Detalhes básicos da linguagem FOCA

Primeiros Passos

A linguagem FOCA é "Case Sensitive"

A linguagem FOCA é "Case Sensitive", isto é, maiúsculas e minúsculas fazem diferença. Se declarar uma variável com o nome soma ela será diferente de Soma, SOMA, SoMa ou sOmA. Da mesma maneira, os comandos do FOCA if e for, por exemplo, só podem ser escritos em minúsculas pois senão o compilador não irá interpretá-los como sendo comandos, mas sim como variáveis.

Primeiro Programa

Vejamos um primeiro programa na linguagem FOCA:

```
int main ()
begin
    writeln "Hello world!”;
end
```

Compilando e executando este programa você verá a mensagem “Hello world” no terminal.

Vamos analisar o programa por partes.

A linha int main() indica que estamos definindo uma função de nome main. Todos os programas na linguagem FOCA têm que ter uma função main, pois é esta função que será chamada quando o programa for executado. O conteúdo da função é delimitado por begin e end.

A única coisa que o programa realmente faz é executar o comando writeln, que imprime um valor seguido de uma quebra de linha na tela.

Introdução às Funções

Abaixo o tipo mais simples de função:

```
function mensagem ()
begin
 writeln “Hello
 world!”;
end
int main()
begin
 mensagem();
end
```

Este programa terá o mesmo resultado que o primeiro exemplo da página anterior

- Argumentos

Os argumentos são passados dizendo o tipo e a variável. Podem ser passados mais de um agrumento, bastando usar o separador “,” para cada argumento.

- Retornando valores

Uma função só pode retornar um valor. Este valor deve ser do mesmo tipo que a função foi declarada. O formato do retorno é: return val; onde val é um valor do mesmo tipo da função, este pode ser uma variável, uma expressão ou um valor explícito.

Forma geral

Apresentamos aqui a forma geral de uma função:

```
function tipo_de_retorno nome_da_função (lista_de_argumentos)
begin
   código_da_função
end
```

Para fazer a chamada de uma função, se esta função tiver um tipo, ela pode ser atribuida à uma variável daquele tipo. Por exemplo:

```
function int foo ( )
begin
   código_da_função
end
int main()
begin
   int a = foo( );
end
```

Se esta função não tiver um tipo ou o objetivo não é atribuí-la à uma variável daquele tipo, ela deve ser chamada com o prefixo func. Por exemplo:

```
int main()
begin
   func foo( );
end
```

Introdução Básica às Entradas e Saídas

O comando de entrada é o comando read. Este deve ser usado da seguinte forma: read variavel; Não importa qual o tipo que está sendo lido, para todos os tipos o formato é o mesmo.

O comando de saída, como vimos na introdução, é o write. Este comando pode também ser escrito como writeln, se desejar uma quebra de linha após a impressão dos valores. É possível imprimir vários valores concatenados da seguinte forma: write valor_1, valor_2, …, valor_n; Ou writeln valor_1, valor_2, …, valor_n;

Introdução a Alguns Comandos de Controle de Fluxo

- if

A sua forma geral é:

```
if (condição)
begin
    comandos
end;
```

A condição do comando if é uma expressão que será avaliada. Seu resultado deve ser do tipo boleano. Se o resultado for falso os comandos não serão executados.

Os operadores de comparação são: == (igual), != (diferente de), > (maior que), < (menor que), >= (maior ou igual), <= (menor ou igual).

- if/else

A sua forma geral é:

```
if (condição)
begin
    comandos
end
else
    comandos
end;
```

Se a condição for falsa, os comandos do bloco else são executados.

- if/elif

A sua forma geral é:

```
if (condição1)
begin
    comandos
end
elif(condição2)
    comandos
end;
```

Se a condição1 for falsa, é testada a condição2. Esta por sua vez, se for verdadeira, os comandos do bloco elif são executados, caso contrário, não seráo executados.

- if/elif/else

A sua forma geral é:

```
if (condição1)
begin
    comandos
end
elif(condição2)
    comandos
end
else
    comandos
end;
```

Se a condição1 for falsa, é testada a condição2. Esta por sua vez, se for verdadeira, os comandos do bloco elif são executados, caso contrário, os comandos do bloco else serão executados.

- for

Sua forma geral é:

```
for (inicialização;condição;incremento)
begin
   comandos
end;
```

A inicialização também pode ser uma declaração. A condição deve ter como resultado um valor boleano. O incremento pode ser na forma de expressão comum, ou atribuição unária: ++i;

- while

Sua forma geral é:

```
while (condição)
begin
   comandos
end;
```

A condição deve ter como resultado um valor boleano.

- do while

Sua forma geral é:

```
do
begin
   comandos
end
while (condição) ;
```

A condição deve ter como resultado um valor boleano.

Comentários

O compilador FOCA desconsidera qualquer coisa que esteja começando com /* e terminando com */. Sendo este o comentário multilinha. Já o comentário de apenas uma linha inicia com // e vai até a primeira quebra de linha.

Palavras Reservadas da linguagem FOCA

Como a linguagem FOCA é "case sensitive" podemos declarar uma variável For, apesar de haver uma palavra reservada for.

Apresentamos a seguir as palavras reservadas da linguagem FOCA.

int	function	do	read
float	func	if	continue
char	begin	elif	superContinue
string	end	else	break
boolean	for	write	superBreak
main	while	writeln	global

Nomes de Variáveis

As variáveis na linguagem FOCA podem ter qualquer nome se três condições forem satisfeitas: o nome deve começar com uma letra ou sublinhado “_” e os caracteres subsequentes devem ser letras, números ou sublinhado “_”; o nome de uma variável não pode ser igual a uma palavra reservada.

Os Tipos da linguagem FOCA

A linguagem FOCA tem 5 tipos básicos: char, int, float, boolean e string.

Declaração e Inicialização de Variáveis

As variáveis na linguagem FOCA devem ser declaradas antes de serem usadas. A forma geral da declaração de variáveis é:

```
tipo_da_variável variável; //ou
tipo_da_variável variável = valor; //ou
tipo_da_variável lista_de_variáveis;
```

As variáveis da lista de variáveis terão todas o mesmo tipo e deverão ser separadas por vírgula.

Há três lugares nos quais podemos declarar variáveis. O primeiro é fora de todas as funções do programa, nas primeiras linhas. Estas são as variáveis globais. São declaradas da seguinte forma:

```
global tipo_da_variável variável;
```

Quando for fazer referência à uma variável global, deve-se usar o modificador global, assim:

```
global variável = valor;
```

O segundo lugar no qual se pode declarar variáveis é dentro de um bloco de código, delimitado por begin e end. Estas variáveis são chamadas locais e só têm validade dentro do bloco no qual são declaradas.

O terceiro lugar onde se pode declarar variáveis é na lista de parâmetros de uma função. Apesar de estas variáveis receberem valores externos, estas variáveis são conhecidas apenas pela função onde são declaradas.

Quando uma variável não é encontrada em um bloco, ela é buscada no bloco pai, sucessivamente, até encontrar aquela variável, ou, caso não encontre, da erro de compilação.

Quando uma variável é declarada sem atribuição, ela é inicializada com um valof default.

Operadores Aritméticos e de Atribuição

Os operadores aritméticos são usados para desenvolver operações matemáticas. A seguir apresentamos a lista dos operadores aritméticos do C:

Operador	
+	Soma ou concatenação, para string
-	Subtração ou Troca de sinal
*	Multiplicação
/	Divisão
++	Incremento
--	Decremento

Os operadores de incremento e decremento são unários que alteram a variável sobre a qual estão aplicados. O que eles fazem é incrementar ou decrementar, a variável sobre a qual estão aplicados, de 1. Então

```
++x; --x;
```

são equivalentes a

```
x=x+1;
x=x-1;
```

Estes operadores devem ser pré-fixados.

Operadores Relacionais e Lógicos

Os operadores relacionais da linguagem FOCA realizam comparações entre variáveis.
São eles:

Operador	
>	Maior do que
>=	Maior ou igual a
<	Menor do que
<=	Menor ou igual a
==	Igual a
!=	Diferente de

Os operadores relacionais retornam verdadeiro (true) ou falso (false).

Para fazer operações com valores lógicos (verdadeiro e falso) temos os operadores lógicos:

Operador	
and	AND (E)
or	OR (OU)
!	NOT (NÃO)

Expressões

- Conversão de tipos em expressões

Quando a linguagem FOCA avalia expressões onde temos variáveis de tipos diferentes o compilador verifica se as conversões são possíveis. Se não são, ele não compilará o programa, dando uma mensagem de erro. Se as conversões forem possíveis ele as faz.

- Modeladores (Casts)

Um modelador é aplicado a uma expressão ou um valor. Ele força este a ser de um tipo especificado. Sua forma geral é:

```
(tipo)expressão // ou
(tipo)valor
```

O Comando break

O comando break faz com que a execução do programa continue na primeira linha seguinte ao fim do loop que está sendo interrompido.

O Comando superBreak

O comando superBreak faz com que a execução do programa continue na primeira linha seguinte ao fim do maior loop.

O Comando continue

O comando continue faz com que a execução do programa continue na primeira linha seguinte ao inicio do loop que está sendo interrompido.

O Comando superContinue

O comando superContinue faz com que a execução do programa continue na primeira linha seguinte ao inicio do maior loop.

Para se declarar um vetor podemos utilizar a seguinte forma geral:

```
tipo_da_variável nome_da_variável [tamanho];
```

Uma matriz funciona de forma análoga ao vetor. Pode ser declarada da seguinte forma:

```
tipo_da_variável nome_da_variável [tamanho1][tamanho2];
```

Para matrizes multidimensionais, basta adicionar um novo tamanho [tamanho3]... [tamanhoN].

Um vetor ou matriz deve não pode ter uma declaração com atribbuição. Deve ser declarado, para depois ter seus valores atribuidos.

Fatias

Uma fatia de um vetor ou matriz é obtido da seguinte maneira:

```
tipo a [tamanho1][tamanho2];
tipo b = a[0:N, 0:M];
```

Neste caso, b é uma fatia da matriz a. b contém os elementos de 0 até N das linhas e de 0 até M das colunas de a. Esta operação pode ser feita com matrizes de qualquer dimensão. A única restrição é que a operação só pode ser feita com variáveis dos mesmos tipos.

Strings

Alguns operadores para este tipo são sobrecarregados:

+ → Concatenação;
Operadores relacionais → Realizam comparações entre as strings.

Exemplos:

```
string a = “Hello ”;
string b = “world!”;
string c = a + b;

//Resultado: c == Hello world.

boolean d = a == b;

//Resultado: d == false.
```

Instruções para compilação

Acessar a pasta raiz do projeto (“Compiladores”). Executar o comando ./bin/glf < “caminho do arquivo .foca”.

É gerado na pasta raiz do projeto o arquivo no código intermediário e este já compilado.