**UNIVERSIDADE FEDERAL DE ALAGOAS**

**CARLLOS EDUARDO FERREIRA LOPES**
**RAFAEL EMÍLIO LIMA ALVES**

# COMPILADORES - Linguagem PYragua

**Maceió**
**Dezembro de 2021**

# Sumário

# 1 Estrutura Geral de Um Programa

## 1.1 Escopo

O escopo utilizado pela linguagem Pyragua é estático, logo existe um escopo local definido por bloco de instruções para cada subprograma, onde cada instrução é delimitada por ";"e cada bloco de instruções é delimitada pelas palavras reservadas "Initiate"e "Halt".

Ademais, o aninhamento só é permitido em instruções com abertura de bloco (jamais para subprogramas). Para instruções que abrem um bloco temos que esse bloco compartilha seu escopo com o subprograma ao qual pertence essa instrução.

## 1.2 Ponto Inicial do Programa

O ponto inicial de um programa na linguagem PYragua é a função obrigatória **Central** ela é declarada a partir da palavra reservada Central e seu escopo é delimitado pelas palavras **Initiate** e **Halt**, as quais são também utilizadas para delimitar o escopo das demais funções e instruções com blocos. Como a função não necessita de um retorno específico, seu tipo é "Vazio".

**Exemplo**:

```
Funcao Vazio Central () Initiate

...;

Halt
```

## 1.3 Definição de Funções e Procedimentos

As funções e procedimentos nessa linguagem devem ser declarados fora da função "Central", sendo então declaradas antes do ponto inicial do programa. Sendo necessário, para a chamada de um subprograma, que ele esteja declarado antes do ponto onde o mesmo é chamado.

### 1.4 Definição e Escopo de Variáveis

O escopo de uma variável declarada dentro de um bloco de instruções pertecente a uma função é local, enquanto que se ela for declarada fora, ela assume um escopo global.

### 1.5 Nomes

A estrura dos nomes de variáveis e funções nessa linguagem devem seguir o seguinte padrão: iniciam-se por letras [a-zA-z] seguidas das letras [a-zA-z] ou os dígitos [0-9] ou [_], sendo assim temos a seguinte expressão regular: **'([a-zA-z])([a-zA-z] | [0-9] | '_')'*** . Além disso, a linguagem PYragua é case sensitive. Por fim, na linguagem Pyragua não há um limite de tamanho dos nomes.

## 2 Conjunto de Tipos de dados e Nomes

### 2.1 Palavras reservadas

**Initiate, Halt, Central, Funcao, Retorna, Int, Str, Float, Char, Bool, Array, Vazio, Verdade, Falso, Se, SeNao, Loop, Enquanto, Nulo, Pare, Ler, Escrever e Escreverpl.**

### 2.2 Declaração de Variáveis

Ao inicializarmos uma variável declaramos seu tipo seguido do identificador da mesma, podendo ou não ser seguida na mesma instrução por uma operação de atribuição. Caso não seja atribuído um valor em sua declaração, essa variável ficará com o valor default correspondente ao seu tipo. Entretanto, posteriormente a sua declaração pode ser feita uma instrução de atribuição.
Ademais, cada instrução não suporta múltiplas declarações ou atribuiçãos de variáveis.

### 2.3 Comentário

Nessa linguagem não podem existe comentário por blocos, apenas por linha, sendo cada linha de comentário iniciada com o caracter: "'#'".

### 2.4 Inteiro

A palavra reservada Int define as váriaveis/funções do tipo Inteiro, que armazenam um número inteiro de 32 bits cujos valores atribuídos podem ser de um ou mais digitos

obedecendo a expressão regular **'([0-9])*'**.

Exemplo: **Int** x = 10;

## 2.5 Ponto Flutuante

A palavra reservada Float define as variáveis/funções do tipo Ponto Flutuante, as quais possuem um número de bits limitados a 64 bits. Onde os valores atribuídos são constituídos de um ou mais digitos, representando a parte inteira, seguidos de um ponto final e um ou mais digitos representando os valores decimais, sendo assim: **'([0-9])+(\.)([0-9])+'** .

Exemplo: **Float** media = 9.25;

## 2.6 Caractere

A palavra reservada Char define as variáveis do tipo caractere, as quais possuem tamanho de bits 8 referentes aos códigos da tabela ASCII. Sendo assim, o valor atribuído à esse tipo pode ser qualquer letra, símbolo ou dígito desde que esteja entre apostrofe. Portanto, obedecendo à seguinte expressão regular: **'(\')(.)(\')'**

Exemplo: **Char** a = 'a';

## 2.7 Booleano

A palavra reservada Bool define as variáveis ou funções do tipo booleano que pode assumir apenas um entre os dois valores: Verdadeiro ou Falso, ambos palavras reservadas. Portanto, obedecendo à seguinte expressão regular: **'("Verdadeiro"|"Falso")'**

Exemplo: **Bool** isRunning = Falso;

## 2.8 Cadeia de Caracteres

A palavra reservada Str define as variáveis de Cadeia de Caractere, as quais não possuem um tamanho definido. Os valores que podem ser atribuídos à essas variáveis são uma sequência de 0 ou mais caracteres.Portanto, obedecendo à seguinte expressão regular: '(")(.)*(")'

Exemplo: **Str** Nome = "Alberto";

## 2.9 Arranjo Unidimensional

Um Arranjo Unidimensional (vetor) na linguagem PYragua é um conjunto de elementos do mesmo tipo primitivo e de tamanho fixo. A declaração de um arranjo obedece a mesma estrutura vista previamente, entretanto após o tipo e identificador, temos um colchete aberto seguido de um valor inteiro que representa o tamanho do arranho e um colchete fechado, sendo assim temos a seguinte estrutura: **Tipo Identificador [Tamanho]**.

O Tamanho, o valor inteiro mencionado anteriormente, pode ser tanto uma constante literal, quanto uma variável inteira declarada e atribuída previamente, entretanto é válido citar que a partir do momento que o arranjo é declarado, seu tamanho é fixo, dessa forma se usarmos uma variável, mesmo que alteremos seu valor em uma instrução posterior, o tamanho do arranjo permanece imutável. O tamanho define quantos dados do tipo determinado serão armazenados no Arranjo.

A atribuição de valores a um arranjo pode ser feita em sua declaração ou posteriormente, todos os valores a ser atribuídos devem estar entre colchetes e separados entre si por vírgula, caso a atribuição seja feita posteriormente, basta apenas informar o identificador seguido do operador de atribuição e o conjunto de valores na estrutura especificada previamente. Os valores serão atribuídos na ordem que estão dentro dos colchetes. Caso sejam informados um número de valores acima do valor do tamanho do arranjo, ocorrerá um erro, ademais, caso seja informado uma quantidade valores inferior, as posições sem valores irão receber o valor padrão daquele tipo primitivo.

Por fim, para acessar apenas um elemento basta que escrevamos o identificador seguido de um colchete aberto, o índice do elemento (valor inteiro) e colchete fechado: **Identificador [Índice]**.A primeira posição em um arranjo nessa linguagem corresponde ao índice 0, as posições seguintes seguem de modo crescente, logo a segunda posição será o índice 1, a terceira será o índice 2 e assim sucessivamente.

Exemplo:

```
Float Notas [4];                    # Declaração

Notas = [7.3, 6.8, 4.5, 8.6];       # Atribuição

Int Idades [4] = [18, 26, 17, 19]; # Declaração e Atribuição

Idades[0] = Idades[0] + 1;          # Acessando o primeiro valor
```

## 2.10 Equivalência de tipos

A linguagem Pyragua permite que valores de qualquer tipo possam ser convertidos para uma cadeia de caracteres através de uma operação de concatenação. Sendo assim, valores do tipo Char irão virar uma cadeia de tamanho 1, enquanto que valores booleanos ou numéricos irão se tornar uma cadeia correspondente aos seus valores literais.

Ademais, também é permitida a conversão implícita de valores do tipo ponto flutuante para inteiros, através de operações onde só serão consideradas as parcelas inteiras do ponto flutuante.

Por fim, não há coerção explícita nessa linguagem.

# 3 Conjunto de operadores

## 3.1 Operadores Aritméticos

| Operação | Símbolo | Exemplo |
|---|---|---|
| Adição | + | a + b |
| Subtração | - | a - b |
| Multiplicação | * | a * b |
| Divisão | / | a / b |
| Resto de Divisão | % | a % b |
| Exponenciação | ^ | a ^ b |
| Unário Negativo | - | - a |

## 3.2 Operadores Relacionais

| Operação | Símbolo | Exemplo |
|---|---|---|
| Igualdade | == | a == b |
| Diferente | != | a != b |
| Menor que | < | a <b |
| Maior que | > | a >b |
| Menor igual | <= | a <= b |
| Maior igual | >= | a >= 2 |

## 3.3 Operadores Lógicos

| Operação | Símbolo | Exemplo |
|---|---|---|
| Negação | ! | !a |
| Conjunção | & | a & b |
| Disjunção | \| | a \| b |

## 3.4 Operadores de Concatenação

| Operação | Símbolo | Exemplo |
|---|---|---|
| Concatenação | @ | a@b |

## 3.5 Operações de cada tipo de dado

| Operador | Tipos que realizam a operação |
|---|---|
| +, - , *, %, ^ ,<, >, <=, >= | Int, Float |
| ==, !=, @ | Int, Float, Bool, Char, Str |
| &, \|, ! | Bool |

## 3.6 Valores Padrão

Os valores padrões da linguagem são atribuídos às variáveis na declaração das mesmas, antes que o usuário faça uma instrução atribuindo novos valores de acordo com a necessidade de sua aplicação. Sendo assim, quando as variáveis são atribuídas, os seguintes valores padrões são associados de acordo com o tipo:

| Tipo | Valor Default |
|---|---|
| Int | 0 |
| Float | 0.0 |
| Char | Nulo |
| Str | Nulo |
| Bool | Falso |

## 3.7 Precedência e Associatividade

| Operadores | Operação | Associatividade |
|---|---|---|
| = | Atribuição | Não associativo |
| - | Unário Negativo | Direita ->Esquerda |
| * / | Multiplicação, divisão | Esquerda ->Direita |
| %, ^ | Resto e Exponenciação | Esquerda ->Direita |
| - + | Subtração e Adição | Esquerda ->Direita |
| ! | Negação | Direita ->Esquerda |
| <><= >= | Comparação | Esquerda ->Direita |
| == != | Igualdade e Desigualdade | Esquerda ->Direita |
| & \| | Lógicos | Esquerda ->Direita |

# 4 Instruções

## 4.1 Atribuição

A atribuição na linguagem é definida pelo operador '=', onde à esquerda temos o identificador de uma variável já definida que receberá a atribuição e à direita do símbolo temos um valor que será atribuído correspondente ao tipo da variável, o qual pode ser uma outra variável, uma constante literal, uma expressão ou uma chamada de função.

**Exemplos**:

```
Int A = 10;

Int B;

B = A;
```

## 4.2 Estruturas Condicionais de Uma ou Duas Vias

As estruturas condicionais na linguagem podem ser de via única ou dupla. A primeira estrutura condicional deve ser usando a palavra reservada **Se**, seguida de uma expressão lógica entre parênteses ou de uma variável booleana e dos delimitadores do bloco de código onde instruções estarão definidas. Caso o valor da expressão ou da variável entre parênteses seja Verdadeiro, as instruções dentro do bloco de código serão executadas. Em contrapartida, caso seja de valor Falso, então será ignorado.
Após o Se, caso a expressão seja de via dupla teremos um **SeNao**, onde caso a expressão

lógica ou valor de variável do Se não assuma um valor verdadeiro, então será executado o bloco de código contido no SeNao, o qual também é delimitado pelas palavras reservadas Initiate e Halt.

**Exemplo uma via:**

```
Se (EXPRESSÃO LÓGICA) Initiate

...;

Halt
```

**Exemplo duas vias:**

```
Se (EXPRESSÃO LÓGICA) Initiate

...;

Halt


SeNao Initiate

...;

Halt
```

## 4.3 Estrutura Iterativa de Controle Lógico

A palavra reservada atribuída para essa instrução é o **Enquanto** o qual deve ser seguido de uma expressão lógica, variável ou constante booleana entre parênteses e os delimitadores de escopo (Initiate e Halt), os quais irão conter em seu interior instruções que serão executadas em um laço de repetição enquanto o valor entre parenteses resultar em Verdadeiro.

A verificação ocorre antes de entrar no bloco, portanto se a expressão não assumir um valor Verdadeiro desde a primeira exeecução então as instruções no bloco não serão executadas. Ademais, tanto essa quanto a estrutura iterativa por contador podem ser

interrompidas de imediato pelo comando "Pare;".

**Exemplo:**

```
Enquanto (EXPRESSÃO LÓGICA) Initiate

    ...;

Halt
```

## 4.4 Estrutura Iterativa por Contador

Na linguagem PYragua, essa instrução é definida pela palavra reservada **Loop**, a qual é seguida de parênteses com valores para a execução da instrução, cada um deles separados por vírgula.
O primeiro valor é uma variável inteira que define o contador que será incrementado a cada iteração do laço, pode ser utilizada uma variável declarada anteriormente ou se declarar uma nova variável. Após a variável contador, temos o valor inicial do contador, seguido do valor final, ou seja, o limite que deve ser atingindo e por último temos o valor que será incrementado à variável contador após cada iteração. v

Após esses valores, temos os delimitadores de escopo (Initiate e Halt), onde estão contidas as instruções que serão realizadas em cada iteração do ciclo.

As instruções dentro da estrutura só deixarão de ser executadas quando a variável utilizada como contador ultrapassar ou atingir o valor do limitante.

**Exemplo 1:** Contador declarado no comando, iniciado no valor 0, com limitante 10 e variável de incremento explícita no valor 2.

```
Loop (Int contador, 0, 10, 2) Initiate

    ...;

Halt
```

**Exemplo 2:** Contador declarado antes, iniciado no valor 0, com limitante 10 e variável de incremento não fornecida, portanto assumindo o valor 1.

```
Int c;

Loop (c, 0, 10) Initiate

    ...;

Halt
```

## 4.5 Desvios Incondicionais

A linguagem Pyragua não apresenta suporte para desvios incondicionais.

## 4.6 Entrada e Saída

### 4.6.1 Entrada

Para entrada de dados é utilizada a palavra reservada **Ler**, segunda de parênteses que contém o identificador de uma ou múltiplas variáveis previamente declaradas, em caso de múltiplas variáveis, elas devem ser separadas por vírgula. Os valores informados pelo usuário serão atribuídos às variáveis na ordem em que foram informados. Dessa forma, permitindo que o usuário informe o valor de variáveis.

**Exemplo:**

```
Int a;

Int b;

Ler(a, b);
```

### 4.6.2 Saída

Para a saída de dados, a linguagem utiliza dois comandos, sendo esses o **Escrever** e o **Escreverpl**.

Os dois comandos mantém uma estrutura semelhante, onde após eles temos parênteses que irão comportar os dados que serão impressos, podendo ser apenas um, ou uma sequência, desde que estejam separados por vírgula. Os valores podem ser uma chamada de função (será printado seu retorno), uma variável, uma constante literal ou uma operação. A diferença entre os dois comando é que enquanto o Escrever irá exibir os valores

na mesma linha, o Escreverpl irá escrever um valor por linha. Ademais, em qualquer um dos dois comandos uma quebra de linha pode ser definida pelo uso de
**n**

**Exemplo 1:**

```
Int c = 20;

Escrever (3, "Olá", 'a', c);
```

Saída:

```
3 Olá a 20
```

**Exemplo 2:**

```
Str nome = "Alfredo";

Escreverpl ("Bom dia", nome, 20.8);
```

# Saída:

```
Bom dia

Alfredo

20.8
```

## 4.7 Funções

Como falado anteriormente, as funções podem ser definidas em qualquer parte do documento, exceto dentro do escopo de outras funções e abaixo da função Central. A declaração de uma função é iniciada pela palavra reservada **Funcao** seguida do tipo de dado do retorno da função (Int, Bool, Str, Float, Char) ou Vazio caso não haja retorno, seguido do identificador da função e parênteses que podem ou não conter parâmetros. Por fim, seu escopo é delimitado pelas palavras reservadas Initiate e Halt entre as quais estarão as instruções.

Caso hajam parâmetros, é necessário declarar o seu tipo e identificador, podendo haver nenhum ou múltiplos parâmetros. Por fim, na chamada da função basta apenas que se passe a variável já declarada, constante literal, operação ou até mesmo função (desde que ela retorne um valor correspondente ao tipo de parâmetro), bastando apenas que o tipo do parâmetro passado seja correspondente ao tipo que está declarado na função. Tanto na chamada, quanto na declaração, em caso de múltiplos parâmetros, todos devem estar separados por vírgula.

**Exemplo 1:** Declaração da função

```
Funcao Bool ChecaMaiorIdade (Int idade) Initiate

        Se (Idade >= 18) Initiate

            Retorna Verdadeiro;

        Halt

        SeNao Initiate

            Retorna Falso;

        Halt

Halt
```

**Exemplo 2:** Chamada da função

```
Bool aux;

aux = ChecaMaiorIdade(15);
```

# 5 Algoritmos

## 5.1 Alô Mundo

```
Funcao Vazio Central() Initiate

    Escrever('Alô mundo!')

Halt
```

## 5.2 Fibonacci

```
Funcao Vazio fibonacci (Int n) Initiate

    Int n1 = 0;
    Int n2 = 1;
    Int n3;

    Escrever(n)

    Se ( n == 1 ) Initiate

        Escrever ("0, ", n)

    Halt

    SeNao  Initiate

        Char separador = ',';
        Escrever("0, 1, ");

        Enquanto (Verdadeiro) Initiate

            n3 = n1+n2;
            Se (n3 > n) Initiate
                Retorna;
            Halt

            Escrever(separador, " ", n3)
            n1 = n2;
```

```
            n2 = n3;

        Halt

    Halt

Halt



Funcao Vazio Central() Initiate

    Int n;
    Ler(n);
    fibonacci(n);


Halt
```

## 5.3 Shell Sort

```
Funcao Vazio shellsort(Int array[ ], Int n) Initiate

Int m = 1;
Int c;
Int j;


    Enquanto (m < n) Initiate
        m = m * 3 + 1;
    Halt

    m = m / 3;

    Enquanto(m > 0) Initiate

        Loop (Int i = m, 1, n) Initiate

                c = array[i];
```

```
                j = i;

                Enquanto (j >= m E array[j - m] > c) Initiate

                        array[j] = array[j - m];
                        j = j - m;

                Halt

                array[j] = c;
        Halt

        m = m / 2;

    Halt

    Retorna;

Halt

Funcao Vazio Central() Initiate
 Int n;
 Int atual;

 Escrever("Tamanho do array: ");
 Ler(n);

 Int array[n];

 Escrever("Elementos do array: ");

        Loop (Int i = 0, 1, n) Initiate
         Ler(array[i]);
        Halt

        Escrever("Array: ");

        Loop (Int i = 0, 1, n) Initiate
            atual = array[i];
```

```
         Escreverpl(atual);
        Halt

 shellsort(array[n], n);

 Escrever("Valores ordenados: ");

        Loop (Int i = 0, 1, n) Initiate
            atual = array[i];
         Escreverpl(atual);
        Halt

 Retorna;
Halt
```