

Faculdade de Ciências Exatas e da Engenharia

Licenciatura em Engenharia Eletrónica e Telecomunicações

Arquitetura de Computadores

2017/2018

**PizzO – Pizzaria Online**



Rodrigo Jardim, nº 2032416

André Brás, nº 2032216

Funchal, 6 de maio de 2018

Indice

# 1. Introdução

Este trabalho consistiu no desenvolvimento de um programa que permita o controlo dos pedidos de comida *online* de uma pizzaria que faz entregas. O *display* usado para este trabalho tem uma dimensão de 7\*16 (7 linhas de 16 carateres – *bytes*), sendo que isto irá permitir simular a pizzaria *online* onde os clientes poderão fazer as suas encomendas e consoante o seu histórico, poderá ter um desconto na pizza de menor valor na compra atual.

O programa desenvolvido para a pizzaria online foi criado em *assemby* para o processador PEPE. Para testar este mesmo programa foi necessário recorrer ao simulador JAVA.

# 2. Objetivos

O objetivo principal deste trabalho, já referido na introdução, foi a criação de um *site* de uma pizzaria onde é possível a compra de pizzas, sendo que para efetuar essa compra, o utilizador precisa de estar já registado. Caso ainda não esteja registado, também é possível fazer o seu registo para poder usufruir das compras de pizzas. Cada utilizador, após cada compra, o valor dessa compra será guardado no histórico do mesmo, sendo que quando esse histórico é igual ou ultrapassa os 50 euros, ou então se na compra atual o valor acumulado ultrapassa o valor de 50 euros gastos, o utilizador usufrui de um desconto na pizza de menor valor.

A interface com o utilizador é feita através de um *display,* tendo dois botões de controlo, o “**Pin\_OK**” e “**NR\_SEL**” que permitem ao utilizador escolher uma das opções apresentadas no *display* e fazer a referida validação dessa opção. Para o caso de criar um registo ou fazer *login* no *site,* o utilizador também tem mais dois campos para poder introduzir o *username* e a *password*.

A nível de implementação, criou-se uma base de dados em que esta armazena todos os dados referente ao utilizador em questão (dados de login e histórico de compras).

A programação em linguagem *assembly* também foi um dos objetivos deste trabalho, sendo esta linguagem de baixo nível. Em relação a esta linguagem, é necessário ter uma gestão criteriosa da memória para que não exista sobreposição de dados e instruções, sendo que isto levaria a um mau funcionamento do programa.

# 3. Discussão de Resultados

Como já foi referido anteriormente, é necessário ter uma gestão equilibrada de memória para não haver falhas de programa. Assim sendo no anexo (A) é apresentado o mapa de utilização da memória. Em relação aos periféricos de entrada, estes foram posicionados da seguinte forma:

* Endereço 0150H até 0157H: Periférico de entrada para a introdução do *username*
* Endereço 0160H até 0167H: Periférico de entrada para a introdução da *password*
* Endereço 0170H: Periférico de entrada para a validação numérica escolhida (“**Pin\_Ok**”)
* Endereço 0180H: Periférico de entrada para a introdução da opção pretendida (“**NR\_SEL**”)

Em relação aos periféricos para a introdução do *username* e da *password* é necessário ter os seguintes cuidados:

* Como a limpeza destes periféricos é feito com o carater 0H, este carater não pode ser usado no *username* nem na *password*, ou seja, caso o utilizador introduza no *username* “AA00A”, apenas será validado “AA”, o mesmo acontecendo para a password, caso seja inserido “BBB0B”, apenas será validado “BBB”;
* O username tem de ter um valor mínimo de 1 carater e um máximo de 8;
* A password tem de ter um valor mínimo de 3 carateres e um máximo de 8.

## 3.1. Menu Inicial e Menu Principal

O programa é inicializado com um menu em que apresenta uma mensagem de boas vindas ao utilizador ao *site* da pizzaria “PizzO”. Após a validação do OK (“**Pin\_OK**”) é mostrado o menu principal em que apresenta 3 escolhas: criar uma conta, fazer *login* ou sair do site. Estas opções são escolhidas através do periférico de entrada “**NR\_SEL**” sendo posteriormente validado pelo periférico de entrada “**Pin\_OK**”.

### 3.1.1. Criação de uma nova conta (Registo)

Para a criação de uma nova conta é necessário inicialmente verificar de ainda existe espaço na base de dados para a introdução de um novo registo, sendo esta verificação feita pela rotina “**RotVerificacaoReservas**” (rotina que verifica quantos utilizadores já estão escritos na base de dados), estando no anexo B o fluxograma desta rotina. Caso a base de dados já esteja lotada, é mostrado ao utilizador uma mensagem a dizer que não é possível fazer mais nenhum registo. É de salientar que a base de dados deste programa, por questões de simplicidade, apenas é possível o registo de 10 utilizadores.

Após feita essa verificação, é necessário ainda é necessário fazer mais algumas, sendo estas feitas nos periféricos de entrada destinados à introdução do *username* e a *password*. As verificações feitas são as seguintes:

* Verificar se o utilizador não preencheu um dos campos (*username* ou *password*);
* Verificar se a password apresenta um mínimo de 3 carateres;
* Verificar se o *username* introduzido é igual a algum dos que já estão na base de dados.

Para a verificação do não preenchimento de um dos campos, foi implementado a rotina “**RotVerificarCampos**”, que verifica quantos carateres tem o *username* e a *password* inserida*.* Caso não apresente nenhum carater nalgum destes, é apresentado uma mensagem de erro no *display*, sendo que depois volta ao menu para introduzir novamente o *username* e a *password.*

Para a verificação do mínimo de carateres precisos para a password, a rotina implementada foi a “**RotVerificacao\_Pass**”, em que verifica quantos carateres tem a *password* inserida. Caso a password apresente um valor inferior a 3 carateres, é mostrado uma mensagem de erro no *display*, levando a que depois volte ao menu para introduzir o *username* e a *password*.

Para a verificação do *username* introduzido seja igual ou não a algum que esteja na base de dados, a rotina feita para essa verificação foi a rotina “**RotVerificacao\_Username**” em que verifica a quantidade de carateres iguais em cada passagem de utilizador na base de dados. Caso esse valor chegue a 8 (máximo de carateres do *username*) quer dizer que existe um utilizador com o mesmo *username* e assim sendo, é mostrado no display uma mensagem de erro em que o utilizador necessita de introduzir um *username* diferente voltando para o menu do registo para a introdução do *username* e *password*.

Após feita estas verificações todas e nenhuma delas tenha dado uma mensagem de erro, é mostrado no display que o registo foi efetuado com sucesso e ao clicar no OK (“Pin\_OK”), o utilizador terá o seu registo na base de dados e assim poderá visualizar o menu pedido/*logout*.

### 3.1.2. *Login* do cliente

Para fazer o *login* no *site* é necessário verificar se a base de dados apresenta já algum registo efetuado, sendo isso feito pela rotina “**RotVerifiacacaoReservas**”. Caso ainda não tenha nenhum registo efetuado, é mostrado no *display* uma mensagem de erro em que diz que não existe nenhum registo na base de dados, sendo que depois da validação do OK (“**Pin\_OK**”) do menu, é apresentado o menu principal para o utilizador pode fazer o seu registo no *site*.

Após essa verificação, é necessário fazer as mesmas verificações que foram explicadas anteriormente no registo, tais como a verificação se algum dos campos não foi inserido e a verificação se a *password* tem pelo menos 3 carateres.

Também é necessário verificar se o *username* e a *password* são iguais a que estão na base de dados, tendo a rotina “**RotVerificacao\_User\_Pass\_Login**” em que verifica na base de dados se existe um utilizador com as credenciais iguais às que foram inseridas. Caso não existe nenhum utilizador na base de dados com as mesmas credenciais (*username* ou *password*), é mostrado uma mensagem de erro em que diz qual foi a credencial de está incorreta e volta ao menu anterior para poder introduzir corretamente as credenciais.

Caso esteja tudo correto, é mostrado no display que o login foi efetuado com sucesso e ao clicar no OK (“**Pin\_OK**”), o utilizador poderá visualizar o menu pedido/*Logout*.

## 3.2. Menu Pedidos

A partir do momento que o utilizador seja validado, será apresentado um menu com duas opções: Fazer pedido ou *Logout*. Tal como nos pontos anteriores, para proceder à escolha das opções, o utilizador terá de pressionar a tecla 1 ou 2 e após a escolha terá de pressionar obotão OK (“***PIN\_OK”***) para que a sua escolha seja validada. Se o utilizador optou pela opção “Fazer Pedido”, será apresentado o menu das pizzas ao utilizador, caso contrário será feito o *Logout* do utilizador.

### 3.2.1. Logout

Caso o utilizador opte pelo Logout, o sistema irá limpar o *username* e *password,* voltando assim ao menu de boas vindas da pizzaria.

### 3.2.2. Menu \_Pizzas, Menu\_Escolha\_Tamanhos e Menu\_Continuar\_Compras\_Pagamento

Este menu é responsável pela apresentação das pizzas existentes no sistema, perfazendo um total de 5 pizzas. Existem 2 tamanhos à escolha do consumidor, pequeno ou grande, sendo o preço de cada tamanho 5 e 8 euros, respetivamente. Após ser concluído esta etapa, será apresentado ao utilizador um menu com 2 opções: pedir mais pizzas ou pagamento. Caso opte pela opção de pedir mais pizzas, será apresentado novamente o menu das pizzas ficando guardado na memória o valor da pizza que foi pedida, através da rotina “**RotAtualizarValoresCompras**”. Caso o utilizador selecione o pagamento, o sistema irá apresentar um menu onde será mostrado o total das compras efetuadas pelo utilizador, o valor do histórico e o desconto que irá usufruir caso ultrapasse os 50 euros de histórico.

### 3.2.3. Menu\_Visualizar\_Total\_Compras

Como já foi referido anteriormente, quando o sistema chegar a este menu, será apresentado 3 coisas: Total das compras, histórico do utilizador e o desconto. Para que seja apresentado ao utilizador estes parâmetros, foi necessário implementar uma série de rotinas, como por exemplo:

* **RotDescontaHistorico**: Responsável por verificar se o histórico do utilizador possui um valor igual ou superior a 50 euros. Caso isto se verifique, será descontado 50 euros ao histórico e será verificado quais foram os tamanhos comprados por parte do usuário. Se optou por uma pizza pequena, este terá um desconto de 2,50 euros, caso contrário terá um desconto de 4,00 euros.
* **RotConverte\_Num\_Char\_Total\_EUR\_CENT:** Destina-se a representar o valor total das compras do utilizador no display.
* **RotConverte\_Num\_Char\_Historico\_EUR\_CENT:** Destina-se a representar o valor do histórico do utilizador no display
* **RotConverte\_Num\_Char\_Desconto\_EUR\_CENT:** Destina-se a representar o valor do desconto no display caso este exista.

Após o pagamento, o sistema terá de limpar o total de compras efetuadas pelo utilizador também como as quantidades de pizza que este escolheu. Para que isto se torne possível, foi necessário implementar a rotina, **RotLimpaTotalUtilizador**. Depois deste procedimento, o sistema fará com que apresenta o menu que permite ao utilizar fazer mais um pedido ou então proceder ao *logout*.

## 3.3. Rotinas implementadas

As seguintes rotinas foram criadas de forma a que o trabalho estivesse mais compacto e que fosse mais fácil de fazer alterações. Algumas destas rotinas já foram explicadas no decorrer da explicação de alguns menus, sendo que apenas serão explicadas aquelas que ainda não feito nenhuma referência. Assim sendo, as rotinas ainda não referidas são:

* “**RotMostrarDisplay**” – Copia os menus pré desenhados;
* “**RotLimparDisplay**” – Limpa o display com o carater 20H;
* “**RotLimparPerifericosOpcoes**” – Limpa os periféricos “**Pin\_OK**” e “**NR\_SEL**”;
* “**RotLimparUserPass**” – Limpa a linhas onde estão situados os periféricos para a introdução do *username* e da *password*;
* “**RotValidacao**” – Espera um OK (“**Pin\_OK**”) do utilizador para continuar o programa;
* “**RotLimpa\_Char\_Erro\_User\_Pass**” – Limpa os carateres que não poderão ser utilizados no *username* e *password* (ex: caso o utilizador introduza no *username* “AA00AA”, esta rotina verifica carater a carater e quando apanha um 0 limpa tudo para a frente, sendo que depois de encontra o 0, o que está para a frente não pode ser usado);
* “**RotMostrarDisplay\_User\_Pass**” – Copia os periféricos *username* e *password* para o *display*;
* “**RotEscreve\_Na\_Memoria**” – Guarda os periféricos username e password para a base de dados;

# 4. Conclusão

# 5. Anexo A

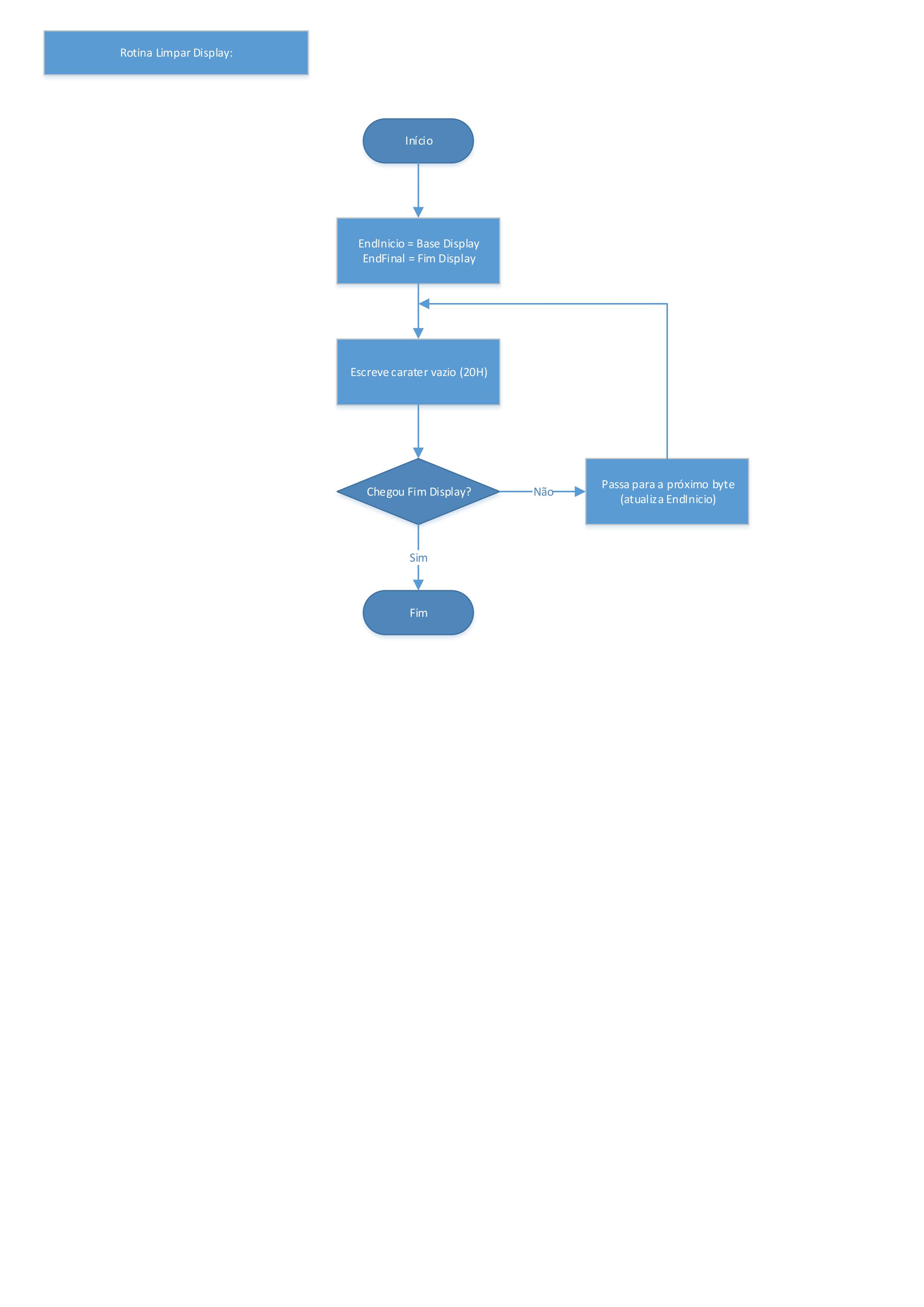
## 5.1. Mapa de utilização de memória



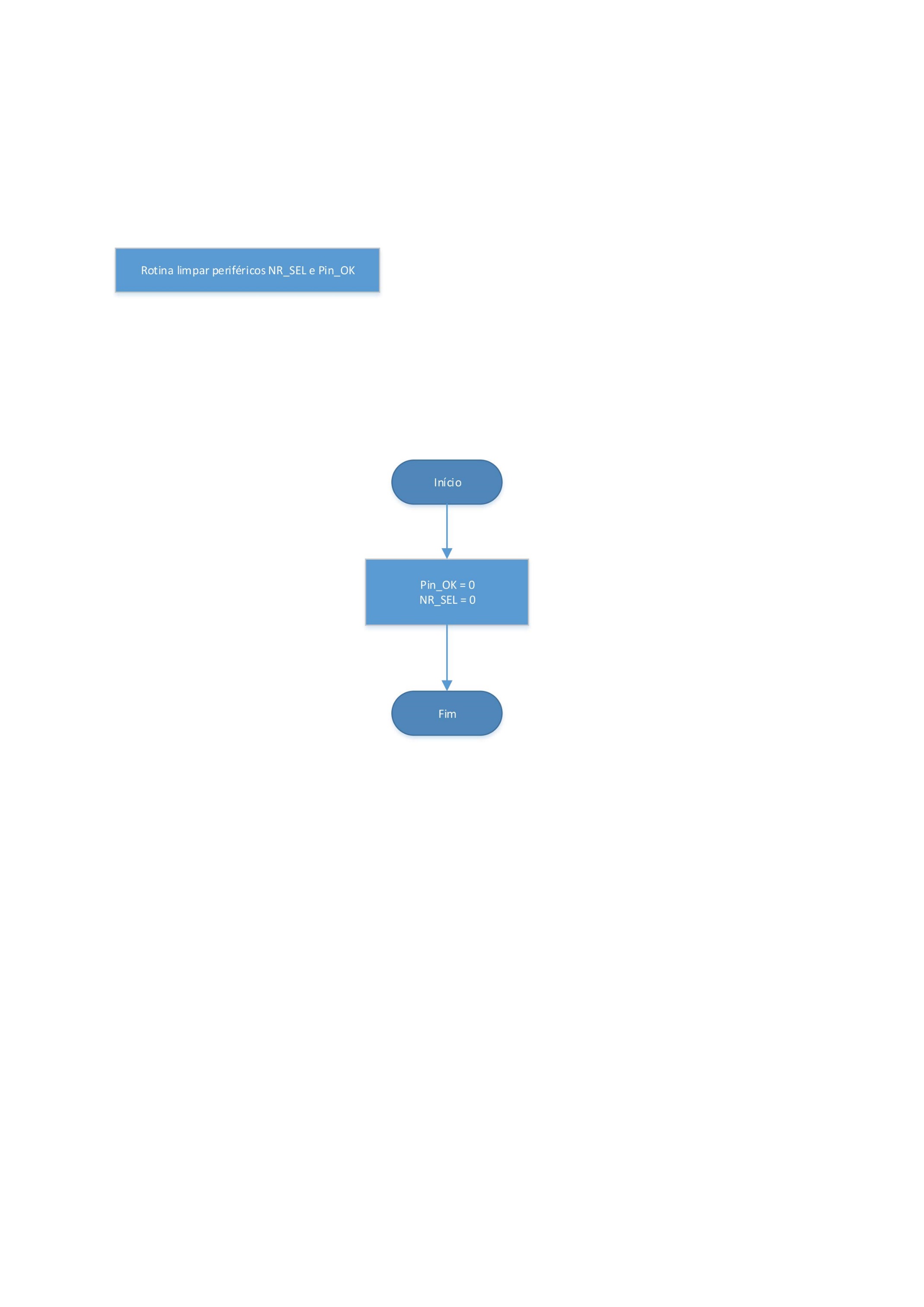
## 5.2. RotMostrarDisplay

## 

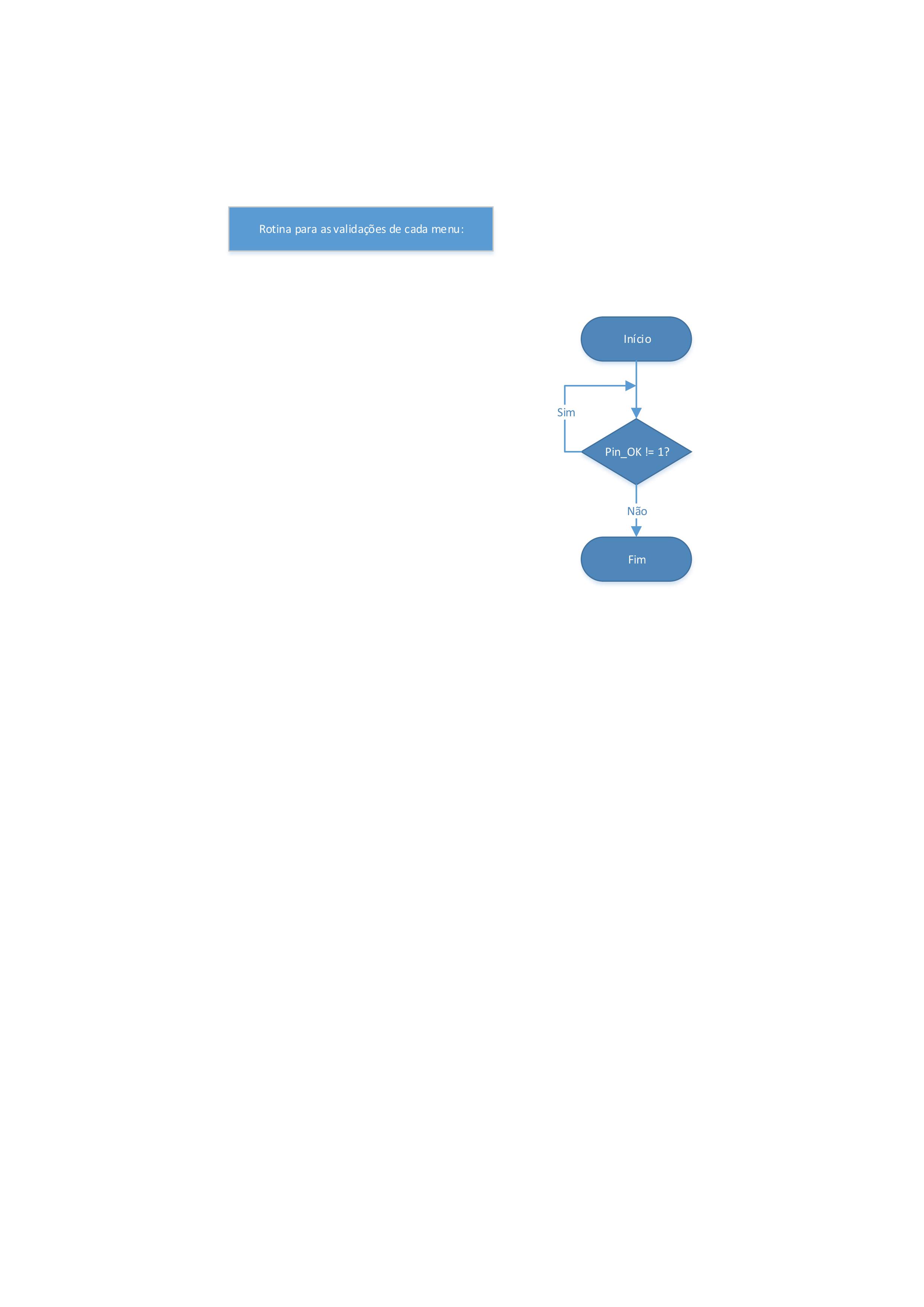
## 5.3. RotLimparDisplay



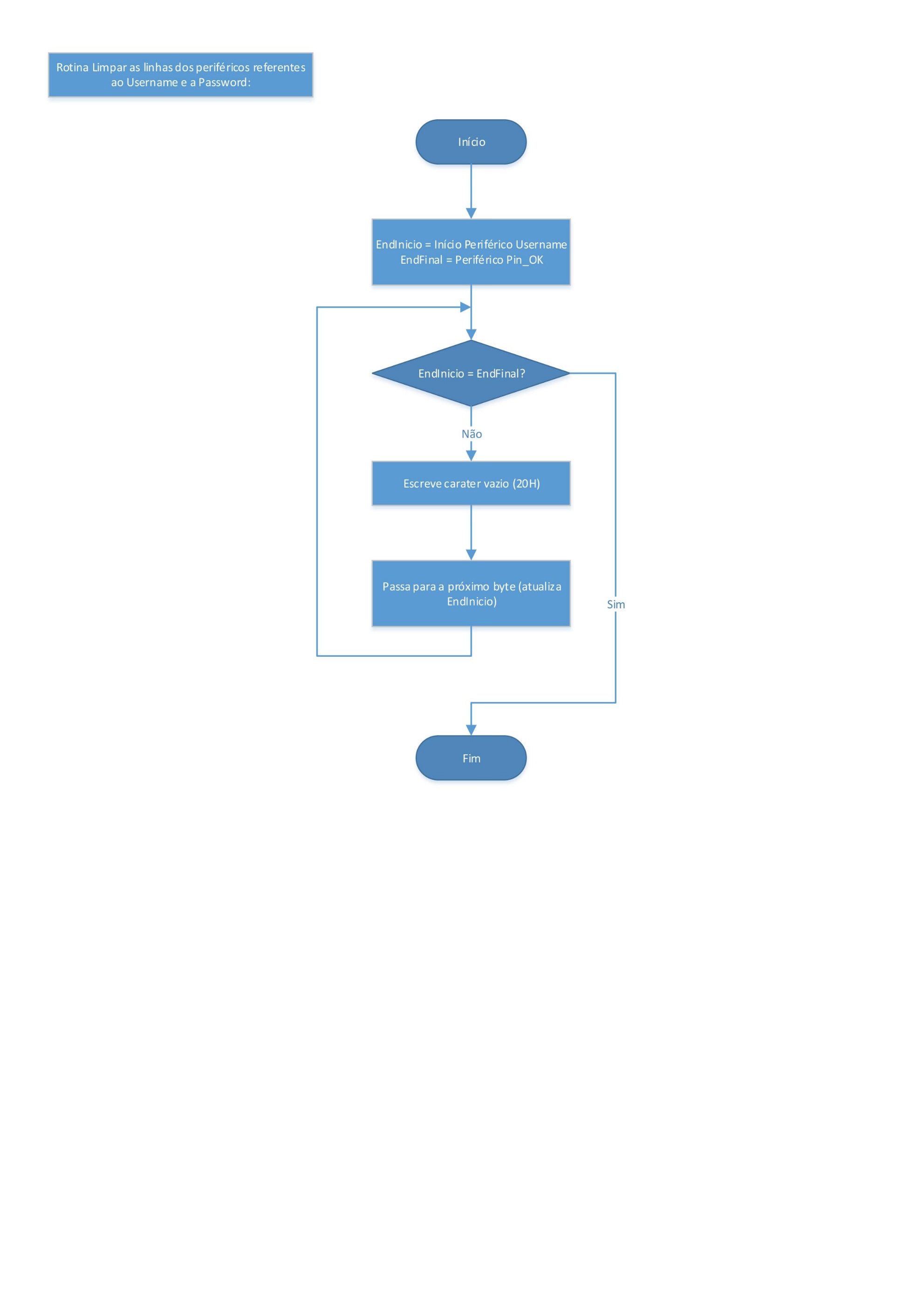
## 5.4. RotLimparPeriféricosOpcoes



## 5.5. RotValidacao



## 5.6. RotLimparUserPass



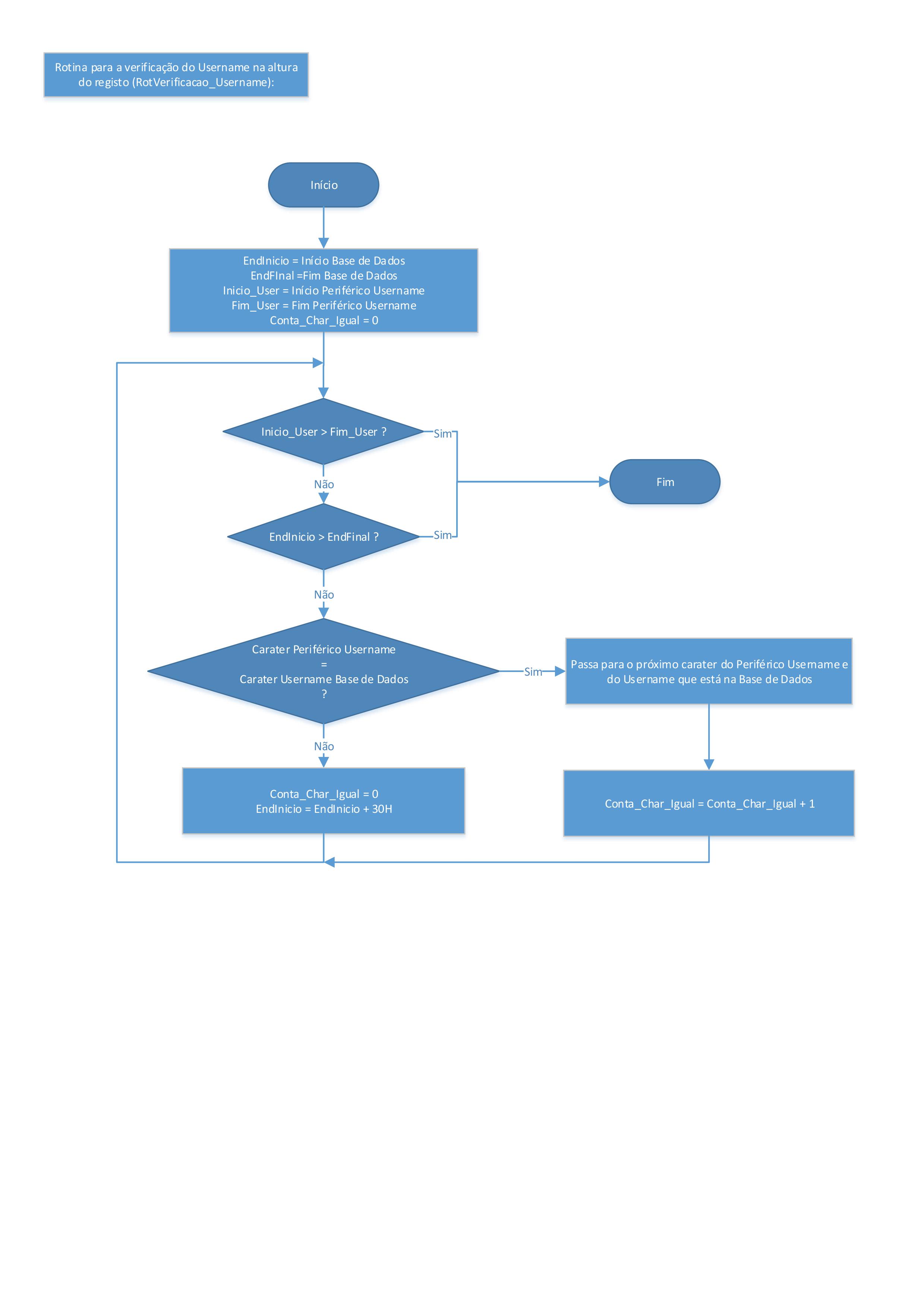
## 5.7. RotLimpa\_Char\_Erro\_User\_Pass

## 5.8. RotVerificacaoReservas

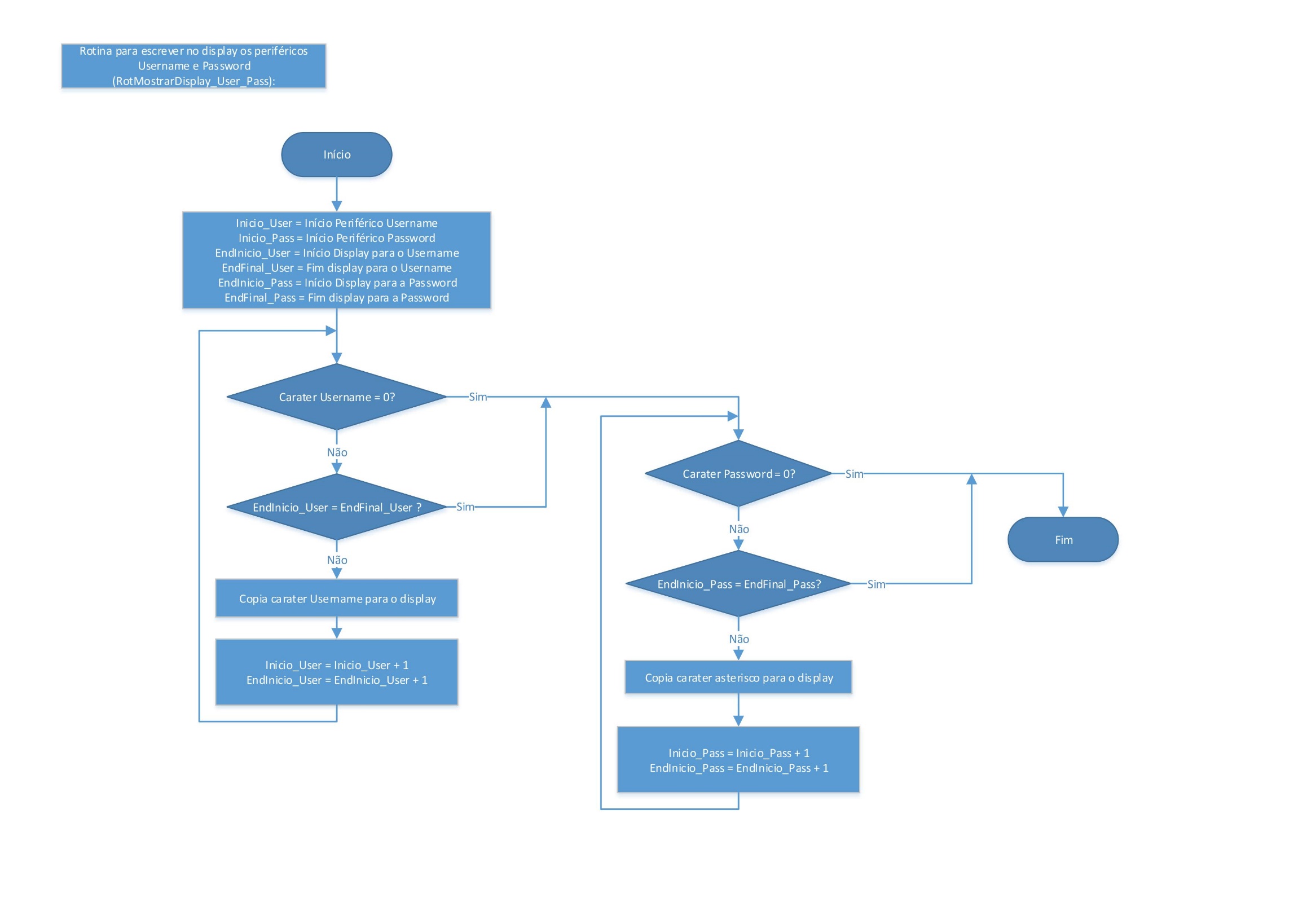
## 

## 5.9. RotVerificarCampos

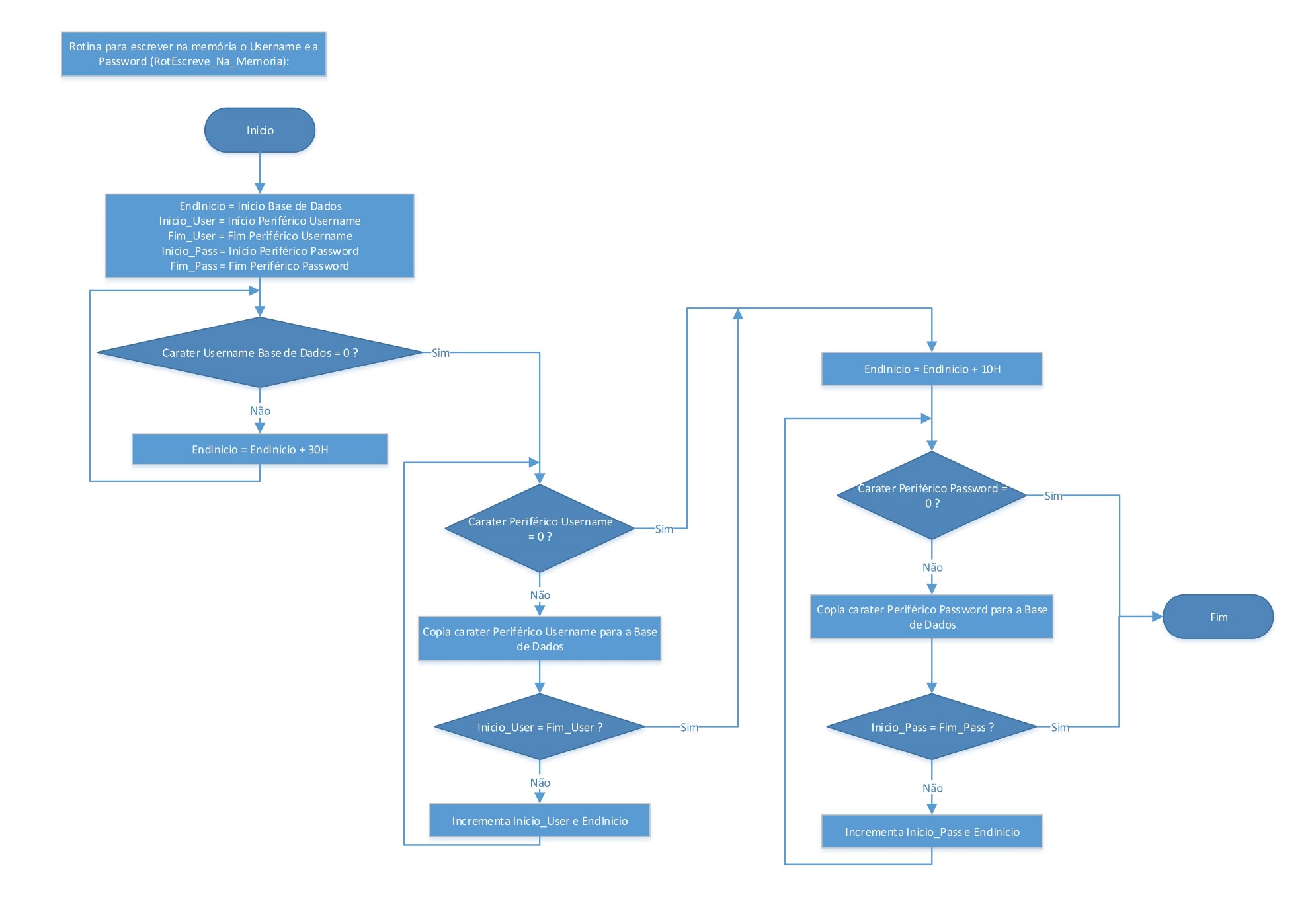
## 5.10. RotVerificacao\_Username



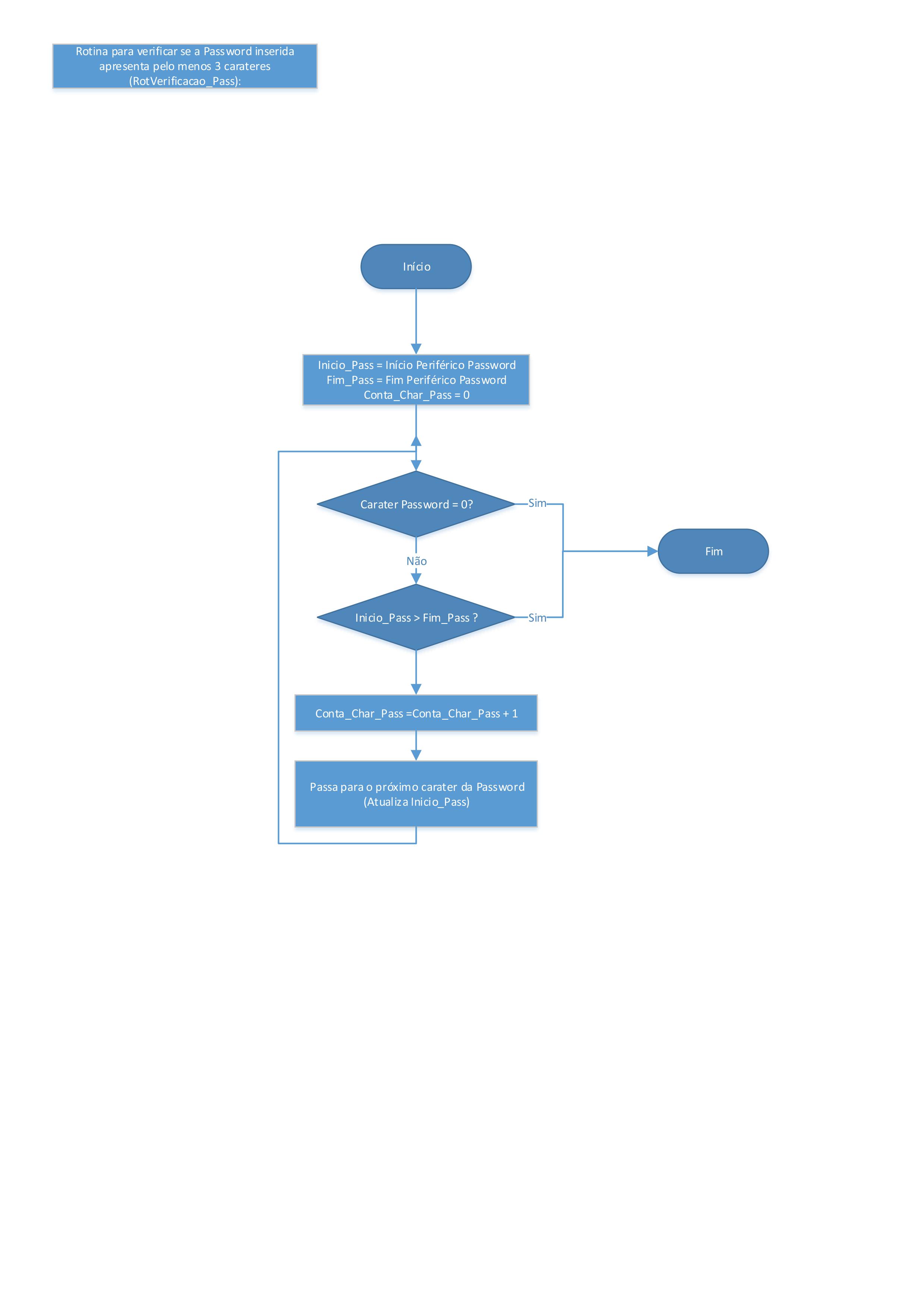
## 5.11. RotMostrarDisplay\_User\_Pass



## 5.12. RotEscreve\_Na\_Memoria



## 5.13. RotVerificacao\_Pass



## 5.14. RotVerificacao\_User\_Pass\_Login

