Universidade Federal da Fronteira Sul - UFFS

Interpretador da linguagem de programação OQUE

Alunos: Cleiton A. Ambrosini
Marco Aurélio Alves Puton

Chapecó, Maio de 2015.

Linguagem de programação OQUE

A linguagem de programação OQUE foi desenvolvida para a implementação de um interpretador, utilizando a linguagem java.

Funcionalidades da linguagem OQUE:

- Declaração de variáveis;
- Atribuições;
- Operações aritméticas com expressões com 1 ou mais operandos;
- Laço;
- Comando de entrada;
- Comando de saída;
- Controlador de fluxo;
- Aninhamento de comandos;

Sintaxe:

Sintaxe geral:

A linguagem OQUE é super-flexível, ou seja, permite ao usuário a liberdade de colocar espaços, linhas em branco e TABs entre instruções, e também entre palavras. Para deixar mais claro, vejamos o exemplo de quatro linhas que fazem exatamente a mesma coisa:

```
linha=2&2^4;
linha = 2 & 2 ^ 4;
linha    = 2  &2              ^    4;
li       nh      a=2 &  2^        4      ;
```

OQUE não possui precedência de operações, deixando a responsabilidade para o programador fazer expressões na sequência correta;

O caracter ‘;’ é utilizado para delimitar instruções, exceto SEs e SENAOs (geralmente IFs e ELSEs de outras linguagens).

Tokens:

Caracteres utilizados na linguagem:

‘**&**’ -----> Soma
‘**^** ’ ----->Subtração
‘**~**’ -----> Multiplicação
‘**$**' -----> Divisão
‘**#**’ -----> Resto inteiro da divisão
‘{’ -----> Abre escopo
‘}’ -----> Fecha escopo
‘;’ ----->Término de instrução
‘@’ -----> Comparação de igualdade
’!’ -----> Comparação de desigualdade
‘<’ -----> Comparação de menor
‘>’ -----> Comparação de maior
‘=’ -----> Atribuição

Declaração e atribuição de variáveis:

Para a declaração de variáveis, OQUE não necessita que o programador especifique o tipo que a variável irá receber. Quando a variável receber seu valor, a mesma passa a ser do tipo recebido, e pelo resto da execução do programa será do mesmo tipo. A linguagem suporta 3 tipos de variáveis: Inteiro, double e string. Para atribuir um valor, usa-se o caractere ‘=’.

Ex.:

```
a = 13;
b = 3.1415;
c = “um texto qualquer 123\n”;
```

Operações Aritméticas:

Operações aritméticas são feitas em atribuições. O programador pode colocar quantas operações quiser na mesma instrução. *Lembrando que não ocorre precedência de operações. Operações de soma envolvendo string resultam em valores do tipo string.

Ex.:

```
nome = “Fulano” + “ da Silva tem ” + idade + “ anos.\n”;
areaC = 5 ~ 5 ~ 3.14;
base = 5;
altura = base&3;
```

```
areaT = base ~ altura $ 2;
```

Controlador de fluxo:

Para controlador de fluxo, utiliza-se o SE e o SENAO. Assim como outros controladores de fluxo, SE necessita de um condicional para executar seu escopo. O condicional deve conter um teste com exatamente 2 operandos. Se o condicional for verdadeiro, o escopo será executado, se for falso e se houver eu SENAO em seguida, o SENAO será executado.

Ex.:

```
a = 2;
SE(a@2)
{
	b = 2~a;
}

SE(b@a)
{
	b = b ^ a;
}
SENAO
{
	b = b & a;
}
```

Laço:

O laço de repetição implementado é o repetix. Ele funciona da mesma forma que o while da linguagem C, por exemplo. Após escrever repetix, coloca-se um condicional e em seguida um escopo a ser executado e após fechar o escopo, o caracter ‘;’.

Ex.:

```
a = 0;
condicao = 1;
repetix(condicao@1)
{
	a = a & 1;
	SE(a@5)
	{
		condicao = 0;
	}
};
```

Comando de entrada:

O comando de entrada é genérico, funcionando o mesmo para os três tipos de variáveis da linguagem. Ao ler, o valor lido é colocado em uma variável. Primeiro coloca-se a variável que vai armazenar o valor e em seguida “[;” para representar uma leitura.

Ex.:

```
a[;
```

*Neste exemplo, a recebe o valor lido.

Comando de saída:

O comando de saída é o IMPRIME. Ele recebe como parâmetro, apenas um valor a ser impresso, não aceitando operações dentro dele.

Ex.:

```
a = 1;
texto = “o valor de a é ” & a & “\n”;
IMPRIME(texto);
```

Aninhamento:

O aninhamento funciona tanto para para SEs e SENAOs como para repetix.