

**UNIVERSIDADE FEDERAL DE ALAGOAS**  
**Instituto de Computação**  
**Bacharelado em Ciência da Computação**

**Linguagem FiatTipo - Especificações**

Matheus Gomes de Oliveira

**Junqueiro - AL, 2021**

## Sumário

<b>Sumário</b>	<b>1</b>
<b>1. Introdução.</b>	<b>2</b>
<b>2. Estrutura Geral do Programa.</b>	<b>3</b>
<b>3. Conjuntos de Tipo de Dados e Nomes.</b>	<b>4</b>
3.1 Palavras Reservadas.	5
3.2 Identificador.	5
3.3 Comentário.	5
3.4 Inteiro.	5
3.5 Ponto Flutuante.	5
3.6 Caracteres.	6
3.7 Cadeia de Caracteres.	6
3.8 Booleanos.	6
3.9 Arranjo Unidimensional.	6
3.10 Operações Suportadas	6
3.11 Valores Default.	7
3.12 Coerção	7
<b>4. Conjuntos de Operações</b>	<b>7</b>
4.1 Aritméticos	7
4.2 Relacionais	8
4.3 Lógicos	8
4.4 Concatenação de Cadeia de Caracteres	8
4.5 Precedência e Associatividade	8
4.5.1 Operadores multiplicativos e aditivos	9
4.5.2 Operadores Comparativos e igualdade	9
4.5.3 Operadores de negação e conjuração.	9
<b>5. Instruções.</b>	<b>9</b>
5.1 Atribuição.	9
5.2 Estrutura condicional de uma ou duas vias.	10
5.2.1 Se e Porem.	10
5.3 Estruturas Iterativas.	10
5.3.1 Controle Lógico - Enquanto.	10
5.3.2 Controle por Contador - Repita.	11
5.4 Entrada e Saída.	11
5.4.1 Entrada.	11
5.4.2 Saída.	12
5.5 Funções.	12
<b>6.Exemplos de Algoritmos.</b>	<b>13</b>
6.1 Hello World.	13
6.2 Série de Fibonacci.	13
6.3 Shell Sort.	13

## 1. Introdução.

A linguagem de programação FiatTipo tem como objetivo ser uma linguagem estaticamente tipada, não orientada a objetos e tendo como base a escritabilidade de linguagens como C e Pascal. Sua finalidade é ser uma linguagem de aprendizado e de simples compreensão de comandos e leitura de códigos para leigos.

FiatTipo, como já foi dito, é uma linguagem estaticamente tipada, ou seja, não admite coerção. Por não possuir coerção, trata-se de uma linguagem com uma maior confiabilidade em relação às que permitem coerção, porém por ela ser estaticamente tipada, não existe nenhum tratamento para erros de detecção de tipo. Sua estrutura é simples, não possuindo estruturas em blocos pois não permite criação de funções dentro de outras funções. A linguagem possui palavras reservadas que são de fácil identificação pelo usuário, tornando assim, a linguagem mais legível. A criação de um bloco é denominado pelas palavras reservadas INICIO e FIM.

## 2. Estrutura Geral do Programa.

Um programa em FiatTipo basicamente se escreve da seguinte forma:

- Para iniciar e fechar uma função/bloco em FiatTipo deve-se usar as palavras reservadas **INICIO** (abre) e **FIM**(fecha).
- A função principal é declarada usando-se a palavra reservada **Principal**, abrindo e fechando ela como descrito acima com as palavras **INICIO** e **FIM**. Seu retorno padrão é de acordo com o tipo referenciado na função, (inclusive na função **Principal** também), mas caso seja um retorno de função, será feita com a palavra reservada **Devolve** <argumento>.
- Na declaração de uma função, após seu nome deve vir o bloco de parâmetros delimitado por parênteses com a declaração do tipo e o nome da variável, separados por vírgula das outras variáveis.
- A função principal é declarada ao fim do programa, ou seja, precedendo todas as demais funções declaradas ao longo do código.

Ex.:

```
1  ▾ Funcao Inteiro subtracao(Inteiro value1, Inteiro value2) INICIO
2
3      Inteiro result;
4      result = value1 - value2;
5
6      Devolve result;
7
8  FIM
9
10 ▾ Funcao Inteiro Principal() INICIO
11
12     Inteiro value_one;
13     Inteiro value_two;
14
15     Entrada(value_one);
16     Entrada(value_two);
17
18     Imprimir(subtracao(value_one,value_two));
19
20     Devolve 0;
21
22 FIM
```

### 3. Conjuntos de Tipo de Dados e Nomes.

A linguagem de programação FiatTipo é case-sensitive e a atribuição de valores é uma instrução.

#### 3.1 Palavras Reservadas.

Nada, Imprimir, Verdade, Mentira, Entrada, Se, Porem, Booleano, Inteiro, Flutuante, ConjuntosDePalavras, Enquanto, Repita, Funcao, Principal, INICIO, FIM, Vazio, Ou, E, Caracter, Devolve.

#### 3.2 Identificador.

Os identificadores da linguagem FiatTipo seguem as seguintes regras:

- Um identificador começa **OBRIGATORIAMENTE** com letra minúscula.
- O resto da cadeia de caracteres do identificador podem ser maiúsculas ou minúsculas, números e/ou underline.
- O tamanho máximo de um identificador na linguagem é de 16 caracteres.
- É proibido o uso de espaços em brancos ou palavras reservadas.

#### 3.3 Comentário.

Na linguagem em existe apenas os comentários por linha, que é feito através do símbolo '#’.

#### 3.4 Inteiro.

**Inteiro** identifica as variáveis com números do tipo inteiro possuindo um número de bits limitados a 32 bits. Seus literais são expressos em uma sequência de números inteiros.

**Ex.: Inteiro** valor = 5;

### 3.5 Ponto Flutuante.

**Flutuante** identifica as variáveis com números do tipo ponto flutuante (números decimais, quebrados, etc) possuindo um número de bits limitados a 32 bits. Seus literais são expressos em uma sequência de números inteiros seguidos por um ponto e outra sequência de números que representam a parte decimal.

**Ex.: Flutuante** media = **34.65**.

### 3.6 Caracteres.

**Caracter** identifica variáveis do tipo caracter com tamanho de 8 bits. Onde seus literais são representados por apenas um caractere. **Ex.: Caracter** letra;

### 3.7 Cadeia de Caracteres.

**ConjuntosDePalavras** identifica variáveis do tipo caracter com tamanho de 8 bits, combinado com o tamanho da cadeia (arranjo). Seus literais são uma cadeia de caracteres de tamanho mínimo 0 e tamanho máximo ilimitado.

**Ex.: ConjuntoDePalavras** palavra[<tamanho da cadeia>];

### 3.8 Booleanos.

**Booleano** identifica variáveis do tipo Booleano, sendo possível atribuir apenas dois valores para variáveis deste tipo: Verdade ou Mentira.

**Ex.: Booleano** condicao = **Mentira**;

### 3.9 Arranjo Unidimensional.

Um arranjo (vetor) em FiatTipo é definido como em outras várias linguagens já existentes, exceção ao C. A definição é feita com a seguinte sentença <Tipo> Identificador[<Tamanho do Arranjo>].

**Exemplos:** **Inteiro** numeros[10]; **Flutuante** medias[10];

### 3.10 Operações Suportadas.

As operações suportadas pela linguagem FiatTipo estão descritas abaixo.

Tipo	Operações
Inteiro	Atribuição, aritméticos e relacionais
Flutuante	Atribuição, aritméticos e relacionais
ConjuntoDePalavras	Atribuição, relacionais, concatenação

Caracter	Atribuição, Relacionais
Booleano	Atribuição, lógica

O tipo **Flutuante** não suporta a operação de resto entre dois operandos, apenas o tipo Inteiro.

### 3.11 Valores Default.

Os valores default atribuídos a cada variável declarada são:

Tipo	Valores Default
Inteiro	0
Flutuante	0.0
ConjuntoDePalavras	Nada
Caracter	Nada
Booleano	Mentira

### 3.12 Coerção

A linguagem **FiatTipo** é estaticamente tipada, não aceitando coerção entre variáveis com tipos diferentes. As verificações de compatibilidade por tipo serão efetuadas estaticamente. Com isso, a linguagem passa a poder ter uma confiabilidade maior.

## 4. Conjuntos de Operações

### 4.1 Aritméticos

Operadores	Operações
+	Soma de dois operandos
-	Subtração de dois operandos
*	Multiplicação de dois operandos
/	Divisão de dois operandos

%	Resto da divisão entre operandos
-	Unário de Inversão: nega variáveis do tipo <b>Inteiro</b> e <b>Flutuante</b>
&	Concatena dois <b>ConjuntoDePalavras</b>

## 4.2 Relacionais

Operadores	Operação
==	Igualdade entre dois operandos
!=	Desigualdade entre dois operandos
>=	Maior ou igual que
<=	Menor ou igual que
>	Maior que
<	Menor que

## 4.3 Lógicos

Operadores	Operação
!	Negação
E	Conjunção
Ou	Disjunção

## 4.4 Concatenação de Cadeia de Caracteres

A concatenação na linguagem FiatTipo é representado pelo caracter “&” (ponto) e suporta apenas o tipo **ConjuntoDePalavras** de dados possuindo associatividade da esquerda para a direita. [rever]

## 4.5 Precedência e Associatividade

Precedência	Operadores	Associatividade
-------------	------------	-----------------



-	Menos unário	<i>Direita → Esquerda</i>
* /	Multiplicativos	<i>Esquerda → Direita</i>
%	Resto	<i>Esquerda → Direita</i>
+ -	Aditivos	<i>Esquerda → Direita</i>
!	Negação	<i>Direita → Esquerda</i>
< > <= >=	Comparativos	<i>Esquerda → Direita</i>
== !=	Igualdade	<i>Esquerda → Direita</i>
E Ou	Conjunção	<i>Esquerda → Direita</i>

#### 4.5.1 Operadores multiplicativos e aditivos

Quando se realiza uma operação em variáveis de tipos iguais utilizando os operadores, a saída produzida por estas operações têm de ser atribuídas a variáveis do mesmo tipo.

Caso se deseje empregar os operadores em variáveis de tipos diferentes, como por exemplo um **Inteiro** e um **Flutuante**, o tipo que prevalece é será o Flutuante. Não existe tratamento para os demais casos.

#### 4.5.2 Operadores Comparativos e igualdade

As operações comparativas e de igualdade geram um valor de tipo **Booleano** (verdadeiro ou falso) e não são associativas. Nas operações com tipos diferentes só é permitida a operação de comparação ou igualdade entre **Inteiro** e **Flutuante**.

#### 4.5.3 Operadores de negação e conjunção.

A determinação resultantes destas operações gera-se uma resposta também de tipo **Booleano**.

### 5. Instruções.

Na linguagem FiatTipo as instruções de uma linha são encerradas exclusivamente pelo símbolo “;”. Já os blocos (funções, condicionais, repetição, etc) são iniciados e terminados pelas delimitadores **INICIO** (*abertura*) e **FIM** (*fechamento*).[rever]

## 5.1 Atribuição.

Atribuições em FiatTipo são feitas através do símbolo “ = ”, sendo o lado esquerdo se refere ao identificador do tipo da variável, enquanto o lado direito se refere ao valor ou a expressão atribuída a mesma. Vale ressaltar que os dois lados da igualdade tem que ser do mesmo tipo, porque a linguagem não permite coerção.

Visando o aumento da confiabilidade da linguagem, a FiatTipo trata a atribuição como uma instrução e não como uma operação.

Ex.:

```
1  Inteiro num;
2  num = 10;
```

## 5.2 Estrutura condicional de uma ou duas vias.

### 5.2.1 Se e Porem.

O bloco da estrutura condicional **Se** sempre terá uma condição de entrada alguma operação lógica ou tão somente uma variável do tipo **Booleano**, seguido do bloco de instruções a serem executadas, demarcado pelas palavras **INICIO** e **FIM**.

O programa executa as instruções dentro do **Se**, se e somente se, a sua condição lógica for verdade. Se o condicional for falso as instruções a serem executadas serão as que estiverem contidas no bloco **Porem**.

Ex.:

**#Via única (Um bloco Se)**

**Se(<expressão lógica>)INICIO**

**<instruções Se>**

**FIM**

**#Via dupla (Bloco Se associado a um Porem)**

**Se(<expressão lógica>)INICIO**

**<instruções Se>**

**FIM**

**Porem INICIO**

**<instruções Porem>**

**FIM**

## 5.3 Estruturas Interativas.

### 5.3.1 Controle Lógico - Enquanto.

A estrutura de interação com controle lógico da linguagem FiatTipo implementa a estrutura **Enquanto**. A repetição na estrutura ocorre enquanto a condição de controle for **verdadeira**. O laço de repetição é executado enquanto a condição de controle lógica da estrutura for verdade. O laço encerra a partir do momento que a condição se torna falsa. Condições neste bloco são em sua maioria do tipo lógicas ou com variáveis booleanas.

**Ex.:**

```
Enquanto(<condição lógica>)INICIO
    <instruções do bloco Enquanto >
FIM
```

### 5.3.2 Controle por Contador - Repita.

Já na estrutura **Repita**, o número de interações é definido pelo programador e seu controle é feito por um contador, fazendo contraponto ao **Enquanto** descrito no 5.3.2. Contudo, por também se tratar de um bloco, utiliza-se as palavras reservadas **INICIO** e **FIM**.

Em uma estrutura interativa controlada por um contador, tem que possuir um valor inicial, um passo e um valor final (**start,step,stop**). O valor inicial é incrementado internamente pelo passo ao final de cada ciclo. O valor final deve ser sempre maior ou igual ao inicial para que o **Repita** seja executado. O número de interações é definido por:

**Nº de interações = (valor final - valor inicial)/step**

**Ex.:**

```
Funcao Inteiro count (Inteiro cont) INICIO
    Repita (Inteiro num=0,1, 10) INICIO
        cont = num + 1;
    FIM
    Devolve cont;
FIM
```

```
Funcao Inteiro Principal () INICIO
    Imprima (count);
FIM
```

Percebam que nesse caso o número de acréscimo é dado da seguinte forma: **Nº de iterações = 10 - 0 + 1**. Logo o valor que é retornado e impresso na tela será **11**.

## 5.4 Entrada e Saída.

A linguagem FiatTipo implementa as funções de entrada e saída através das palavras reservadas **Entrada** para o *input* de dados e **Imprimir[rever]** para a saída de dados (basicamente mostrar na tela).

### 5.4.1 Entrada.

Na função de *input* temos a atribuição de uma entrada fornecida pelo usuário a uma determinada variável de tipo correspondente, através do método **Entrada**.

### 5.4.2 Saída.

Irá mostrar na tela o(s) valor(es) correspondente(s) ao algoritmo específico no programa através do método **Imprimir**. Além disso, pode mostrar na tela um **ConjuntoDePalavras** sem ter sido necessariamente declarado antes, sendo preciso apenas colocar entre aspas simples o que for escrito na função **Imprimir** e será impresso na tela. Ou seja, podemos mostrar na tela textos escritos que não foram armazenados em variáveis.

Todas as declarações dentro de um só **Imprimir** serão mostradas na mesma linha e em **Imprimir** separados terá a quebra de linha.

Ex.:

**Mesma Linha:**

```
Imprimir(string1,string2);
```

**Linha Distintas:**

```
Imprimir(string1&"\n"&string2);
```

**Concatenação de variáveis com ConjuntoDePalavras:**

```
Imprimir(string1&"Olá mundo");
```

## 5.5 Funções.

De forma geral, as funções são muito semelhantes à linguagem de programação C. Onde de início ocorre a declaração da palavra reservada **Funcao**, para dizer ao compilador que ali começa uma função, em seguida o tipo de retorno da função, podendo ser **Inteiro**, **Vazio**, **Flutuante**, **Caracter**, **ConjuntoDePalavras** ou **Booleano**. Em seguida é necessário definir um identificador para a função, onde obrigatoriamente tem que se iniciar com letra minúscula. Em seguida, abertura de parênteses, parâmetros (ou não) e fechamento de parênteses. Por fim, **INICIO** para iniciar a escrita do bloco de instruções a serem executadas e ao fim **FIM** para informar o encerramento do bloco.

A linguagem **FiatTipo** não aceita sobrecarga de funções, ou seja, não é definida outra função com o mesmo identificador. Antes do **FIM** é obrigatório ter o **Devolve[rever]**, palavra reservada essa que efetua o retorno da função, caso não seja atribuído valor a ele, o valor default devolvido por ele é **zero**.

Para chamar uma função, deve ser utilizado o seu identificador, e dentro dos parênteses, os valores que serão utilizados pela função.

**Ex.:**

```
Funcao <Tipo (Inteiro,Flutuante...etc> <nome> (<Tipo> <identificador>) INICIO
    <declarações de Tipos e variáveis>
    <instruções>

    Devolve <valor a ser retornado>;
FIM
```

## 6.Exemplos de Algoritmos.

### 6.1 Hello World.

```
Funcao Inteiro Principal() INICIO
    Imprimir("Olá Mundo");
    Devolve 0;
FIM
```

### 6.2 Série de Fibonacci.

```
Funcao Inteiro fibonacci (Inteiro n) INICIO
    Se(n < 2) INICIO
        Devolve n ;
    FIM
    Porem INICIO
        Devolve fibonacci(n - 1) + fibonacci(n - 2);
    FIM
FIM
```

```
Funcao Inteiro Main ( ) INICIO
    Inteiro n;
    Imprimir("Digite o tamanho da sequencia: ");
    Entrada(n);
    Inteiro i = 0;
    Enquanto(i < n) INICIO
        Imprimir(fibonacci(i) & " , ");
        i = i + 1;
    FIM

    Devolve;
```

**FIM**

### 6.3 Shell Sort.

**Funcao Vazio shellsort(Inteiro array[ ], Inteiro n) INICIO**

**Inteiro** h = 1, c, j;

**Enquanto** (h < n) **INICIO**

        h = h \* 3 + 1;

**FIM**

    h = h / 3;

**Enquanto**(h > 0) **INICIO**

**Repita** (**Inteiro** i = h, 1, n) **INICIO**

            c = array[i];

            j = i;

**Enquanto** (j >= h && array[j - h] > c) **INICIO**

                array[j] = array[j - h];

                j = j - h;

**FIM**

            array[j] = c;

**FIM**

        h = h / 2;

**FIM**

**Devolve**;

**FIM**

**Funcao Inteiro Principal ( ) INICIO**

**Inteiro** n;

**Imprimir**("Digite o tamanho do array a ser ordenado: ");

**Entrada**(n);

**Inteiro** array[n];

**Imprimir**("Digite os número para serem ordenados: ");

**Repita** (**Inteiro** i = 0, 1, n) **INICIO**

**Entrada**(array[i]);

**FIM**

**Imprimir**("Valores adicionados: ");

**Repita** (**Inteiro** i = 0, 1, n) **INICIO**

**Imprimir**(array[i]);

**FIM**

**shellsort**(array[n], n);

**Imprimir**("Valores ordenados: ");

**Repita** (**Inteiro** i = 0, 1, n) **INICIO**

```
    Imprimir(array[i] & " ");
```

```
FIM
```

```
Devolve 0;
```

```
FIM
```