UNIVERSIDADE FEDERAL FRONTEIRA SUL
CIÊNCIA DA COMPUTAÇÃO – 3º Período
ALUNOS: Emerson Martins, Leonardo Vargas
DATA: 15/05/15

TRABALHO DE PROGRAMAÇÃO I :

Implementar um programa em java que interprete uma linguagem de programação criada pelos alunos. O interpretador deve ser capaz de analisar e reagir corretamente as seguintes situações no arquivo fonte que ele esteja interpretando:

- ✪ Declaração de variáveis;
- ✪ Atribuição de valor a variável;
- ✪ Expressões com no minimo dois operandos;
- ✪ Operações de adição, subtração, divisão e multiplicação;
- ✪ Laço;
- ✪ Comando de saída (ex: Mostrar algo na tela);
- ✪ Controlador de fluxo (ex: If);
- ✪ Aninhamento de comandos (ex: ifs dentro de ifs, laços dentro de laços).

Sobre a linguagem:

A linguagem chama-se **Toon World** e é definida e especificada pelos seguintes comandos:

Arquivo Especifico:

Para a utilização do interpretador é necessário um arquivo na forma “Arquivo.toon”.

Inicio e fim do programa:

Não é necessário ter nenhum comando especifico de começo ou fim do programa, somente o código.

Uso do terminador:

O único caractere obrigatório nos comandos é o terminador. Para funções diferentes de laços e condições (será explicado posteriormente) o terminador é o simbolo “?”.

★ Exemplo de programa:

“código” ?
“código” ?

Declaração de variáveis:

A linguagem suporta os tipos Int, Double e String. No momento da declaração é permitida a atribuição de valores usando o simbolo “<)” depois do nome da variável. É possível atribuir um valor usando uma operação aritmética. No caso de Strings são usadas “”(aspas) para a atribuição. Para declarar uma variável sem definir seu valor é necessário apenas usar o terminador após o nome da mesma.

★ Exemplo de programa:

Int A <) 10?
Int B <) 12*2?
Double C?
String J <) “sou uma string”?

Principais operadores:

Os principais operadores são + (soma), - (subtração), / (divisão), * (multiplicação) e % (Mod). Não é possível fazer mais de uma operação por vez.

Existem ainda outras operações como:

<) Atribuição, atribui à uma variável o valor desejado.
<< Menor, compara números ou variáveis e retorna verdadeiro ou falso.
>> Maior, compara números ou variáveis e retorna verdadeiro ou falso.
|=| Igual, compara números ou variáveis e retorna verdadeiro ou falso.
=|= Diferente, compara números ou variáveis e retorna verdadeiro ou falso.
>| Maior ou Igual, compara números ou variáveis e retorna verdadeiro ou falso.
<| Menor ou Igual, compara números ou variáveis e retorna verdadeiro ou falso.
? Terminador das linhas que não definem inicio ou fim de função If, four e while.
{ Inicio de expressão,utiliza a mesma da ideia de “(” em outras linguagens.
} Fim de expressão,utiliza a mesma da ideia de “)” em outras linguagens.
[Abre escopo, define o inicio de uma função if, four ou while.
] Fecha escopo, define o fim de uma função if, four ou while.

Função If:

Funciona da seguinte maneira:
“if{} [” dentro das chavetas vai a condição a ser analisada. Retorna verdadeiro ou falso.
Há também a possibilidade de usar um “] else [”. A cada fechamento de escopo é necessário colocar um i(if) ou e(else) ao lado da chaveta.

★ Exemplo de programas:

```
if{A<<B}[          se A é menor que B
        “código”
]i

if{A>>B}[          se A é maior que B
        “código”
]i

if{A|=|B}[         se A é igual a B
        “código”
]i

if{A=|=B}[         se A é diferente de B
        “código”
]i else [
        “código”
]e
```

```
if{A>|B}[               se A é maior ou igual a B
        “código”
]i else [
        “código”
]e

if{A<|B}[               se A é menor ou igual a B
        “código”
]i else [
        “código”
]e
```

Função four

Funciona da seguinte forma:
“four{}[” Dentro das chavetas é necessário ter 3 “comandos”. O 1º de atribuição a variável, o 2º de condição de execução (do mesmo modo da função if) e o 3º também de atribuição de variável (var++ ou var--).

Para separar esses comandos é necessário o uso do terminador.

```
#       Atribuição de variável no 3º comando:
var...?         Tem a mesma ideia de var++ em outras linguagens.
Var,,,?         Tem a mesma ideia de var-- em outras linguagens.
```

★ Exemplo de programa:

```
Int A?
four{a<)0? A<<1? a...?}[
        “código”
]f
```

Função While

Funciona da seguinte forma:
“while{cond}[” Dentro das chavetas estará a condição a ser analisada (do mesmo modo da função if). Retorna verdadeiro ou falso.

★ Exemplo de programa:

```
Int A <) 0?
while{A<<10}[
        “código”
        “código”
]w
```

Função print

Essa função tem a mesma ideia do comando System.out.println() em Java. Funciona de 2 maneiras:
“print{}?” Imprime o que esta dentro das chavetas sem quebrar linha.
“println{}?” Imprime o que esta dentro das chavetas quebrando a próxima linha.

Dentro das chavetas também existem 2 maneiras de funcionamento:
“print{“sou um teste”}?” Imprime o que estiver entre as “”(aspas) exatamente como estiver escrito.
“print{“Valor de A: ” + A}?” Imprime o texto seguido do valor de A. Pode ser feito quantas vezes forem necessárias.

★ Exemplo de programa:

```
Int A <) 5?

println{“oi ” + A + “você”}?        Imprime: oi 5voce (e quebra a próxima linha)
print{A}?                           Imprime: 5
```

Função Ler

Essa função tem a mesma ideia de scanf em C. Ela pode ser usada para as variáveis de tipo Int e Double. Funciona da seguinte maneira:
“ler(#NomeVariavel)?” .

★ Exemplo de programa:

```
Int A?

ler(#A)?            valor sera atribuído a variável A.
```

Linguagem Flexível

É possível utilizar sintaxe flexível no código desde que o comando inteiro fique na mesma linha. Isso torna a identação possível fazendo com que a compreensão do programa se torne mais fácil.

★ Exemplo de programa:

```
Int A <) 0?
Int B <) 5?
while{A<<B}[
        A...?
        if{A |=| 2}[
                print{“Entrou no if”}?
        ]i
]w
```