Universidade Federal de Alagoas
Instituto de Computação
Ciência da Computação
Período 2019.1

Linguagem F1

Especificação da linguagem de programação

Aluno: França Mac Dowell da Silva Sales
Professor: Alcino Dall'Igna

Forma geral do código da linguagem F1

Os arquivos fonte da linguagem F1 deverão ter a extensão “.fum” para serem reconhecidos. Para execução de qualquer programa, será necessário uma função principal, onde a mesma servirá de ponto de partida do programa e deverá ter um *return* ao fim do programa. Exemplo:

```
func void main():
      ...
      return 0
end
```

As demais funções devem ser definidas acima desta função *main*.

Como podemos observar no exemplo acima, os escopos são iniciados a partir do sinal “**:**” e finalizados com a palavra reservada “**end**”. A indentação é opcional, porém é recomendada para melhor legibilidade do código. Os comentários da linguagem são feitos por linha e são posicionados após o símbolo “**@**”.

Características Léxicas da linguagem

Suas funções e variáveis são nomeadas por identificadores. Esses identificadores só podem ser declarados por letras, não podem coincidir com as palavras reservadas da linguagem e não são permitidos caracteres especiais.

Ao nomear identificadores, letras maiúsculas e minúsculas fazem diferença, ou seja, são case-sensitive. O tamanho da palavra usada como identificador não pode superar os 20 caracteres.

Especificação de tipo de dados

- **int**
- **float**
- **bool**
- **char**
- **str**

Função principal

- **main**

Funções de entrada e saída de dados

- **input**
- **print**

Retornar de uma função

- **return**

Comandos de condição

- **if**
- **else**
- **elif**

Comandos de iteração

- **while**
- **for**

Operadores lógicos

- **and**
- **or**
- **not**

Variáveis e seus Tipos

Na linguagem F1 não há variáveis globais, portanto elas devem estar declaradas dentro de alguma função e são vinculadas estaticamente, dessa forma no momento de sua declaração o tipo deve ser especificado.

As variáveis podem ser do tipo:

- **float:** ponto flutuante;
- **int:** inteiro;
- **bool:** booleano;

- **char:** caractere;
- **str:** cadeia de caracteres.

É possível a criação de arrays na linguagem F1, os arrays são para todos os tipos acima e apenas um tipo por variável. A declaração de array é feita de seguinte forma:

tipo[*tamanho****] id***

O acesso será através de índices dentro dos colchetes, e o índice vai de 0 a *tamanho - 1*:

id[0] @acessando o array *id* no índice zero.

Coerção

Tipo String e um tipo qualquer

Toda operação de concatenação de uma variável tipo string com uma variável de outro tipo, haverá coerção deste segundo tipo para string.

Tipo Inteiro e Tipo Flutuante

Quando houver alguma operação entre uma variável de valor inteiro e uma de valor de ponto flutuante, a variável inteira terá o valor de sua parte decimal mantido em zero, e sua parte inteira idêntica à variável inteira.

Já em caso de um valor em ponto flutuante ser atribuído a uma variável inteira, esta armazenará apenas a parte inteira do valor, desprezando a parte decimal independente de seu valor.

Escopo

As variáveis são visíveis apenas na função em que foram criadas.

Operadores

Aritméticos

+	Adição
-	Subtração

*	Multiplicação
/	Divisão
^	Exponenciação
-	Unário negativo

Relacionais

>	Maior que
<	Menor que
>=	Maior ou igual
<=	Menor ou igual
==	Igual
!=	Diferente

Lógicos

not	Negação
and	Conjunção
or	Disjunção

Concatenação de string

&	Concatenação

Atribuição

=	Atribuição

Operações suportadas

int	atribuição, aritméticos e relacionais
float	atribuição, aritméticos e relacionais
bool	atribuição, aritméticos e relacionais
char	atribuição, aritméticos e relacionais (de igualdade)
str	atribuição, relacionais e concatenação

Precedência e Associatividade

A precedência é dada pela mais alta até a mais baixa na ordem da tabela a seguir. As regras de associatividade é sempre da esquerda para direita, exceto para o operador de exponenciação. Neste caso particular será da direita para esquerda.

not
- (unário negativo)
^
*, /
+, -, &
<, <=, >, >=, ==, !=
and, or
=

Utilizando parênteses para priorizar algum conjunto de operações, as operações dentro deles sempre serão resolvidas primeiro.

Funções

As funções são da seguinte forma:

```
func tipo_de_retorno nome_da_funcao(tipo parametro_1, ...):
    ...
    return ...
end
```

Toda função começa com a palavra reservada ***func***, seguida do tipo de retorno, nome da função, parâmetros, comandos e então a palavra reservada ***end*** que significa fim dos comandos daquela função.

O programa sempre começará a execução a partir da função com nome "**main**", funções aninhadas não são permitidas. Cada função deve ser declarada separadamente.

É obrigatório o uso da palavra reservada ***return*** para retornar o valor do tipo especificado em seu cabeçalho, com exceção de quando o tipo de retorno for especificado como void. São permitidos como retorno qualquer tipo primitivo e arranjos unidimensionais.

As chamadas de função são do formato:

```
nome_da_funcao(tipo parametro_1, ...)
```

O modelo de passagem de parâmetro adotado para os tipos primitivos é por passagem de valor, já para arranjos unidimensionais é de passagem por referência.

Instruções

Estrutura condicional

Estrutura iterativa com controle

Estrutura iterativa controlada por contador

Entrada

Saída

Exemplos

Linguagem de implementação

Categoria dos Tokens