A Linguagem de

Programação Factum

Caio Furtado

Douglas Cavalcanti

Gabriel Ferreira

Lucas Pereira

2020

C A P I T U L O1

Introdução

**O** Factum é uma linguagem de âmbito educacional, feita no paradigma procedural. Está linguagem tem como referência a linguagem C. Seu propósito é facilitar o estudante de computação desenvolver a capacidade de fazer a descrição de um algoritmo em uma linguagem de programação.

A linguagem de programação Factum possui tipos fortes de dados e estáticos. Com a finalidade de no tempo de compilação ser possível encontrar erros básicos, que o iniciante pode causar, que abrangem sua sintaxe ou a utilização de alguma variável supérflua para o programa. O processo de compilação consiste em traduzir para um formato de instruções binarias, baseados em pilha, em que denominamos de *webassembly*.

A linguagem não permite ao programador alocar memória manualmente por não possuir representação de máquina, contrapondo o que acontece na linguagem C, a qual possui *maloc* e *free* - por exemplo - a fim de evitar o programador pensar na lapidação de bits e somente se focar na parte lógica do programa . Devido a isto, a linguagem possui um *garbage collertor* para gerenciar a memória de execução.

O Factum possui capacidade para criar métodos, tipos abstratos e tipos enumerados. Além de possuir polimorfismo estático possibilitando criação de diversos métodos sobrecarregados.

Como a linguagem Factum é educacional os tipos primitivos são baseados em conjuntos matemáticos, sendo assim o nome do tipo é o nome do conjunto, por exemplo Inteiro, Real, além de possuir o tipos *símbolo e cadeia* e do tipo lógico binário *boolean.*

**Manual de Referencias da Linguagem Factum**

**Capítulo 1 – Introdução ao Tutorial**

Esse tutorial tem o intuito de mostrar o mais básico da linguagem Factum, a fim de facilitar o aprendizado e possibilitar os iniciantes da área da computação a começarem a programar. Como esse tutorial não possui o objetivo de aprofundar o conhecimento na linguagem proposta, nos focaremos em variáveis estáticas, constantes, aritmética, fluxo de controle, funções e rudimentos de entrada e saída. Para iniciantes é recomendado aprender pelo tutorial, com o objetivo de aprender o funcionamento da linguagem aos poucos e evitar possíveis enganos. Em contrapartida para os programadores mais experientes recomenda-se pular o primeiro capítulo.

**1.1 Começando**

Para começarmos a aprender a linguagem Factum iremos fazer um programa simples, o qual é normalmente desenvolvido pelo menos uma vez na vida de qualquer programador…

**O “Hello Word”**!

Exemplo 1.

1 INICIO vazio main();

2 imprima(“Hello World!”);

3 FIM;

Resultado no console :Hello World!

Para entender este programa teremos que analisá-lo por partes. Na linha 1 temos “INICIO vazio Hello\_World();” , em que o “INICIO” significa de onde o método deve começar e na linha 3 temos “FIM Hello\_World();”, na qual “FIM” determina o fim do método. Por fim, temos o “;” que serve para indicar o fim de uma instrução. Além disso temos o tipo do método, nesse caso ele é do tipo “vazio”, que significa que a função não retorna nenhum valor, porém podemos utilizar outros tipos de dados como: dos inteiros, reais, entre outros. Assim como o exemplo 2:

Exemplo 2.

INICIO inteiro main();

(...)

retornar(0);

FIM;

O método main do exemplo acima é do tipo inteiro, por consequência, necessita do valor de retorno compatível com mesmo, bem como outros tipos primitivos ou tipos abstratos. Em síntese o método main pode ser declarado em diferentes tipos dependendo da aplicação feita.

Por último temos a instrução “imprima(“Hello World”);”, que possui a finalidade de imprimir uma mensagem no console, neste caso a mensagem “Hello World”. Para efetuar o comando corretamente, a mensagem precisa estar entre parênteses e aspas. Na instrução imprima, é possível representar de outra forma, assim como no exemplo a seguir:

Exemplo 3.

1 INICIO vazio main();

2 imprima(“Hello ”);

3 imprima(“World”);

4 imprima(“!”);

5 FIM;

Resultado no console :Hello World!

Com esse último exemplo podemos perceber que mesmo com programas escritos de forma diferente podemos reproduzir o mesmo resultado. Além disso, a instrução imprima possui outros comandos, como o “\n” que quebra a linha e o “\t” que executa o comando “tabela”, ou seja, 4 espaços em branco. O exemplo 4 a seguir demonstra isso:

Exemplo 4

1 INICIO vazio main();

2 imprima(“Hello \nWorld\t!”);

3 FIM;

Resultado no console :Hello

World !

**1.1** **Expressões aritméticas**

Para aprendermos a utilizar qualquer linguagem de programação temos que aprender como é representado as operações aritméticas. Para isso, vamos fazer um programa de exemplo para entendermos melhor, sabendo que o caractere “+” é para soma, “-” é para subtração, “\*” é para multiplicação e “/” para divisão. Além de que os tipos primitivos são:

|  |  |
| --- | --- |
| Inteiro | Representa o conjunto dos números inteiros |
| Real | Representa o conjunto dos números reais |
| Símbolo | Representa o conjunto do alfabeto ASCII utilizado na língua Factum |
| Cadeia | Cadeia de símbolos do alfabeto ASCII |

Exemplo 5

1 INICIO vazio main();

2 inteiro x;

3 inteiro y;

4 inteiro z;

5 x = 5;

6 y = 2;

7 z = x + y;

8 imprima(“Valor de z = ” +z);

5 FIM;

Resultado no console :Valor de z = 7

No exemplo 5 podemos ver uma nova utilidade para o comando imprima, a qual é utilizada para imprimir o valor de z. Além de demonstramos como é feita a declaração de uma variável, representada nas linhas 2, 3 e 4. Por outro lado, temos as atribuições de dados feitas pelo caractere “=”, como no caso das linhas 5 e 6. Por consequência “x” terá o valor 5 e y terá o valor 2. Devido a isto, a atribuição de x + y em z resultou em 7, a qual foi imprimido no console. Ademais tipos de dados primitivos podem ser declaradas da mesma forma, como representado.

Há também a compatibilidade de alguns tipos de dados, apesar da possível perda de dados, como de real para inteiro, acarretando a perda de casas decimais. No exemplo 6 podemos ver isso como mais clareza:

Exemplo 5

1 INICIO vazio main();

11 inteiro x;

12 real y;

13 inteiro z;

14 x = 5;

15 y = 2.5;

16 z = x + y;

17 imprima(“Valor de z = ” +z);

5 FIM;

Resultado no console :Valor de z = 7

Sabendo-se que 5 + 2.5 = 7.5, logo vemos que existe perda de dados nesse caso, mas não há perda de dados de inteiro para real.

**Apêndice A**

**A.1 Introdução**

Essa seção tem como objetivo especificar o funcionamento dos mecanismos da linguagem Factum, a fim de esclarecer o máximo possível sobre a língua.

**A.2 Conversões Lexicais**

Em Factum o compilador começa a ler a partir do comando “INICIO”, de maneira independente do tamanho da aplicação. A função main() é a última função a ser traduzida.

**A.2.1 Tokens**

Existem seis classes de tokens: identificadores, palavras-chave, constantes, operadores, e outros separadores. Os tokens são reconhecidos pela sequência de caracteres, ou seja, pela cadeia de símbolos. Por outro lado, o espaço seve para a separação das cadeias utilizadas, com essa propriedade quando se tem mais de um espaço adjacente em sequência, os espaços são diminuídos para um único.

**A.2.2 Comentários**

Para se fazer comentários em uma aplicação utiliza-se “//” para delimitar a parcela da direita da linha a partir do uso do “//”. Em somatória tem como deixar na forma de comentário mais de uma linha utilizando “/\*” e terminando com “\*/”.

**A.2.4 Palavras-Chave**

As seguintes palavras-chave são utilizadas na linguagem factum.

|  |  |  |
| --- | --- | --- |
| **simbolo** | **real** | **se** |
| **vazio** | **inteiro** | **escolha** |
| **senao** | **Faz** | **enquanto** |
| **para** | **cadeia** |  |

**A.2.5 Constantes**

Existem constantes nos tipos inteiros, carácter e flutuante.

O tipo primitivo real é utilizado para representar os números do conjunto dos reais, porém como o conjunto dos reais possui infinitos elementos e o computador possui memoria limitada, isto implica que o tipo real possui limite superior e limite inferior, além de arredondar números irracionais e racionais, sendo um subconjunto do conjunto dos Reais.

O tipo primitivo inteiro é utilizado para representação do conjunto dos inteiros, sendo limitado com um maior positivo e menor negativo.

O tipo primitivo simbolo é utilizado para representar símbolos do alfabeto ASCII. Sendo possível o armazenamento estático dos caracteres pertencentes ao alfabeto ASCII.

**A.3 Notação sintática**

Neste documento para se representar a sintaxe de palavras pertencentes a linguagem Factum, usamos a fonte de texto Courier New.

**A.3.1 Operações Aritméticas**

As operações aritméticas nessa língua de programação são escassas, assim não há suporte para operações que não sejam de soma, subtração, divisão ou multiplicação. Para os tipos numéricos contém as quatro operações possíveis. Já para o tipo símbolo ou cadeia só pode ser utilizado com o operador de soma, a qual executa uma concatenação de cadeias.

**A.4 Operadores**

**A.4.1 Operador de Igualdade**

O operador de igualdade, “=”, deve ser utilizado da seguinte maneira:

z = x;

Desta forma, pode-se ver que o novo valor de z será igual idêntico ao valor de x.

**A.4.2 Operador de Soma**

O operador de soma, “+”, deve ser utilizado da seguinte maneira

z = x + y;

Assim, o resultado da soma entre os dois símbolos nas laterais de + é armazenado na variável z.

**A.4.3 Operador de Divisão**

O operador de divisão, “/”

z = x / y;

O resultado da divisão do valor ou variável à esquerda de / pelo valor à direita do símbolo é armazenado na variável z.

OBS: O compilador não aceita o denominador ser zero por aberração matemática e com isso um erro é demonstrado em tempo de compilação.

**A.4.4 Operador de Multiplicação**

O operador de multiplicação, “\*”

z = x \* y;

O resultado da multiplicação entre os dois símbolos nas laterais de \* é armazenado na variável z.

**A.4.5 Operador de Subtração**

O operador de subtração, “-”

z = x - y;

O resultado da subtração do número à direita do símbolo “-” do valor da variável ou valor à esquerda dele.

**A.4.6 Operador de Resto**

O operador de resto, “@”

z = x @ y;

O resto do resultado da divisão de x por y é armazenado na variável z.

Referencias

“The C Programming Language” de Brian W. Kernighan e Dnnis M. Ritchie