Universidade Federal de Santa Catarina - Campus Araranguá
Disciplina: Construção de Compiladores
Grupo: Ale Chaito, Thiago Becker, Mihael Zamin

Manual da Linguagem Coffee

Introdução

A linguagem Coffee é composta por três partes sendo elas: declaração de variáveis, declaração de procedimentos e corpo do programa. Sendo as duas primeiras opcionais e a terceira obrigatória. Dentro do corpo do programa pode haver diversos ou nenhum comando, que desempenham uma tarefa específica. Todos os comandos são finalizados com o caracter ponto e vírgula. Os demais detalhes de como implementar as estruturas de comentário, repetição e decisão serão vistas mais adiante no manual.

Case Sensitive

O compilador da linguagem coffee reconhece somente caracteres minúsculos, caso seja inserido em alguma string um ou mais caracteres maiúsculos, será gerado um erro de interpretação léxica;

Tipos e Variáveis

As variáveis da linguagem apresentam tipagem estática, ela podem ser setadas como um tipo integer, char, float, string. Antes de atribuir um valor elas precisam estar declaradas.

Sintaxe de Declaração de Variáveis

Restrição: 20 caracteres para um nome de variável, ao ultrapassar esse valor gera um erro léxico.

var_name : integer; @declarada uma variável do tipo integer
var_name1, varname2, varname3 : string; @declarada tres variaveis do tipo string

Conhecendo os Tipos

- **Integer**
    - O tipo integer representa um número do conjunto dos números inteiros Z, que compreende os números positivos e negativos. Os valores que podem ser

representados por um integer devem ser menores que 2^{10} (1024 – valor máximo). A declaração é feita através da palavra reservada ***integer***.

Exemplo de programa

a: integer; @declaracao de variaveis, antes da atribuição
a = 5; @a variável criada recebe o valor 5

- **Char**
  - O tipo char é utilizado para representar caracteres. Um caractere é representado através de um byte na memória. Lembre-se que um byte tem 8 bits, ou seja, é possível representar 256 números (ou no caso, codificar até 256 caracteres distintos). A linguagem utiliza esse número como um índice na tabela ASCII. A declaração é feita através da palavra reservada ***char***.

Exemplo de programa

b: char; @declaracao de variaveis, antes da atribuição
b = 32; @a variável criada recebe o valor 32 presente na tabela ASCII

- **Float**
  - O tipo float representa números fracionários e números reais (o que inclui os números inteiros). A faixa de valores varia de 1,2e-10 até 3,4e+10. A declaração é feita através da palavra reservada ***float***.
  - Para caracterização de um número float, as casas decimais devem ser separadas por ponto (.), exemplo: 2,35 / 0,5

Exemplo de programa

c: float; @declaracao de variaveis, antes da atribuição
c = 0.0345; @a variável criada recebe o valor 0.0345 com ponto flutuante

- **String**
  - O tipo string representa um conjunto de caracteres encodados com ASCII sequenciados em um vetor. O tamanho máximo da palavra e de 100 caracteres. A declaração é feita através da palavra reservada ***string***.

Exemplo de programa

d: string; @declaracao de variaveis, antes da atribuição
d = “lorem ipsum”; @a variável criada recebe um conjunto de caracteres

Operadores

Os operadores existem na linguagem para que se possa fazer possível criar operações com as variáveis ou um valor. Existem 3 tipos de operadores os aritméticos, comparativos e os de atribuição.

Aritméticos

OPERAÇÃO	COMANDO
soma	+
subtração	-
divisão	/
multiplicação	*

Exemplo de programa

```
x, y, z: integer; @criacao de 2 variáveis tipo integer
x = 5;
y = 2;
z = x + y; @a variável z criada recebe o resultado da soma x,y
```

Comparadores

OPERAÇÃO	COMANDO
igual	==
diferente	!=
menor	<
menor ou igual	<=
maior	>
maior ou igual	>=

Exemplo de programa

```
if(x <= 5) { @variavel menor ou igual ao valor 5
        ……
}
```

Atribuição

Sintaxe

var_name = expressão;

Exemplo

x = 5; @valor 5 sendo atribuído a variável x, exemplo 1

z = x + b; @valor x + b sendo atribuído a variável z, exemplo 2

Literais

Os literais são sequências de caracteres delimitadas por aspas duplas (“), são úteis para propagar mensagens de output ou input. Permitido somente caracteres minúsculos.

Restrição: 256 caracteres, ao ultrapassar esse valor gera um erro léxico.

Exemplo

x : string;
x = “hello world”

Comandos

Comando é uma instrução da linguagem para ser executada pelo programa. Todos os comandos tem que terminar com um ponto-e-vírgula (;).

Output

A palavra reservada (cout <<) permite mostrar dados ao usuário;

Sintaxe

cout << “lorem ipsum”; @imprime na tela a string literal
cout << var_name; @imprime na tela o valor na variável

Exemplo

cout << “sistema indisponível”;

Input

A palavra reservada (cin >>) permite receber dados do usuário.

Sintaxe

cin >> var_name

Exemplo

a:string;
cin >> a; @atribui os caracteres digitados pelo usuário para a variável

Estruturas de Decisão

As estruturas de decisão define para onde o código deve seguir a partir de uma dada expressão, controlando assim a execução de um bloco de código ou outro.

Sintaxe

```
if(expressão) {
        … @ caso a expressão retorna um valor verdadeiro este bloco é executado
}
else {
        … @ caso a expressão retorna um valor falso este bloco é executado
}
```

Decisão Simples

Exemplo

```
if(z == k) {
        … @ se o valor de z for igual a k, este bloco é executado
}
else {
        … @ se o valor de z for diferente, este bloco é executado
```

}

Decisão Composta

Exemplo

```
if(y != 0) {
        … @ se o valor de y for diferente de 0 este bloco é executado
}
else {
        … @ se o valor de y for igual a 0 este bloco é executado
}
```

Decisão Encadeada

Exemplo

```
if(a >= 1) {
        … @ se o valor de a for maior ou  igual a 1 esse bloco é executado
}
else {
        … @ se o valor de a for menor que 1 esse bloco é executado
}
if(b == 2) {
        … @ se o valor de b for maior ou  igual a 2 esse bloco é executado
}
else {
        … @ se o valor de b for menor que 2 esse bloco é executado
}
if(c >= 3) {
        … @ se o valor de c for maior ou  igual a 3 esse bloco é executado
}
else {
        … @ se o valor de c for menor que 3 esse bloco é executado
}
```

Estruturas de Repetição

A estrutura de repetição for e while executa o bloco de código até que a dada expressão seja cumprida.

FOR

O **for** e a única estrutura de repetição que dispõe de operadores de **incremento** e **decremento** em sua estrutura.

Sintaxe

```
for(expressão) {
        … @bloco de código será repetido enquanto a expressão for cumprida
}
```

Exemplo : for

O incremento e o decremento são seguidos por um número inteiro, que define os passos do for.

```
for(i=0; i <= 3; ++1) { @for com operador incremento de passo 1
        …
}

for(y=100; y >= 5; --3) { @for com operador decremento de passo 3
        …
}
```

WHILE

Sintaxe

```
while(expressão) {
```

```
        … @bloco de código será repetido enquanto a expressão for cumprida
}
```

Exemplo

```
while(x != 0) {
        … @bloco de código será repetido enquanto x for diferente de 0
}
```

DO WHILE

Sintaxe

do{...} while (expressão)

Exemplo

do{ cout << “hello world”; x = x +1ç } while(x <= 5) @imprime mensagem até que x seja diferente de 0

Funções

As funções são uma das partes mais poderosas de uma linguagem, permitindo aproveitar uma grande linha de códigos e até dividi los em módulos.

Sintaxe de Definição de Função

Restrição: 20 caracteres para o nome da função, ao ultrapassar esse valor gera um erro léxico.

```
tipo_retorno name_func ( parâmetros ) {
        inicio
                …. @bloco da função
        fim
};
```

Sintaxe de Uso de Função já Declarada

name_func(parâmetros);

Exemplo de Definição de Função

```
string say_hello (  ) { @exemplo 1
        inicio
```

```
        return “ola”;
    fim
};

int somador ( integer ; integer  ) { @exemplo 2
    inicio
        return x + y;
    fim
};
```

Exemplo de Uso de Função já Declarada

```
say_hello(); @exemplo 1

somador( 3, 3 ); @exemplo 2
```

Comentários

Os comentários existem para ajudar na documentação do código facilitando a leitura e compreensão do código quando ele é feito por outra pessoa. Indicando um comentário os caracteres pertencentes não serão interpretados pelo compilador, sendo totalmente ignorados e não gerando linguagem de máquina ao final do processo.

Existem dois tipos de comentários o em linha e bloco, ambos não possuem limites de caracteres. O comentário em linha se estende por apenas uma linha enquanto o em bloco pode se estender por várias. Não é possível a declaração de um comentário dentro de outro.

Comando de comentário em linha:

Sintaxe: // texto do comentário;

Exemplo:

```
// este é um comentário em linha;
```

Comando de comentário em bloco

Sintaxe: /* texto do comentário */

Exemplo:

```
/*      primeiro comentário
        segundo comentário
*/
```

Changelog

- Tags dos comentários em linha alterado para “//”;
- Correção feita na introdução onde dizemos que todo corpo do programa iniciava com a palavra reservada **início** e terminava com a palavra **fim**;
- Retirada todas variáveis **int,** existe apenas integer correção feita tanto no documento quanto no léxico;
- Definida a pontuação específica para números do tipo **float**;
- Operadores **++** e **--** retirados da documentação, eles existem apenas para os laços de repetição;
- Adicionada uma seção na documentação para explicação dos **literais**;
- Detalhamento das estrutura de decisão no documento, complementando com os tipos simples, composta e encadeada;
- Exemplos retirados do documento e transformados em arquivos texto separados para serem importados no analisador léxico;