta raíz do projeto:
    - `./main.out “MMpersonalizada.txt” “w256.txt”`

*Nota: O uso das aspas duplas ao passar os parâmetros é opcional, porém facilita na leitura do usuário

5 REFERÊNCIAS BIBLIOGRÁFICAS

W3Schools. ***C++ Maps***. Disponível em: https://www.w3schools.com/cpp/cpp_maps.asp. Acesso em: 20 out. 2025.

Wallace, Evan. ***Finite State Machine Designer***. Disponível em: https://madebyevan.com/fsm/. Acesso em: 19 out. 2025

MENEZES, P. F. B. Linguagens Formais e Autômatos. Série Livros Didáticos n° 3. 4ª. edição. Ed. Sagra Luzzato, 2002.