**funções da linguagem *FSCRIPT***

*Estrutura do documento:*

1. *Funções de Texto*;
2. *Funções de Data;*
3. *Leitura/Escrita (I/O) de Arquivos;*
4. *Conversão de Tipos;*
5. *Automação Ambiente Gráfico;*
6. *Agendamento de tarefas;*
7. *Funções SISBB*;

# *Funções de Texto*

**concat( arg1, arg2, ... argN ):**

Concatena os argumentos em um texto (*string*) único.

string texto = concat( “João ”, “Ninguém” )

**contains( arg1, arg2 ):**

Verifica se o primeiro texto (arg1) contém o segundo (arg2), retornando *verdadeiro* ou *falso*.

string txt = “Algum texto”

if contains( txt, “texto” )

print( “A variável TXT contém a palavra (texto)!” )

endif

**containsics( arg1, arg2 ):**

Verifica se o primeiro texto (arg1) contém o segundo (arg2), ignorando se são letras maiúsculas ou minúsculas, retornando *verdadeiro* ou *falso*.

string txt = “Algum TEXto”

if containsics( txt, “texto” )

print( “A variável TXT contém a palavra (texto)!” )

endif

**eq( arg1, arg2 ):**

Compara o conteúdo de dois textos (*strings*) e retorna verdadeiro se forem iguais, ou falso caso contrário.

string txt = “12:00”

if eq( txt, “12:00” )

print( “Agora é meio dia!” )

endif

**eqics( arg1, arg2 ):**

Compara o conteúdo de dois textos (*strings*), ignorando letras maiúsculas ou minúsculas, retornando *verdadeiro* se forem iguais, ou *falso* caso contrário.

string txt1 = “poSITIvo”

string txt2 = “POSitivo”

if eqics( txt1, txt2 )

print( “Os dois textos são iguais!” )

endif

**extract( texto, posicao, tamanho ):**

Extrai uma parte do texto de acordo com a posição e o tamanho.

string texto = “algum texto sem sentido”

string parte = subs( texto, 12, 3 )

print( “O conteúdo de PARTE é (sem): “ + parte )

**findstr( arg1, arg2 ):**

Encontra o segundo texto no primeiro, retornando o número da posição da primeira letra (começando em 0), ou *-1* se não encontrar.

string txt = “algum texto”

int pos = findstr( txt, “texto” )

print( “TEXTO está na posição (6): “ + pos )

**findlast( arg1, arg2 ):**

Encontra o segundo texto no primeiro, retornando o número da **última** posição (começando em 0), ou *-1* se não encontrar.

string txt = “algum texto texto”

int pos = findlast( txt, “texto” )

print( “TEXTO está na posição (12): “ + pos )

**lower( arg ):**

Converte um texto (*string*) em letras minúsculas.

string txt = “LETRAS MAIÚSCULAS”

print( “Imprime minúsculas: “ + lower(txt) )

**print( arg ):**

Imprime/mostra no console de saída qualquer tipo de argumento <arg>.

int dez = 10

print( “O valor da variável DEZ é “ + dez )

**replace( texto, arg, rep ):**

Substitui todas as ocorrências de <arg> no texto por <rep>.

string cpf = “123.456.789-00”

cpf = replace( cpf, “.”, “” )

print( “CPF sem pontos (123456789-00): “ + cpf )

**str( arg ):**

Converte qualquer tipo de argumento <arg> em texto.

int dez = 10

string texto = str( dez )

**strsize( arg ):**

Retorna o tamanho de um texto (*string*) informado como argumento.

string txt = “Algum texto”

print( “O tamanho de TXT é (11): “+ strsize(txt) )

**upper( arg ):**

Converte um texto (*string*) em letras maiúsculas.

string txt = “letras minúsculas”

print( “Imprime maiúsculas: “ + upper(txt) )

# *Funções de Data*

**addday( date, nr ):**

Adianta a data <date> em <nr> dias, retornando a nova data.

object hoje = date(“31/03/2014”)

object amanha = addday( hoje, 1 )

print( “Amanha (01/04/2014): “+ amanha )

**addhour( date, nr ):**

Adianta as horas da data em <nr> horas, retornando a nova data.

object hoje = date(“31/03/2014 13:00:00”)

addhour( hoje, 1 )

print( “Hoje é (01/04/2014 14:00:00): “+ hoje )

**addmin( date, nr ):**

Adianta os minutos da data em <nr> minutos, retornando a nova data.

object hoje = date(“31/03/2014 13:00:00”)

addmin( hoje, 45 )

print( “Hoje é (01/04/2014 13:45:00): “+ hoje )

**addmon( date, nr ):**

Adianta os meses da data em <nr> meses, retornando a nova data.

object hoje = date(“31/03/2014 13:00:00”)

addmon( hoje, 2 )

print( “Hoje é (31/05/2014 13:00:00): “+ hoje )

**addsec( date, nr ):**

Adianta os segundos da data em <nr> segundos, retornando a nova data.

object hoje = date(“31/03/2014 13:00:00”)

addsec( hoje, 2 )

print( “Hoje é (31/03/2014 13:00:02): “+ hoje )

**addyear( date, nr ):**

Adianta os anos da data em <nr> anos, retornando a nova data.

object hoje = date(“31/03/2014 13:00:00”)

addyear( hoje, 2 )

print( “Hoje é (31/03/2016 13:00:00): “+ hoje )

**after( date1, date2 ):**

Compara duas datas, retornando verdadeiro se a primeira data for posterior à segunda, ou retorna falso caso contrário.

object hoje = date(“31/03/2014 13:00:00”)

object amanha = date(“01/04/2014”)

if after( amanha, hoje )

print(“Amanhã vem depois de hoje!”)

endif

**before( date1, date2 ):**

Compara duas datas, retornando verdadeiro se a primeira data for anterior à segunda, ou retorna falso caso contrário.

object hoje = date(“31/03/2014 13:00:00”)

object amanha = date(“01/04/2014”)

if before( hoje, amanha )

print(“Hoje vem antes de amanhã!”)

endif

**date( [text] ):**

Cria uma data a partir do argumento de texto informado. Se nenhum argumento for fornecido, retorna a data atual.

object hoje = date()

object amanha = date( “01/04/2014” )

object depois = date( “31/03/2014 18:00:00” )

**dayofw( date ):**

Retorna o dia da semana da data informada, sendo que 1 (um) representa Domingo e 7 (sete) representa Sábado.

object amanha = date( “01/04/2014” )

print(“Amanhã é terça (3): “+ dayofw( amanha ))

**diff( date1, date2 ):**

Retorna a diferença de tempo entre duas datas.

object hoje = date( “31/03/2014” )

object amanha = date( “01/04/2014” )

object diferenca = diff( hoje, amanha )

**eqdate( date1, date2 ):**

Compara se duas datas são iguais, ignorando as horas, retornando verdadeiro ou falso.

object hoje = date( “01/04/2014 15:00:00” )

object data = date( “01/04/2014” )

if eqdate( hoje, data )

print(“As duas datas são iguais, ignorando as horas”)

endif

**eqtime( date1, date2 ):**

Compara se duas datas são iguais, inclusive em horas, minutos e segundos, retornando verdadeiro ou falso.

object hoje = date( “01/04/2014 15:00:00” )

object data = date( “01/04/2014 15:00:00” )

if eqtime( hoje, data )

print(“As duas datas são iguais, inclusive as horas”)

endif

**getday( date ):**

Retorna o dia do mês da data informada.

object hoje = date( “01/04/2014” )

print( “Hoje é primeiro de abril (1): “+ getday( hoje ) )

**gethour( date ):**

Retorna a hora do dia da data informada.

object hoje = date( “01/04/2014” )

print( “Agora é meia noite (0): “+ gethour( hoje ) )

**getmin( date ):**

Retorna os minutos da hora da data informada.

object hoje = date( “01/04/2014” )

print( “Agora é meia noite em ponto (0): “+ getmin( hoje ) )

**getmon( date ):**

Retorna o mês do ano da data informada.

object hoje = date( “01/04/2014” )

print( “Estamos em abril (4): “+ getmon( hoje ) )

**getsec( date ):**

Retorna os segundos do minuto da data informada.

object hoje = date( “01/04/2014 00:00:02” )

print( “Agora é meia noite e dois segundos (2): “+ getsec( hoje ) )

**getyear( date ):**

Retorna o ano da data informada.

object hoje = date( “01/04/2014” )

print( “Estamos em 2014 (2014): “+ getyear( hoje ) )

**getyear( date ):**

Retorna o ano da data informada.

object hoje = date( “01/04/2014” )

print( “Estamos em 2014 (2014): “+ getyear( hoje ) )

**setday( date ):**

Define o dia do mês da data informada, retornando a.

object hoje = date( “01/04/2014” )

setday( hoje, 2 )

**sethour( date ):**

Define a hora do dia da data informada, retornando a.

object hoje = date( “01/04/2014” )

hoje = sethour( hoje, 15 )

**setmin( date ):**

Define os minutos da hora da data informada, retornando a.

object hoje = date( “01/04/2014” )

setmin( hoje, 45 )

**setmon( date ):**

Define o mês do ano da data informada, retornando a.

object hoje = date( “01/04/2014” )

hoje = setmon( hoje, 10 )

**setsec( date ):**

Define os segundos da hora da data informada, retornando a.

object hoje = date( “01/04/2014” )

hoje = setsec( hoje, 25 )

**setyear( date ):**

Define o ano da data informada, retornando a.

object hoje = date( “01/04/2014” )

hoje = setyear( hoje, 2015 )

# *Leitura/Escrita (I/O) de Arquivos*

Variáveis Globais:

Variáveis globais são variáveis que podem ser referenciadas em qualquer parte do script e possuem valores estáticos.

**CSV:** (CSV = “;”)

Armazena um caractere de divisão de colunas em arquivos de planilha com extensão CSV.

**EOF:** (EOF = “EOF”)

Armazena as letras “EOF” e representa o final de um arquivo, geralmente utilizada em funções de leitura de arquivos.

**LN:** (LN = “\n”)

Armazena um caractere de quebra de linha em arquivos de texto.

**SPACE:** (SPACE = “ ”)

Armazena um caractere de espaço em branco.

FUNÇÕES:

**frclose( arq ):**

Fecha um arquivo após sua leitura, onde o argumento recebido é o nome do arquivo.

string arquivo = “C:/dados.txt”

frclose( arquivo )

**fread( arq, [delim] ):**

Lê o conteúdo de texto do arquivo <arq> até encontrar o delimitador informado por [delim]. Se nenhum delimitador for fornecido, é assumido a quebra de linha LN como delimitador.

string arquivo = “C:/dados.txt”

string coluna = fread( arquivo, CSV )

**frbytes( arq, bytes ):**

Lê o conteúdo binário do arquivo <arq>. O segundo argumento <bytes> define a quantidade de bytes que deve ser lida. O valor retornado é um objeto que contém uma coleção com os bytes lidos.

string arquivo = “C:/compactado.zip”

object cont = frbytes( arquivo, 1000 )

**fwclose( arq ):**

Fecha um arquivo após escrita, onde o argumento recebido é o nome do arquivo.

string arquivo = “C:/dados.txt”

fwclose( arquivo )

**fwrite( arq, conteudo ):**

Escreve o conteúdo de texto no arquivo <arq>.

string arquivo = “C:/dados.txt”

string conteudo = “João Ninguém”

fwrite( arquivo, conteudo )

**fwbytes( arq, bytes ):**

Escreve o conteúdo binário do arquivo <arq>. O segundo argumento <bytes> define o conteúdo binário a ser escrito.

object cont = frbytes( arquivo, 1000 )

fwbytes( arquivo, cont )

**fwimage( arq, img ):**

Escreve uma imagem no arquivo <arq> em formato JPEG.

object imagem = screencapture( 0, 0, 800, 600 )

fwimage( “C:/tela.jpg”, imagem )

**length( bytes ):**

Retorna o tamanho de uma coleção de bytes lido de um arquivo.

object cont = frbytes( “C:/programa.exe”, 5000 )

print( “Bytes lidos (5000): “+ length( cont ) )

**syscommand( cmd ):**

Executa um comando no sistema operacional, retornando a saída do comando.

string comando = “cmd /c dir G:”

string saida = syscommand( comando )

print( saída )

“O volume na unidade G é New Dinop

O Número de Série do Volume é 8A19-F0E4

Pasta de G:\

04/06/2013 20:37 <DIR> .

04/06/2013 20:37 <DIR> ..

04/06/2013 19:47 <DIR> CONFIDENCIAL

20/02/2014 19:11 <DIR> INTERNA

04/06/2013 20:37 <DIR> PUBLICA

04/06/2013 20:37 <DIR> RESTRITA

0 arquivo(s) 0 bytes

6 pasta(s) 898.831.851.520 bytes disponíveis”

# *Conversão de Tipos*

**isnumber( arg ):**

Verifica se o argumento informado é um número, retornando verdadeiro ou falso.

double temp = 26.4

object nr = temp

if isnumber( nr )

print( “<nr> é um número!” )

endif

**toint( arg ):**

Converte um texto ou double em um número inteiro.

string dez = “10”

double temp = 26.4

int x = toint( dez )

print( “O valor de x é (10): “+ x )

x = toint( temp )

print( “O valor de x é (26): “+ x )

**todouble( arg ):**

Converte um texto ou inteiro em um número decimal.

string dez = “10”

int x = 26

double dec = todouble( dez )

print( “O valor de <dec> é (10): “+ dec )

dec = todouble( x )

print( “O valor de <dec> é (26.0): “+ dec )

# *Automação Ambiente Gráfico*

Variáveis Globais:

Variáveis globais são variáveis que podem ser referenciadas em qualquer parte do script e possuem valores estáticos.

**BUTTON1:**

Representa o botão esquerdo do mouse.

**BUTTON2:**

Representa o botão do meio (pressionado) do mouse.

**BUTTON3:**

Representa o botão direito do mouse.

**DLERROR:**

Representa um tipo de janela de erro.

**DLINFO:**

Representa um tipo de janela de informação.

**DLINPUT:**

Representa um tipo de janela de entrada de dados.

**DLQUESTION:**

Representa um tipo de janela de pergunta.

**DLWARNING:**

Representa um tipo de janela de alerta.

**NO:** (NO = “NO”)

Representa um resultado negativo.

**YES:** (YES = “YES”)

Representa um resultado positivo.

**SCREENH:**

Armazena a altura da tela do monitor do computador em pixels.

**SCREENW:**

Armazena a largura da tela do monitor do computador em pixels.

FUNÇÕES:

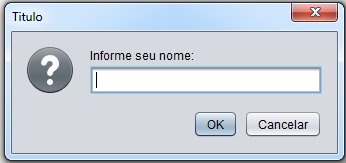
**dialog( tipo, titulo, msg ):**

Exibe uma janela de informações. O tipo de janela é definido pelas variáveis globais iniciadas por DL.

dialog( DLINFO, “Titulo da Janela”, “Processamento Concluido” )



dialog( DLINPUT, “Titulo”, “Informe o seu nome:” )



string resp = dialog( DLQUESTION, “Pergunta”, “Voce e maior de idade?” )

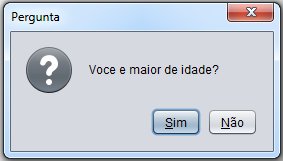
if eq( resp, YES )

print( “>= 18” )

else

print( “< 18” )

endif



**delay( milis ):**

Pausa a execução do script pelo tempo <milis> em milisegundos.

print( “Pausa por 2 segundos...” )

delay( 2000 )

**key( tecla ):**

Pressiona e solta uma tecla do teclado.

key( “S” )

key( “TAB” )

**keypress( tecla ):**

Pressiona uma tecla do teclado, que deverá ser liberada depois.

keypress( “CTRL” )

keypress( “C” )

keyrelease( “CTRL” )

keyrelease( “C” )

**keyrelease( tecla ):**

Libera uma tecla pressionada do teclado.

keypress( “CTRL” )

keypress( “C” )

keyrelease( “CTRL” )

keyrelease( “C” )

**mclick( button, x, y ):**

Causa um click do mouse no lugar especificado por <x> e <y> em pixels, com o botão <button> especificado.

mclick( BUTTON1, 700, 1000 )

**mdrag( x, y ):**

Arrasta o mouse de sua posição atual até o lugar especificado por <x> e <y> em pixels.

mdrag( 700, 1000 )

**mmove( x, y ):**

Move o mouse de sua posição atual até o lugar especificado por <x> e <y> em pixels.

mmove( 700, 1000 )

**mwheel( n ):**

Roda o botão do meio do mouse em <n> vezes.

mwheel( 3 )

**screencapture( x, y, larg, alt ):**

Captura a tela do monitor, iniciando no ponto definido por <x> e <y> e com a largura <larg> e altura <alt> em pixels.

print( “Captura a tela inteira” )

object img = screencapture(0, 0, SCREENW, SCREENH )

**writestr( texto ):**

Escreve o texto informado como se fosse digitado no teclado.

string texto = “Texto digitado”

writestr( texto )

# *Agendamento de tarefas*

**schedule( data, funcao ):**

Agenda a execução de uma função previamente definida.

object data = date( “01/04/2014 14:00:00” )

func imprime()

print( “A função agendada está executando!” )

endfunc

schedule( data, “imprime” )

# *Funções SISBB*

**append():**

Copia e anexa o texto selecionado no terminal, à um conteúdo já copiado anteriormente.

append()

**backspace():**

Apaga um caractere para a esquerda no terminal.

backspace()

**blinkstatus():**

Faz a barra de status do terminal piscar duas vezes.

blinkstatus()

**connect( [endereco], [porta] ):**

Conecta o terminal no servidor no [endereço] e [porta] especificados. Se não nenhum argumento for informado, conecta no servidor padrão Sisbb.

connect()

**copy():**

Copia o texto selecionado no terminal para a área de transferência do sistema operacional.

select( 1, 3, 10, 1 )

copy()

**cursor( lin, col ):**

Posiciona o cursor na posição especificada no terminal.

cursor( 1, 3 )

**cut():**

Recorta o texto selecionado no terminal para a área de transferência do sistema operacional.

select( 1, 3, 10, 1 )

cut()

**delay( milis ):**

Pausa a execução do script pelo tempo especificado em milisegundos.

print( “Pausa por 2 segundos...” )

delay( 2000 )

**delete():**

Deleta o caractere sob a posição do cursor no terminal.

delete()

**disconnect():**

Desconecta o terminal do servidor previamente conectado.

disconnect()

**down():**

Move o cursor uma linha para baixo.

down()

**enter():**

Pressiona a tecla enter no terminal, enviando as modificações na tela para o servidor.

enter()

**end():**

Pressiona a telca end no terminal, apagando o conteúdo de um campo.

end()

**f1():**

Pressiona a telca F1 no terminal.

f1()

**f2():**

Pressiona a telca F2 no terminal.

f2()

**f3():**

Pressiona a telca F3 no terminal.

f3()

**f4():**

Pressiona a telca F4 no terminal.

f4()

**f5():**

Pressiona a telca F5 no terminal.

f5()

**f6():**

Pressiona a telca F6 no terminal.

f6()

**f7():**

Pressiona a telca F7 no terminal.

f7()

**f8():**

Pressiona a telca F8 no terminal.

f8()

**f9():**

Pressiona a telca F9 no terminal.

f9()

**f10():**

Pressiona a telca F10 no terminal.

f10()

**f11():**

Pressiona a telca F11 no terminal.

f11()

**f12():**

Pressiona a telca F12 no terminal.

f12()

**getscreen():**

Retorna o texto exibido na tela do terminal.

string tela = getscreen()

**gettext( lin, col, comp ):**

Captura no terminal e retorna o texto posicionado na linha <lin> e coluna <col>, com o comprimento <comp> especificados.

string texto = gettext( 1, 3, 10 )

**left():**

Move o cursor uma posição para a esquerda.

left()

**paste():**

Cola o conteúdo da área de transferência na posição atual do cursor no terminal.

paste()

**pgdown():**

Pressiona a telca PAGE DOWN no terminal (mesmo que F8).

pgdown()

**pgup():**

Pressiona a telca PAGE UP no terminal (mesmo que F7).

pgup()

**right():**

Move o cursor uma posição para a direita.

right()

**select( lin, col, larg, alt):**

Seleciona uma parte da tela do terminal, iniciando em <lin> e <col>, com a largura de <larg> caracteres e <lin> linhas de altura.

select( 1, 3, 10, 1 )

**setpassword( lin, col, pass ):**

Define uma senha codificada em Base64 na posição indicada do terminal.

setpassword( 16, 14, “MTIzNDU2Nzg=” )

**settext( lin, col, text ):**

Define um texto <text> na posição indicada do terminal.

settext( 15, 14, “clientes” )

**status( text ):**

Define o conteúdo <text> da barra de status do terminal.

status( “Pesquisa Concluída” )

**tab():**

Pressiona a tecla TAB no terminal, movendo o cursor para o próximo campo desprotegido.

tab()

**up():**

Move o cursor uma linha para cima.

up()

**waitelse( lin, col, text, lin2, col2, text2 ):**

Aguarda a atualização do terminal e o texto <text> na posição indicada por <lin> e <col>. Caso o texto <text> não esteja definido na posição indicada após a atualização da terminal, aguarda o segundo texto <text2> na posição <lin2> e <col2>. Se nenhuma condição for atendida é gerado um erro.

A função retorna um dos textos encontrados.

string telaID = waitelse( 1, 2, “BB30”, 1, 3, “IBBM” )

**waitfor( lin, col, text ):**

Aguarda a atualização do terminal e o texto <text> na posição indicada por <lin> e <col>. Se a condição não for atendida é gerado um erro.

string telaID = waitelse( 1, 2, “BB30”, 1, 3, “IBBM” )