AppGlass Master Console 1.0

O Master Console é uma linguagem de script desenvolvida por Felipe Durar que foi feita com o objetivo de ser fácil, pequeno e especialmente para aplicações console.

Foi desenvolvida inteira em C# para ser executado em computadores com Windows de preferência 32bit.

Ele foi baseado um pouco na linguagem Assembly e um pouco nas linguagens de auto nível mas sendo fácil como Batch.

Não pode se esquecer que o Master Console é Case Sensitive, ou seja, ele se diferencia de maiúsculas para minúsculas.

Pontos negativos da linguagem:

O principal problema é que é muito difícil esta linguagem detectar um erro no código fonte, por isso você tem que verificar mais de uma vez seu código pois provavelmente quando o interpretador encontrar um erro ele será simplesmente ignorado.

Os únicos tipos de erro que serão detectados serão os erros graves que fariam o interpretador travar, nestes casos ele mostraria uma mensagem de erro fatal e entraria no modo prompt.

Pontos positivos da linguagem:

O ponto positivo desta linguagem é a facilidade, é muito fácil você decorar como utilizar esta linguagem, ela é tão simples quanto o batch script.

Registradores:

Os registradores são como variáveis que não pode modificar seu nome, somente seu valor e cada um serve para uma tarefa diferente e alguns são vazios par uso de quem esta programando, veja os registradores abaixo:

SC = Screen / Registrador da tela

RC = Replace Char / Não é permitido o uso de espaços então isto é usado para substituir underline por espaço.

A1, A2, A3, A4, B1, B2, B3, B4 = Registradores de uso para o usuário, eles começam vazios.

IR = Intern Registrator / Registrador Interno que não deve ser utilizado pelo usuário.

KB = Keyboard / Recebe dados do teclado

CM = Envia comando ao cmd.exe e executa

Mais adiante você vai ver exemplos de como utilizar os registradores.

Como utilizar comentários:

Os comentários são feitos em uma linguagem para que o código seja mais bem organizado, assim você vai ter mais facilidade na hora de editar o código fonte.

Quando você utiliza um comentário ele é simplesmente ignorado pelo interpretador ou compilador, ou seja, ele não faz nada.

Veja um exemplo de como escrever um comentário abaixo:

```
* Tudo que estiver nesta linha será ignorado
```

O comando MOD:

O comando MOD é muito parecido com o MOV do Assembly, o primeiro valor é o destino e o segundo é o que será copiado, veja um exemplo abaixo:

```
MOD A1 KB
```

Neste caso ele esta indicando que A1 será igual ao valor do que o usuário digitar e pressionar enter pois A1 esta vazio e KB recebera o que o usuário digitar, isto ficaria assim em C#:

```
String A1 = "";

A1 = Console.ReadLine();
```

Podemos estar enviando este valor para a tela movendo A1 para SC:

```
MOD SC A1
```

Assim mostraria o valor de A1 no prompt.

Strings e inteiros:

Quando vamos mover uma string ou um inteiro para um registrador devemos colocar um ‘$’ antes da frase para que ele saiba que aquilo é uma string ou inteiro.

Aqui você não vai encontrar diferença para declarar um string ou inteiro, a partir do momento que você escrever uma frase, automaticamente o interpretador já vai saber que aquilo se trata de uma string e ao mover somente números, automaticamente o interpretador já vai saber que aquilo é um inteiro.

Veja um exemplo de mover uma string e um inteiro para um registrador:

MOD A1 $Minha_frase

MOD A1 $14

Um detalhe que não podemos esquecer é de falar é que não pode ter espaços desnecessários em uma linha, por isso existe o registrador RC do tipo booleano que substitui underlines por espaços automaticamente.Vamos aprender utilizar o RC mais tarde.

Booleanos:

Quando vamos mover um valor True ou False para um registrador é necessário colocar um ‘#’ (sustenido) antes de escrever o valor, você utilizara isto quando aprender a utilizar o registrdor RC que é booleano.Veja exemplos abaixo:

MOD A1 #True

MOD A1 #False

MOD A1 #1

MOD A1 #0

Utilizando o registrador SC:

O Registrador SC (Abreviação de SCreen) é o registrador da tela, ou seja, tudo que for movido para ele será mostrado no console, ao mover valores booleanos ele mostrara 0

ou 1 que são os números que significam False e True.Veja um exemplo abaixo de como utilizar o registrador SC:

MOD SC $Helo_World

Ou

MOD A1 $Helo_World

MOD SC A1

Nos dois casos acima ele simplesmente vai exibir Helo_World na tela, toma cuidado pois não é permitido o uso de espaços desnecessários nas linha então no lugar de espaço coloque underline.

Agora você deve estar se perguntado, como eu faço para pular linha se não tem uma função como WriteLine nesta linguagem, na verdade é só você mover o valor $JL para SC que significa Jump Line, veja um exemplo abaixo:

MOD SC $JL

Utilizando o registrador RC:

O registrador RC (Replace Char) é utilizado par substituir underlines por espaços pois não é permitido o uso de espaços desnecessário, como ele é um registrador booleano basta você defini-lo como True para ativa-lo, veja este exemplo:

MOD RC #True

Quando você digita isto ele automaticamente será ativo e a saída como SC não mostrara mais underlines, veja esta exemplo:

MOD RC #True

MOD A1 $Helo_World

MOD SC A1

Neste caso será exibido na tela Helo World e não Helo_World.

Utilizando o registrador KB:

O Registrador KB (KeyBoard) serve para receber o que o usuário digitar e pode salvar como uma string, inteiro ou booleano, veja um exemplo abaixo da utilização do KB:

MOD RC #True

MOD SC $Digite_alguma_coisa:_

MOD A1 KB

MOD SC $JL

MOD SC $Voce_escreveu_

MOD SC A1

Pause APE

No exemplo acima ele coloca o que o usuário digitar em A1 para ser utilizado depois.

Utilizando o registrador CM:

Este registrador envia um comando ao cmd.exe para que ele interprete o código, por exemplo, se mover $ECHO_Helo_World para CM ele mostrara Helo World na tela já que é este o código utilizado para escrever algo na tela em Batch Script, veja outro exemplo abaixo:

MOD RC #True

MOD SC $Digite_alguma_coisa:_

MOD A1 KB

MOD SC $JL

MOD CM A1

Pause APE

Neste caso o que for digitado será interpretado pelo cmd.exe e o resultado será exibido n tela.

Para que serve o registrador IR:

O registrador IR (Intern Registrator) é um registrador de uso do próprio interpretador.

Não é recomendado que você faça alteração deste registrador pois ele é utilizado por funções internas do interpretador, então a melhor coisa a fazer é deixá-lo parado.

Como utilizar a função interna Pause:

A função Pause você pode enviar dois valores, ou APE ou APK, a diferença é simples, o APE somente sai do pausa quano você pressiona enter e o APK sai do pausa quando você aperta qualquer tecla, veja um exemplo de como utilizar a Pause:

Pause APE

Pause APK

Como utilizar a função interna Sleep:

A função Sleep serve para dar um atraso no código, ao colocar o tempo você deve colocar em milésimos, quando acaba o tempo ele continua normalmente, veja este exemplo abaixo de como utilizar o Sleep:

Sleep 5000

No caso acima ele da uma pausa de 5 segundos e continua o código normalmente.

Como utilizar a função interna Beep:

A função Beep emite um sinal sonoro parecido com o que é emitido no momento do Post na Bios.

Você envia depois a quantidade de vezes que emite o sinal sonoro, veja este exemplo abaixo:

Beep 5

Neste caso ele emite 5 vezes o sinal sonoro.

Como utilizar Areas:

Areas são linhas que são gravadas por nome que podem serem voltadas utilizando o GoTo Area, veja este exemplo abaixo:

Area NomeDaArea

GoTo NomeDaArea

No caso acima é gerado um loop infinito pois toda hora ele fica voltando o código para a linha anterior, se você colocar entre estas duas linhas um MOD SC $Voltar_a_area ele toda hora escreveria na tela Voltar_a_area sem parar.

Veja outro exemplo:

MOD RC #True

Area LoopA

MOD SC $Voltar_a_LoopA

MOD SC $JL

GoTo LoopA

Neste exemplo ele toda hora escreve Voltar a LoopA sem parar.

Como utilizar o Restricted Areas:

O Restricted Area faz a mesma coisa que uma área, só muda que ele não executa o que tem dentro da área enquanto você não usa o GoTo, veja um exemplo abaixo:

MOD SC $Inicio

Restricted Area Loop

MOD SC $Dentro_da_area

End Area

MOD SC $JL

MOD SC $Depois_da_area

GoTo Loop

No programa acima você nota que ele ignora a área loop na primeira vez e quando chega no GoTo ele entra no loop e continua normalmente.

Como utilizar o condicional If:

O If verifica se duas condições são verdadeiras ou falsas, se for verdadeira ele faz o que esta na próxima linha, se for falso ele pula duas linhas e continua o código, veja um exemplo abaixo.

```
MOD RC #True

MOD A1 $10

MOD A2 $11

If A1 > A2

GoTo Maior

GoTo Menor

GoTo Final

Area Maior

MOD SC $JL

MOD SC $A1_e_maior_que_A2

GoTo Final

Area Menor

MOD SC $JL

MOD SC $A1_e_menor_que_A2

GoTo Final

Area Final

Pause APE
```

No exemplo acima é escrito na tela se 10 é maior que 11.

O If tem muita importância nesta linguagem pois é ele quem vai testar as condições e verifica-las se são verdadeiras ou falsas, porem não vamos entrar em muitos detalhes sobre ele.

Operações matemáticas:

Agora você aprendera a utilizar o comando MAT para realizar as quatro operações básicas, a adição, subtração, divisão e multiplicação.

O primeiro valor a ser passado é o destino do resultado, o segundo é o número que será utilizado na conta, o terceiro é o operador e o quarto é o outro valor, veja abaixo um exemplo:

MAT A1 $10 * $10

No caso acima ele colocara em A1 o valor de 100 que é o resultado da operação 10 * 10, lembrando que no computador é utilizado os seguintes operadores para cálculos simples.

+	Adição
-	Subtração
*	Multiplicação
/	Divisão

Utilizando o EC (Extern Comunication):

Este comando foi desenvolvido para a comunicação externa entre dlls e o interpretador, mas não há dlls sendo utilizadas ainda pelo interpretador, mas o EC é bem interessante pois você pode estar colocando vários dados dentro temporariamente.

O Master Console quando é iniciado ele cria um arquivo chamado excom.adr que dentro deste arquivo fica o que é enviado pelo EC, não é possível modificar um valor do que esta dentro do excom por isso é só adicionar outro valor do mesmo nome que ele ganha prioridade e o outro é simplesmente ignorado.

O Sistema de variáveis utiliza o Extern Comunication e sempre que queremos receber o valor que esta la dentro é só utilizar um @ antes do nome, veja um exemplo:

MOD RC #True

EC ADD $Idade $14

EC ADD $Nome $Felipe

MOD SC $Meu_nome_e_

MOD SC @Nome

MOD SC $_e_tenho_

MOD SC @Idade

MOD SC $_anos.

Pause APE

Note basta colocar EC ADD que ele já sabe que vai estar adicionando algo, depois é o nome e o valor, você pode estar recebendo os dados também de outra maneira, utilizando o GET do EC, veja um exemplo abaixo:

MOD RC #True

EC ADD $Idade $14

EC ADD $Nome $Felipe

EC GET A1 Idade

EC GET A2 Nome

MOD SC $Meu_nome_e_

MOD SC A2

MOD SC $_e_tenho_

MOD SC A1

MOD SC $_anos.

Pause APE

No exemplo acima ele recebe o valor utilizando o GET como primeiro argumento, o segundo é o destino e o terceiro é o nome da variável.

Utilizando as Includes:

As includes servem para você estar incluindo arquivos externos no seu código fonte, assim você poderá deixar seu programa mais organizado criando uma include para cada tipo de tarefa.

Você poderá colocar a quantidade de arquivos externos que quiser dentro do seu arquivos principal mas infelizmente você não poderá colocar uma include dentro de um arquivo que já será utilizado como include, ou seja, somente um arquivo poderá ter includes e este arquivo é o principal a ser executado.

Para utilizar um include externa basta escrever Include e na frente colocar o diretório da include, veja um exemplo abaixo:

Include 11inc.mco

GoTo Helo

Area Final

Pause APE

No exemplo acima é incluído o arquivo 11inc.mco e dentro dele tem a área Helo que depois é chamada.

Utilizando funções:

As funções tem grande importância em uma linguagem pois você pode criar uma para cada tipo de tarefa, deixando seu programa mais organizado.

Uma das duvidas que podem surgir é, Qual a diferença de usar funções e utilizar as Restricted Areas?, é muito simples, as Funções podem receber argumentos e quando ela termina ela continua na linha que foi chamda, já as Restricted Areas não recebem argumentos e não volta para linha em que foi chamada. Veja um exemplo abaixo:

MOD RC #True

MOD A1 $Helo_World

CALL ShowStr A1

MOD SC $Antes_da_funcao

Function ShowStr:Text

MOD SC @Text

MOD SC $JL

End Function

Pause APE

No exemplo acima é chamada a função ShowStr que recebe o argumento Text e mostra o texto na tela, note que ao terminar a função ele volta a linha em que foi chamada e continua o código normalmente.

Você pode colocar mais de um argumento para uma função receber, basta você colocá-los entre ponto-e-virgula, veja um exemplo abaixo:

Function ShowText:Text;NumLines

E no momento de chamar a função você utiliza o CALL e delimita por espaço os argumentos, veja um exemplo abaixo.

CALL ShowText $Helo_World $10

Quando você não querer que a função não receba argumentos é só não colocar nada depois dos dois pontos, veja um exemplo abaixo:

Function ShowText:

Lembre se que é obrigatório colocar ponto e virgula depois do nome da função.

Como você deve ter notado acima, quando você recebe um argumento, você deve utilizar um @ e o nome na frente para receber seu valor, isto acontece porque ele vai para o EC.

Variaveis:

As variáveis são capazes de gravar valores temporariamente, para declarar uma variável no Master Console basta utilizar Var, veja um exemplo abaixo.

Var nome $Valor

Para recebê-las você deve utilizar um @ antes do nome pois elas são enviadas ao EC, veja um exemplo abaixo:

MOD A1 @Nome

Infelizmente não tem como mudar o valor de uma variável, então basta declarar outra com o mesmo nome que ela ganha prioridade pois foi a última a ser declarada, veja um exemplo abaixo de como mudar o valor de uma variável.

MOD RC #True

Var Nome $Valor

MOD SC @Nome

Var Nome $Novo_valor

MOD SC @Nome

Pause APE

Neste exemplo acima é declarada uma segunda variável com o mesmo nome que ganha prioridade.

Funções que retornam por apelidos:

Infelizmente esta linguagem não tem Array por isso algumas vezes vai ter funções que pedem apelidos, na verdade quando você coloca por exemplo como apelido o nome div ele vai adicionar variáveis numeradas como div1, div2, div3... e continua, ele é a mesma coisa que as variáveis, só muda que ele adiciona um número na frente.