Memória

A memória é uma das estruturas que vão permitir ao parser o armazenamento dos diferentes símbolos e respetivo valor. Esta encontra-se definida da seguinte forma:

* Indentificador - Nome da Variável
* Valor - Correspondente ao valor atribuído à variável
* Nível – Scope da variável
* Tipo – Tipo da variável (inteiro, real, texto, caracter ou lógico)

Para podermos adicionar uma nova variável à memória, podemos realizar de 3 formas diferentes:

* Através da **atribuição** numa forma **processo**:

Figura 1- Atribuição em processos

a = “oi”,

b = 1,

c = ‘c’,

d = 1.5,

e = true

A atribuição pode ser realizada indicando a sintaxe correta de cada símbolo respetivo, ou caso não seja indicado, será deduzido pelo parser consoante as seguintes diretivas:

Figura 2 - Dedução do tipo de variável

a = oc, <- Texto

b = 1, <- Inteiro

c = c, <- Caracter

d = 1.5, <- Real

e = true <- Lógico

* Através da **atribuição** numa forma **Leitura**:

b

Figura 3 - Leitura de variável

A dedução do tipo de variável que foi introduzido é processado da mesma forma que no caso da Figura 2.

* Através da **atribuição** numa forma **Função/processo**:

Figura 4 – Atribuição em funções

a = fact(a)

e = true <- Lógico

Os parâmetros de uma função (**ex: fact(***a***)** )são necessariamente símbolos, valores ou funções, nas atribuições deverá dar erro.

Sempre que uma variável é declarada e atribuído um valor a esta, o seu símbolo não pode ser alterado futuramente (cast não possível).

Construção da Memória

A memória para os diferentes níveis de execução vai sendo construída da seguinte forma:

|  |  |  |
| --- | --- | --- |
| **Identificador** | **Valor** | **Nível** |
| - | - | - |

|  |  |  |
| --- | --- | --- |
| **Identificador** | **Valor** | **Nível** |
| a | 5 | 0 |

|  |  |  |
| --- | --- | --- |
| **Identificador** | **Valor** | **Nível** |
| a | 5 | 0 |
| b | 5 | 0 |

|  |  |  |
| --- | --- | --- |
| **Identificador** | **Valor** | **Nível** |
| a | 5 | 0 |
| b | 5 | 0 |
| c | 10 | 0 |

|  |  |  |
| --- | --- | --- |
| **Identificador** | **Valor** | **Nível** |
| - | - | - |

Inicio

a = 5

b

c = a+b

c

Fim

5

Sempre que numa variável válida é atribuída um valor, é guardada na memória.

Tanto na forma início como no final do fluxograma a memória deverá ser limpa de todas varáveis existentes.

Nota: Na disjunção de fluxo, será necessário surgir uma caixa de texto indicando o sentido do fluxo (ex: true ou false)

No mesmo caso vai ser incrementado o valor do nível para os próximos símbolos

Este valor é decrementado quando acontece uma junção limpando todos os símbolos existentes nesse mesmo nível.

|  |  |  |
| --- | --- | --- |
| **Identificador** | **Valor** | **Nível** |
| - | - | - |

Inicio

a = 5

b

c = func(a)

c

Fim

|  |  |  |
| --- | --- | --- |
| **Identificador** | **Valor** | **Nível** |
| a | 4 | 0 |

|  |  |  |
| --- | --- | --- |
| **Identificador** | **Valor** | **Nível** |
| a | 4 | 0 |
| b | 2 | 0 |

2

Figura - Memória 1

*Figura 6 - Memória 2*

|  |  |  |
| --- | --- | --- |
| **Identificador** | **Valor** | **Nível** |
| a | 5 | 0 |
| b | “impar” | 1 |

a = a + 1

func(a)

a % 2 == 1

b = “par”

**True**

**False**

b = “impar”

Fim

Devolve “ímpar”

|  |  |  |
| --- | --- | --- |
| **Identificador** | **Valor** | **Nível** |
| a | 4 | 0 |

|  |  |  |
| --- | --- | --- |
| **Identificador** | **Valor** | **Nível** |
| a | 5 | 0 |

*Figura 7 - Memória 2*

Figura 8 - Memória 1

|  |  |  |
| --- | --- | --- |
| **Identificador** | **Valor** | **Nível** |
| - | - | - |

|  |  |  |
| --- | --- | --- |
| **Identificador** | **Valor** | **Nível** |
| a | 4 | 0 |
| b | 2 | 0 |
| c | “impar” | 0 |

Execução:

1. O fluxograma inicia com a memória vazia
2. É criada a variável *a*, atribuída o valor 5 e o tipo inteiro.
3. É criada a variável *b*, atribuída o valor 2 (introduzido pelo utilizador) e tipo inteiro.
4. É criada a variável c e inicia a execução a execução da função *func(a)*
   1. É criada uma memória vazia para a função
   2. É criada a variável b e atribuído o valor de *a* ( 5 )
   3. É verificado se o resto da divisão de *b* corresponde a, o que verifica-se
   4. É criada a variável ae atribuído o texto *“impar”*
   5. Termina a função e retorna *”impar”*
5. Ao valor de c é atribuído o texto devolvido pela função (“ímpar”)
6. Imprime a variável c
7. Termina o fluxograma

|  |  |  |
| --- | --- | --- |
| **Identificador** | **Valor** | **Nível** |
| - | - | - |

|  |  |  |
| --- | --- | --- |
| **Identificador** | **Valor** | **Nível** |
| x | 6 | 0 |

Figura 9 - Memória nivel 0

|  |  |  |
| --- | --- | --- |
| **ID** | **valor** | **Nível** |
| x | 6 | 0 |
| k | 1 | 1 |

Figura 10- Memória nivel 1

|  |  |  |
| --- | --- | --- |
| **ID** | **Valor** | **Nível** |
| x | 5 | 0 |
| k | 1 | 1 |
| y | 0 | 2 |

Figura 11 - Memória nivel 2

|  |  |  |
| --- | --- | --- |
| **ID** | **Valor** | **Nível** |
| x | 5 | 0 |
| k | 1 | 1 |
| y | 0 | 2 |
| z | 5 | 3 |

Figura 12 - Memória nivel 3

**True**

y = 0

K = 1

k

Inicio

x

6

x < 0

**False**

**True**

x < 5

**False**

x > 5

**True**

**False**

z = x

z = x -1

Fim

Execução:

1. O fluxograma inicia com a memória vazia
2. É criada a variável *x,* atribuída o valor 6 (introduzido pelo utilizador) e o tipo inteiro
3. É verificado se o valor de *x* é menor que 0, o que não se verifica
4. É verificado se o valor de xé maior que 5, o que se verifica
5. E criada a variável z*,* atribuída o valor de *x* ( 5 ) e tipo inteiro
6. Termina o fluxograma