**UNIVERSIDADE DO ESTADO DO PARÁ**

**LINGUAGENS FORMAIS**

**LINGUAGEM BES**

**RUAN FELIPE SILVA NEGRÃO, LEOCENDINO NEGRÃO CARDOSO , RONALDO COSTA E SILVA FILHO, ANA CAROLINE DA SILVA ARAÚJO, HUGO TAIYO KIMURA**

**CAMPUS XX - CASTANHAL- PARÁ- BRASIL**

**Endereço**: Rua Pedro Porpino, 1181 - PA 320 - Salgadinho   **CEP**: 68745-000 **Castanhal-PA**

**Centro de Ciências Naturais e Tecnologia – Universidade do Estado do Pará**

**[ana.cdsaraujo@aluno.uepa.br,](mailto:ana.cdsaraujo@aluno.uepa.br,) [leocendino.ncardoso@aluno.uepa.br,](mailto:leocendino.ncardoso@aluno.uepa.br,)**

**[ruan.fsnegrao@aluno.uepa.br,](mailto:ruan.fsnegrao@aluno.uepa.br,)**

**Resumo.** Trabalho apresentado no curso de engenharia de software da universidade do estado do Pará com fins didáticos e orientado pelo professor  **Anderson Costa.**

**1.Informações Gerais**

A linguagem BES foi desenvolvida no período de 22/01/2021 a 07/02/2021 no horário **Diurno e noturno** pela equipe formada por (**Ruan , Leo, Hugo, Ronaldo e Ana**) alunos do curso de engenharia de software. Sua finalidade é facilitar a escrita de códigos de programação.

**2.BES: Uma nova linguagem**

Na linguagem BES temos como alfabeto os algarismos de “a” a “z” (da língua portuguesa), os números indo-arábicos de 0 a 9 e diversos símbolos matemáticos, por exemplo, +,-,/,.,=, entre outros que serão detalhados no decorrer do texto.

A linguagem BES foi desenvolvida com o intuito de otimizar a escrita das funções de programações, tendo em vista que sua principal característica é a substituição de parte das palavras reservadas extensas por números. Entretanto, por conta dessas substituições a linguagem é voltada de maneira mais ampla para aqueles que já tem algum conhecimento em programação.

São outras características marcantes da linguagem, o uso de palavras simples e intuitivas para o usuário brasileiro, tornando a escrita de códigos mais fáceis, já que a mesma exclui o uso de palavras que não são do idioma nativo. E a gramática simples com instruções diretas, as quais o usuário não precisa dedicar muito tempo escrevendo.

Por fim, a linguagem BES tenta diminuir a utilização de muitos caracteres em seus códigos, tornando as preocupações com a escrita extensão menores, o que possibilitaria o manuseio de grandes códigos mais prático para o programador.

**3.Alfabeto da Linguagem Bes**

Alfabeto é um conjunto de símbolos que isoladamente ou seguindo um conjunto de regras formam palavras que serão reconhecidas pela linguagem a qual se aplica. Na Linguagem BES, os símbolos que formam o alfabeto são:

Σ={a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, x, y, z, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, : , ; , ! , ‘,^, ~, =,<,>,/,+,-, . ,%, \* , ü , ´ , [ , ] , ª, {, }, | , &, , ,}

Os quais inseridos nas regras gramaticais da linguagem serão reconhecidos como palavras ou instruções.

**4.Instruções e palavras reservadas**

As instruções são palavras de ordem previamente definidas em um arquivo de texto chamado código-fonte, as seguintes palavras de ordens definidas na linguagem BES, reconheceram comandos que descreveremos:

Principais estruturas e comandos da linguagem Bes:

0 = limpa a tela.

1 = semelhante ao printf do c, ela imprime informações na tela.

2 = estrutura semelhante ao scanf do c, ela lê informações na tela.

3 = substitui o for.

4 = substitui o if.

5 = substitui o else.

6 = substitui o while.

7 = substitui o do.

8 = substitui o swicth.

9 = substitui o exit.

As palavras reservadas por sua vez, são palavras que não podem ser usadas como identificadoras de variáveis, pois são palavras-chaves para a construção de execução de comandos, como por exemplo, # que indica começo e fim de comentário.

caso = substitui o case.

! = substitui o Break.

, = usado para separar os valores de variáveis e parâmetros nas funções, e também para contar as casas decimais de valores reais e grandes

falha = substitui o default.

vamos = substitui o comando goto.

% = pode ser usado como operador para calcular o resto da divisão ou para capturar informações fornecidas pelo usuário.

´ = funciona como ‘aspas’ para mostrar qual informação vai ser impressa na tela.

: = substitui os parêntese das estruturas comuns.

retorno = função retorno.

estrutura = substitui a estrutura struct.

# = faz comentários .

programa= quando o código for salvo será salvo no nome descrito no cabeçalho.

define = define uma valor de variável que não mudara.

vazio = void.

‘\*’= faz com que o compilador reconheça a variável dentro de uma estrutura para imprimir textos.

‘ = defini o dado que será alocado na variável

principal = define a função main.

**Operadores de condição :** < (menor que), >(maior que) , =:(igual), =>(igual ou maior que), =<(igual ou menor que), =/(diferente);

**Operadores lógicos:** || (ou), &(e), !(não);

**Operadores aritméticos:** =(atribuição), / (divisão),+ (soma), -(subtração), . (multiplicação), %(resto da divisão),ü (potenciação), ª (raiz quadrada);

**Tipos de variáveis:** inteiro(int), real, logico(log), grande, caracter(car);

**5.Gramática**

Gramática é o conjunto de regras de produção de cadeias, ou seja, um objeto que permite especificar e reconhecer uma linguagem. A gramática é responsável por definir como o código será escrito.

Na linguagem BES as principais regras da gramática são:

**Estrutura para imprimir na tela:**

S→ 1A

A→ :´texto´;| :´texto \*variável´;

**Estrutura para ler dados na tela:**

S→ 2A

A→ :%variável;

**Estrutura for:**

S→ 3A

A→ :C: {corpo de instruções}

C→ i= ‘0’; i operador de condição n ; i ++

**Estrutura if-else:**

S→4A

A→ :C:{ bloco de instruções}E

C→ V operador V

E→ £ | 7{bloco de instruções} | 5:4A

**Estrutura switch-case:**

S→8A

A→ :V:{E}

E→ caso ‘V’ bloco do programa ! E | caso ‘V’ bloco do programa ! D

D→ falha bloco de instrução !

**Estrutura while:**

S→6A

A→ :C:{bloco de instruções}

C→ i operador de condição n

**Estrutura do:**

S→ 7A

A→{bloco de instrução}E

E→ 6: condição;

**6.Exemplos de código com base na gramática apresentada:**

programa: Exemplo01;

car a ;

int idade=’20’;

int principal::{

1:´Escreva seu nome: ´;

2:%a;

1:´Olá, mundo! Eu sou ‘\*’a ´;

1:´Eu tenho ‘\*’idade anos´;

# esse código irá imprimir “Olá, mundo! <nome da pessoa>” e em seguida “Eu tenho 20 anos” #

retorno ‘0’;

}

-------------------------------------------------------------------------

programa: Exemplo02; # identifica valores maiores que 10#

int x;

int principal::{

1:´Informe um valor: ´;

2:%;

4: x =>10:

{ 1:´É um valor maior que 10´; }

5:{ 1:´Não é um valor maior que 10´ }

}

**7.Automâtos**

Os autônomos podem ser autômatos finitos, autômatos de pilha e Máquinas de Turing, eles são reconhecedores que identificam se uma frase pertence ou não a uma linguagem. Na linguagem BES podemos exemplificar os seguintes autômatos:

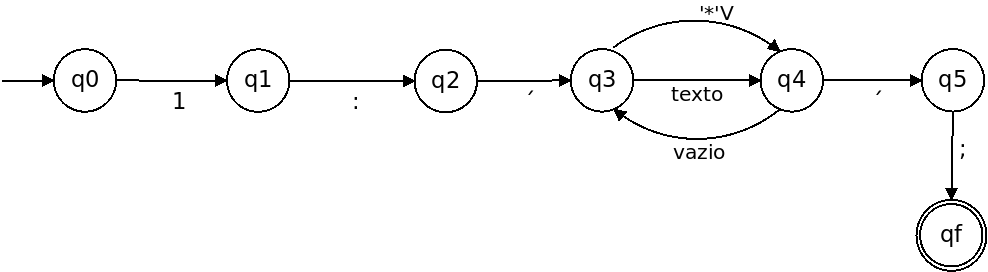
1 Imprime na tela.

Gramática:

S→ 1A

A→ :´texto´;| :´texto ‘\*’variável´;

Autômato:



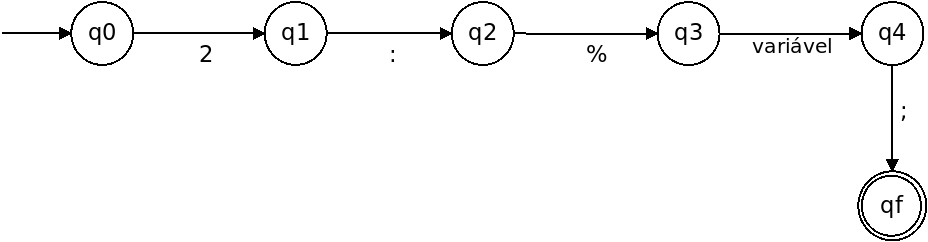
2 Lê informações que o usuário inseriu.

Gramática:

S→ 2A

A→ :%variável;

Autômato:



3 Repete um comando através de comandos determinados

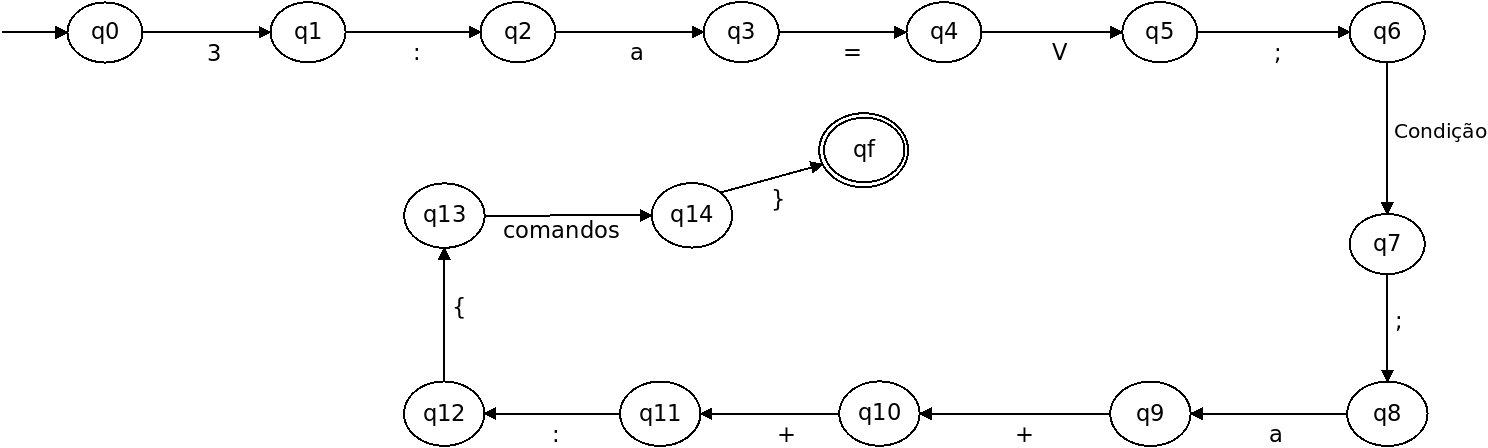
Gramática:

S→ 3A

A→ :C: {corpo de instruções}

C→ i= ‘0’; i operador de condição n ; i ++

Autômato:



4 e 5 Executa um comando de acordo com uma condição verdadeira (4) ou falsa (5)

Gramática:

S→4A

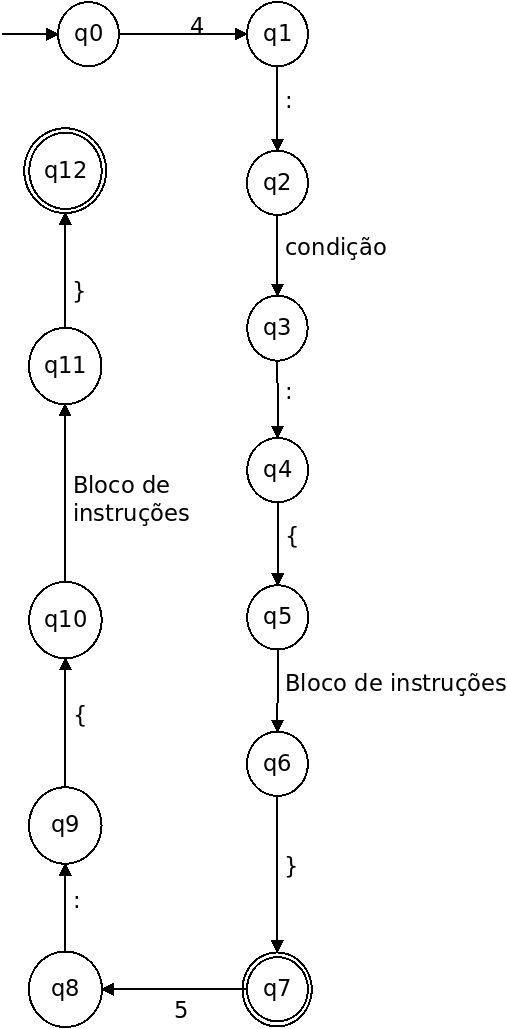
A→ :C:B

B→ { bloco de instruções}E

C→ V operador V

E→ £ | 5:B

Autômato:



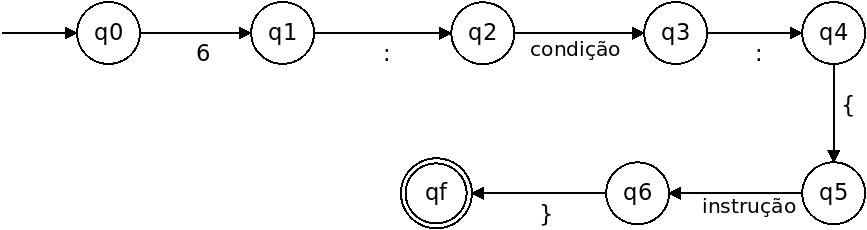
6 Executa um comando até uma condição seja verdadeira

Gramática:

S→6A

A→ :C:{bloco de instruções}

C→ i operador de condição n

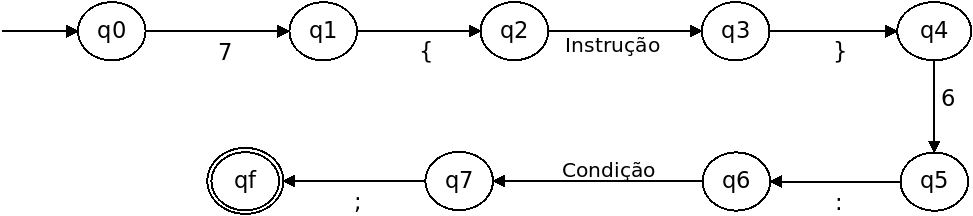
Autômato:

“7” os comandos serão executados ao menos uma vez antes de verificar se a condição é verdadeira.

Gramática:

S→ 7A

A→{bloco de instrução}**E**

E→ 6: condição;

“8” Várias condições dividas em “casos”, executando vários comandos separadamente

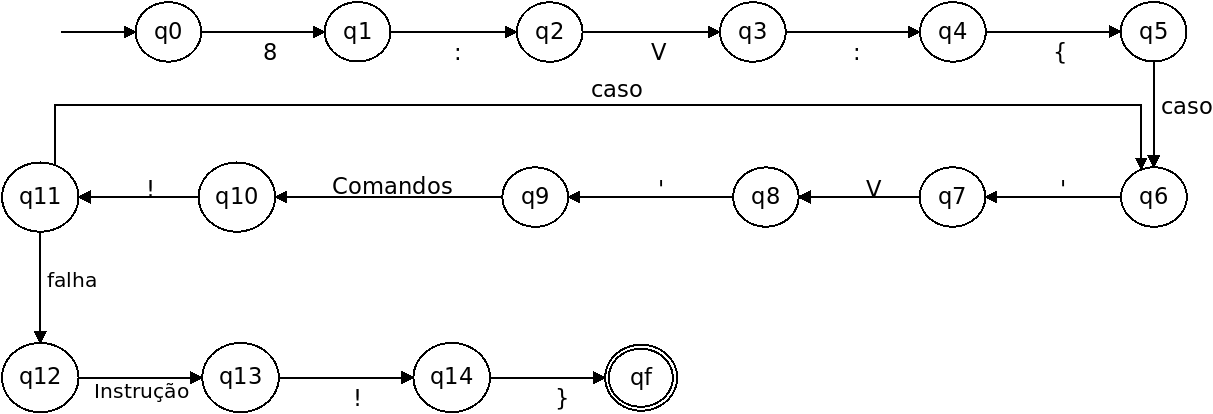
Gramática:

S→8A

A→ :V:{E}

E→ caso ‘V’ bloco do programa ! E | caso ‘V’ bloco do programa ! D

D→ falha bloco de instrução !

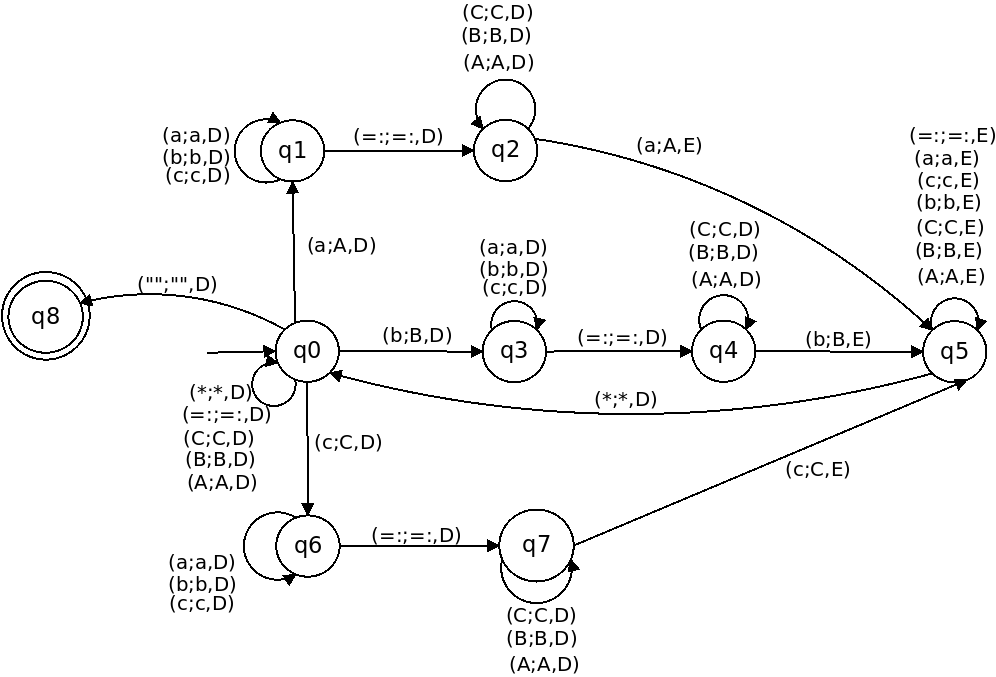
Autômato:

**8.Máquina Turing**

Comparação de palavras em usando uma MT Na Linguagem Bes.

Σ={a,b,c}

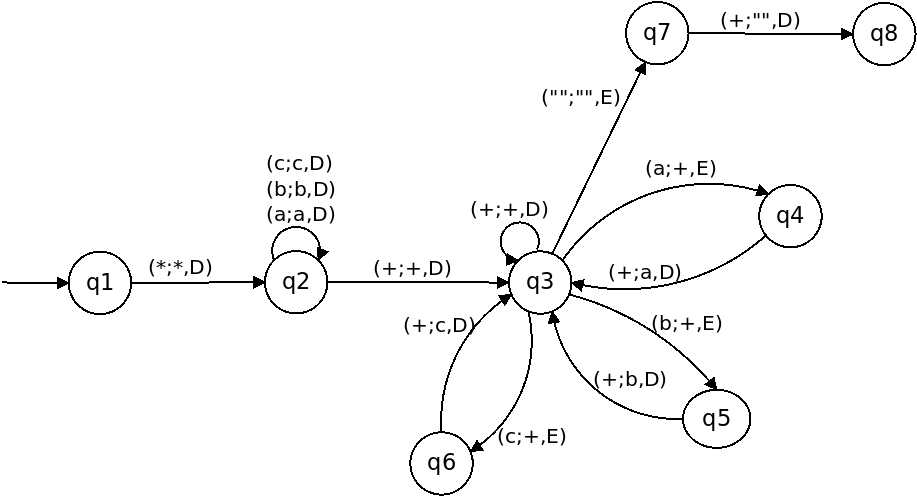
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| \* | B | b | a | =: | B | b | a |  |  |



Concatenação (juntar em uma cadeia sequencial) de palavras usando uma MT na Linguagem BES.

Σ={a,b,c}

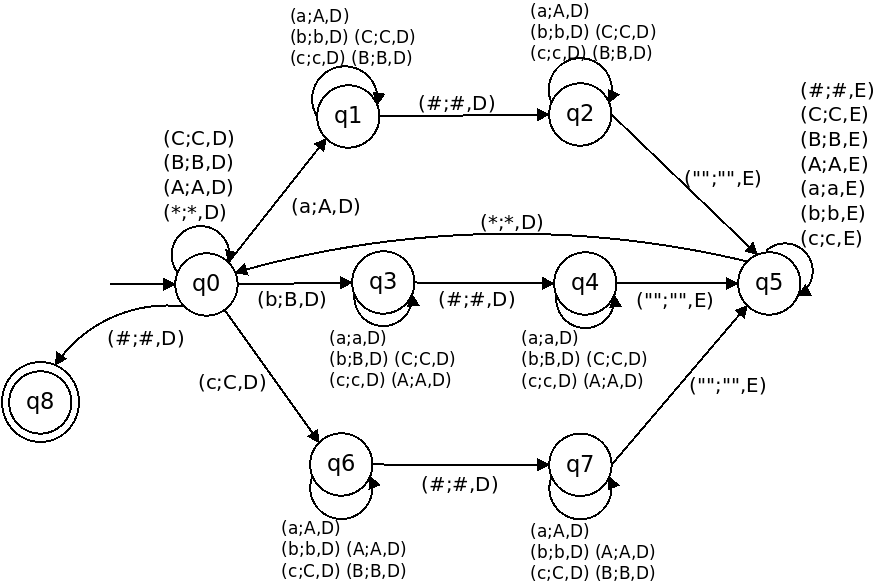
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| \* | C | c | c | c | + | b | b | b |  |



Procura os caracteres de uma palavra em um texto usando uma MT na Linguagem BES.

Σ={a,b,c}





# **9.Referências**

# **Thiago Magalhães,** Thiago Nicolau Magalhães de Souza Conte docente de programação estruturada, Mestre em Engenharia Elétrica (2015), na área de computação aplicada, com ênfase em Inteligência Computacional, formado pelo Programa de Prós-Graduação em Engenharia Elétrica (PPGEE) da Universidade Federal do Pará (UFPA), Especialista em Desenvolvimento de Sistemas Baseados em software livre (2009) pela Universidade da Amazônia (UNAMA) e em Docência para Educação Profissional pelo Centro Universitário SENAC (2016), e Graduado em Tecnologia Em Processamento de Dados (2007 - UNAMA).

**Anderson Costa,** docente de linguagens formais, Mestre em Ciência da Computação pelo Programa de Pós-Graduação em Ciência da Computação da Universidade Federal do Pará (PPGCC-UFPA), possui graduação em Bacharelado em Ciência da Computação pela Universidade Federal do Pará (UFPA) e graduação em Licenciatura Plena em Matemática pela Universidade do Estado do Pará (UEPA).  É sócio da empresa de desenvolvimento de software Execute Soluções em T.I. É implementador oficial MPS.BR, analista e desenvolvedor de sistemas de software. Possui experiência em práticas de ensino e nas seguintes áreas de Ciência da Computação.

**Youtube**

Canal Gabriel Skrilatt

https://www.youtube.com/user/GSorkut

Canal Linguagem C Programação Descomplicada

<https://www.youtube.com/user/progdescomplicada>

Canal Marcelo Pinheiro Montanher

<https://www.youtube.com/user/marcelomontanher>

Canal Alezy Oliveira

https://www.youtube.com/user/alezyoliveira