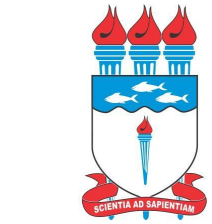


UNIVERSIDADE FEDERAL
DE ALAGOAS

UNIVERSIDADE FEDERAL DE ALAGOAS - UFAL

INSTITUTO DE COMPUTAÇÃO - IC

CIÊNCIA DA COMPUTAÇÃO

JONAS SANTOS DE ALMEIDA ALVES

RICK MARTIN LINO DOS SANTOS

JOSÉ ERALDO DOS SANTOS NETO

Especificações da Linguagem Mopa

TRABALHO DE COMPILADORES

Maceió

2021

JONAS SANTOS DE ALMEIDA ALVES

RICK MARTIN LINO DOS SANTOS
JOSÉ ERALDO DOS SANTOS NETO

Especificações da Linguagem Mopa

Maceió
2021

Sumário

1 Introdução

A linguagem Mopa foi criada com as seguintes características: é estaticamente tipada, ou seja, não admite coerção, não orientada a objetos, não possui tratamento para erros de detecção de tipo, não permite a criação de funções dentro de outras funções, e é baseada em Pascal e C. Seu objetivo é poder ser uma linguagem usada para aprender, pois, é simples de compreender seus comandos e leitura de código para pessoas que nunca programaram algo. O nome Mopa é uma homenagem a um grupo de whatsapp que os participantes desse projeto participam.

2 Estrutura Geral do Programa

Um programa da Linguagem Mopa é escrito usando estes parâmetros:

1) Para abrir e fechar um bloco de função ou um bloco de algum mecanismo no programa, usa-se **Inicio** (abrir) e **Fim** (fechar).

2) A função principal do programa é declarada com a palavra reservada **Principal**, abrindo e fechando seus limites com as palavras reservadas acima. A função principal e as demais funções retornam com a palavra reservada **Devolve.**

3) Quando uma função é declarada, depois do nome, deve-se ter o bloco de parâmetros que são delimitados por parênteses, com seu tipo de retorno também declarado e os nomes de variáveis que serão passadas dentro dos parênteses.

4) A função principal será declarada ao fim do programa, ou melhor, precedendo-se todas as outras funções que serão declaradas no código do programa.

Exemplo:

Figura 1 – Exemplo - Geral

Exemplo1-Especificações.txt

```
1   Funcao Flutuante soma(Flutuante valor1, Flutuante valor2) Inicio
2
3       Flutuante resultado;
4       resultado = valor1 + valor2;
5
6       Devolve resultado;
7
8   Fim
9
10  Funcao Inteiro Principal() Inicio
11
12      Flutuante valor1;
13      Flutuante valor2;
14
15      Entrada(valor1);
16      Entrada(valor2);
17
18      Imprimir(soma(valor1,valor2));
19
20      Devolve;
21
22
23  Fim
24
```

3 Conjunto de Tipos de Dados e Nomes

A linguagem Mopa é sensível a letras maiúsculas e minúsculas (case-sensitive), e a atribuição de valores é uma instrução.

3.1 Palavras Reservadas

Inicio, Fim, Funcao, Imprimir, Inteiro, Verdade, Mentira, Nada, Ou, Principal, Porem, Repita, Se, Vazio, Booleano, Caracter, ConjuntoDePalavras, Devolve, E, Entrada, Enquanto, Flutuante.

3.2 Comentários

Na Linguagem Mopa só é admitido comentários por linha, onde o que o delimita é o caractere '**#**'.

3.3 Identificadores

A Linguagem Mopa é regida por algumas regras:

- A cadeia de caracteres do identificador podem ter underline, letras maiúsculas e minúsculas, mas iniciar com uma letra, maiúscula ou minúscula;
- Foi definido o limite máximo de um identificador é de até 16 caracteres;
- Precisa-se obrigatoriamente que uma variável seja declarada antes que seja usada no bloco de instruções do programa.
- Não será permitido o uso de espaços em branco nas palavras reservadas.

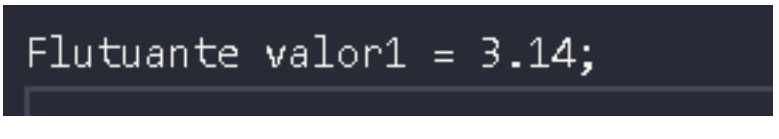
.

3.4 Tipos de Variáveis

3.4.1 Ponto Flutuante

A palavra reservada **Flutuante** identifica as variáveis do tipo ponto flutuante (números decimais), com no máximo 32 bits. É expresso por números inteiros, um ponto para dividi-los, e mais números inteiros para representá-los.

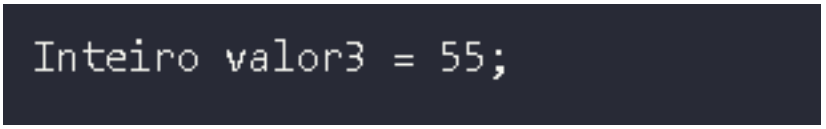
Figura 2 – Exemplo - Ponto Flutuante



3.4.2 Inteiro

A palavra reservada **Inteiro** identifica as variáveis do tipo inteiro com no máximo de 32 bits. É expresso por números inteiros em sequência sem espaços.

Figura 3 – Exemplo - Inteiro



3.4.3 Caracteres

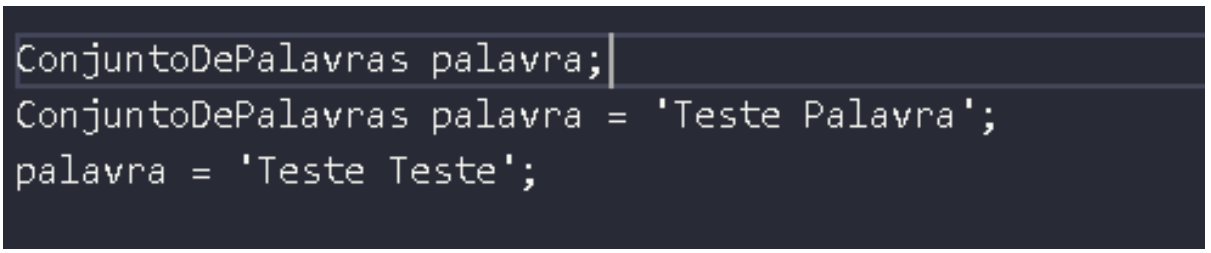
A palavra reservada **Caracter** com tamanho de 8 bits identifica variáveis do tipo caractere. É expresso por apenas um caractere entre aspas simples.

Figura 4 – Exemplo - Caracteres

3.4.4 Cadeia de Caracteres

A palavra reservada **ConjuntoDePalavras** identifica variáveis do tipo caractere com tamanho de 8 bits, que são combinados com o tamanho da cadeia que é dinâmica, com **n** caracteres no padrão ASCII. É expresso em uma cadeia de caracteres com tamanho mínimo de 0. É possível declarar apenas a variável sem atribuir valor, e também pode ser feita uma atribuição direta na linha de declaração. Também é delimitado por aspas simples e é possível usar underscore e espaços em branco.

Figura 5 – Exemplo - ConjuntoDePalavras



3.4.5 Booleano

A palavra reservada **Booleano** identifica variáveis do tipo booleano, onde é possível atribuir apenas dois valores para esse tipo de variável.Verdade ou Mentira.

Figura 6 – Exemplo - Booleano

```
Booleano porta = verdadeiro;
```

3.4.6 Arranjo Unidimensional

Um arranjo (vetor) na linguagem Mopa é definido como em outras linguagens que já existem. A sua definição é dada desta maneira:

<Tipo> Identificador[<Tamanho do Vetor>];

Nas funções, a passagem por parâmetro é feita passando o tamanho do vetor para uma referência que será ligada na função de destino.

3.5 Operações Suportadas

As operações suportadas pelos tipos de variáveis da linguagem Mopa

Tipo	Operação
Flutuante	Atribuição, Aritimética, Relacional, Concatenação
Inteiro	Atribuição, Aritimética, Relacional, Concatenação
ConjuntoDePalavras	Atribuição, Relacional, Concatenação
Booleano	Atribuição, Relacional, Concatenação
Caracter	Atribuição, Lógica, Relacional de Igualdade e Desigualdade, Concatenação

Observação: O ponto flutuante não admite a operação de resto entre dois operandos do tipo dele, e a linguagem Mopa só aceita a operação de resto entre dois operandos do tipo inteiro.

3.6 Valores Default

Tabela 2 – Os valores para cada tipo de variáveis da linguagem Mopa

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

3.7 Coerção

A linguagem Mopa é estaticamente tipada, então, não aceita coerção entre variáveis de tipos diferentes. As verificações de compatibilidade por tipo serão efetuadas estaticamente. Com isso, temos maior confiabilidade.

4. Conjunto de Operações

4.1 Aritméticas

Operadores	Operações
+	Soma de dois operandos
-	Subtração de dois operandos
*	Multiplicação de dois operandos
/	Divisão de dois operandos
%	Resto da divisão entre dois operandos
- Inversão	Negação de váriaveis do tipo **Inteiro** e **Flutuante**
&	Concatenação de dois **ConjuntosDePalavras**

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 Mopa é representada pelo caractere “&“ , e se pode suportar apenas o tipo ConjuntoDePalavras de dados que possuem associatividade da esquerda para a direita.

4.5 Precedência e Associatividade

Precedência	Operadores	Cabeçalho
~	Menos unário	Direita -> Esquerda
* /	Multiplicação e Divisão	Esquerda -> Direita
%	Resto	Esquerda -> Direita
+ -	Soma e Subtração	Esquerda -> Direita
!	Negação	Direita -> Esquerda
<> <= >=	Comparativos	Sem associatividade
== !=	Igualdade	Esquerda -> Direita
E Ou	Conjunção	Esquerda -> Direita

4.5.1 Operadores Multiplicativos e de Soma e Subtração

Quando acontece uma operação em variáveis de tipos iguais utilizando os operadores, a saída produzida pela mesma operação terá de ser atribuída por uma variável do mesmo tipo.

Caso queira implantar os operadores em variáveis de tipos diferentes, como por exemplo a operação de um **Flutuante** com um **Inteiro**, o tipo que prevalecerá será o **Flutuante**. Não existe tratamento para outros casos deste tipo.

4.5.2 Operadores Comparativos e Igualdade

As operações comparativas e de igualdade geram um valor do tipo **Booleano** (Verdade ou Mentira) e não são associativas. Nas operações com tipos diferentes só é permitida a operação ou igualdade entre I**nteiro** e **Flutuante**.

4.5.3 Operadores de Negação e Conjunção

A determinação resultante destas operações geram uma resposta também do tipo Booleano.

Na negação, o que é verdade será mentira, e o que é mentira será verdade. Já na conjunção, depende dos valores dos operandos.

5. Instruções

Na linguagem Mopa as instruções são delimitadas, ou melhor, são dadas como encerradas somente pelo símbolo " ; ". Seus blocos, como já foi dito acima, são delimitados pela palavra reservada Inicio e Fim.

5.1 Atribuição

As atribuições na linguagem Mopa são feitas pelo caractere ' = ' envolvido por dois operandos, o da esquerda precisa ser necessariamente uma variável, onde será atribuída a sua sentença, o da direita pode ser uma variável ou um valor pré-definido, onde os dois lados da igualdade precisam ser do mesmo tipo, pois a linguagem não admite coerção. Além disso, a linguagem Mopa não trata a atribuição como operação, e sim como instrução por trazer mais confiabilidade à linguagem.

Figura 7 – Exemplo - Atribuição

```
Inteiro valor3 = 55;
```

5.2 Estruturas Condicionais

5.2.1 Se e Porem

A estrutura condicional **Se** sempre terá uma condição, que será validada por uma expressão lógica ou terá uma variável do tipo **Booleano**, seguido do bloco de instruções a serem executados, demarcado pelas palavras que delimitam o bloco, **Inicio** e **Fim**.

O programa executa as instruções dentro do bloco associado ao **Se**, se e somente, a sua expressão lógica é verdadeira. Se o condicional for falso as instruções a serem executadas serão as que estiverem contidas no bloco **Porem**. Caso não exista um bloco **Porem**, o programa irá apenas verificar o bloco associado ao **Se**.

Figura 8 – Exemplo - Se e Porem

```
# Bloco Simples

Se(valor1 == 2) Inicio
    valor2 = 5;
Fim

# Bloco Composto

Se(valor1 == 2) Inicio
    valor2 = 5;
Fim
Porem Inicio
    valor2 = 10;
Fim
```

5.3 Estruturas Interativas

5.3.1 Enquanto - Controle Lógico

A estrutura de interação com controle lógico da linguagem Mopa implementa a estrutura de controle lógico definida com a palavra reservada **Enquanto**. A repetição na estrutura ocorre enquanto a condição for **verdadeira**. A repetição se encerrará quando a condição se tornar **falsa**.

Figura 9 – Exemplo - Enquanto

```
Enquanto(valor1 == 1) Inicio
    Imprimir(valor2);
    valor1 = valor1 + 1;
Fim
```

5.3.2 Repita - Controle por Contador

Já na estrutura **Repita,** o número de iterações é definido pelo programador e seu controle é feito por um contador dentro de sua estrutura. Mesmo assim, usa-se a delimitação com as palavras reservadas **Inicio** e **Fim,** como em qualquer outra estrutura.

Na estrutura iterativa controlada por um contador, tem que possuir um valor inicial, um passo para repetição e uma parada final (**Start, Step, Stop**) .

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

Figura 10 – Exemplo - Repita

```
Funcao Inteiro testeContador (Inteiro cont) Inicio

    Repita(Inteiro num = 0, 2, 20) Inicio
        cont = cont + num;
    Fim

    Retorno cont;
Fim

Funcao Inteiro Principal() Inicio
    Imprima(count(0));
Fim
```

5.4 Entrada e Saída

A linguagem mopa implementa as funções de entrada e saída através das palavras reservadas Entrada para o input de dados e Imprimir 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

Será mostrado na tela o valor correspondente ao algoritmo especificado 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.

Mesma Linha:

Imprimir(string1,string2);

Linha Distintas:

Imprimirnl(string1&string2);

Concatenação de variáveis com ConjuntoDePalavras:

Imprimir(string1&”Olá mundo”);

5.5 Funções

Fazendo uma análise 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.

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 o valor padrão em cada tipo.

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

6. Exemplos

6.1 Hello World

Figura 11 – Exemplo - Hello World

```
Funcao Inteiro Principal() Inicio

    Imprimir('Olá Mundo');

    Devolve;
Fim
```

6.2 ShellSort

Figura 12 – Exemplo - ShellSort

```
Funcao Vazio shellsort(Inteiro vetor[ ], 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 = vetor[i];
                    j = i;
                    Enquanto (j >= h E vetor[j - h] > c) Inicio
                            vetor[j] = vetor[j - h];
                            j = j - h;
                    Fim
                    vetor[j] = c;
            Fim
            h = h / 2;
        Fim

        Devolve;
    Fim
```

```
Funcao Inteiro Principal ( ) Inicio
        Inteiro n, v;
        Imprimir('Digite o tamanho do array a ser ordenado: ');
        Entrada(n);
        Inteiro vetor[n];

        Imprimir('Digite aleatoriamente os numero para serem ordenados: ');
        Repita (Inteiro i = 0, 1, n) Inicio
            Entrada(vetor[i]);
        Fim
        Imprimir('Valores adicionados: ');
        Repita (Inteiro i = 0, 1, n) Inicio
            v = vetor[i];
            Imprimirnl(v);
        Fim

        shellsort(vetor[n], n);

        Imprimir('Valores ordenados: ');

        Repita (Inteiro i = 0, 1, n) Inicio
            v = vetor[i];
            Imprimirnl(v);
        Fim

        Devolve;
Fim
```

6.3 Série de Fibonacci

Figura 13 – Exemplo - Fibonacci

C: > Users > Jonas > Documents > UFAL > fibonacci.txt

```
1   Funcao Inteiro fibonacci(Inteiro n) Comeco
2       Se(n < 2) Comeco
3           Retorno n;
4       Final
5       Porem Comeco
6           Retorno fibonacci(n - 1) + fibonacci(n - 2);
7       Final
8   Final
9
10  Funcao Inteiro Principal() Comeco
11      Inteiro n;
12      Imprimir('Digite o tamanho da sequencia:');
13      Entrada(n);
14
15      Repita(Inteiro i = 0, 1, n) Comeco
16          total = fibonacci(i);
17          Imprimir(total);
18      Final
19
20      Retorno;
21  Final
```