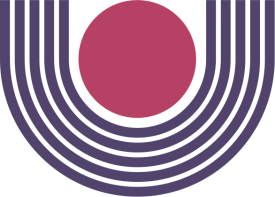
**UNIOESTE – Universidade Estadual do Oeste do Paraná**

Campus de Cascavel PR - Rua Universitária, 1619

(45) 3220-3000

**Bacharelado em Ciência da Computação - CCET**

Colegiado de Ciência da Computação

(45) 3220-3191

|  |  |
| --- | --- |
| Disciplina: | **Linguagens de Programação** |
| Alunos: | **Juliano Felipe da Silva** | |
| **Maycon de Queiroz Oliveira** | |

Descrição da linguagem de programação NOME

# Objetivo

# Operadores

## Operadores lógicos suportados

|  |  |  |
| --- | --- | --- |
| Símbolo Utilizado | Operação Realizada | Exemplo de chamada |
| & | E bit a bit | C = A & B |
| | | OU bit a bit | C = A | B |
| ! | Negação | C = !A |
| ^ | OU exclusivo bit a bit | C = A ^ B |
| << | Shift lógico a esquerda onde o elemento a esquerda sofre a quantidade apresentada à direita do operador | C = A << B |
| >> | Shift lógico a direita onde o elemento a esquerda sofre a quantidade apresentada à direita do operador | C = A >> B |

## Operadores aritméticos suportados

|  |  |  |
| --- | --- | --- |
| Símbolo Utilizado | Operação Realizada | Exemplo de chamada |
| + | Soma | C = A + B |
| - | Subtração | C = A - B |
| \* | Multiplicação | C = A \* B |
| / | Divisão | C = A / B |
| % | Resto da divisão do elemento da direita pelo elemento da esquerda | C = A % B |

## Operadores relacionais suportados

|  |  |  |
| --- | --- | --- |
| Símbolo Utilizado | Operação Realizada | Exemplo de chamada |
| == | Verdadeiro se elementos a direita e a esquerda são iguais | A == B |
| != | Verdadeiro se elementos a direita e a esquerda são diferentes | A != B |
| <= | Verdadeiro se elemento a esquerda é menor ou igual ao elemento a direita | A <= B |
| >= | Verdadeiro se elemento a esquerda é maior ou igual ao elemento a direita | A >= B |
| < | Verdadeiro se elemento a esquerda é menor que o elemento a direita | A < B |
| > | Verdadeiro se elemento a esquerda é maior que o elemento a direita | A > B |

# Tipos de dados

A declaração de variáveis é explicita, com o tipo informado logo à frente da variável a ser declarada. Os possíveis tipos e suas declarações são listados abaixo.

## CHR

|  |  |
| --- | --- |
| Tamanho | 1 byte |
| Operações Permitidas | +, -, \*, /, %, &, |, !, ^, <<, >> |
| Chamada em impressão | %c – Imprime caractere de acordo com tabela ASCII  %u – Imprime valor numérico do byte |
| Descrição | Valor que pode ser tanto caractere como inteiro |

## INT

|  |  |
| --- | --- |
| Tamanho | 4 bytes |
| Operações Permitidas | +, -, \*, /, %, &, |, !, ^, <<, >> |
| Chamada em impressão | %d – Imprime valor numérico |
| Descrição | Valor inteiro |

## FLT

|  |  |
| --- | --- |
| Tamanho | 4 bytes |
| Operações Permitidas | +, -, \*, /, %, &, |, !, ^ |
| Chamada em impressão | %f – Imprime valor numérico |
| Descrição | Valor ponto flutuante |

## BLN

|  |  |
| --- | --- |
| Tamanho | 1 byte (usado apenas bit menos significativo) |
| Operações Permitidas | &, |, !, ^, <<, >> |
| Chamada em impressão | %b – Imprime valor numérico (1 ou 0) |
| Descrição | Valor para uso booleano, 1 para verdadeiro e 0 para falso |

## EBNF

|  |  |
| --- | --- |
| <decl\_stmt> → | chr <id>; | int <id>; | flt <id>; | bln <id>; |
| <int> → | <digi>{<digi>} |
| <flt> → | <digi>{<digi>}’,’<digi>{<digi>} |
| <bln> → | 0 | 1 |

# Estruturas de desvios

## Desvios Incondicionais

### GTO

Salta para uma linha do código denominada por uma label. Chamada sem parâmetros é inválida. Exemplo: gto label1

### BRK

Para a execução do laço de repetição. Chamada com parâmetros ou sem laço é inválida. Exemplo: brk

## Desvios condicionais

### IFF

Executa o código dentro do bloco posterior se a expressão neste for verdadeira. Expressões vazias são inválidas. Um bloco posterior deve ser definido usando chaves. Exemplo:

iff (A < B){

}

### ELS

Executa o código dentro do bloco posterior se a expressão ‘iff’ acima for falsa. Expressões são inválidas. Um bloco posterior deve ser definido usando chaves. Exemplo:

iff(A < B){

} els {

}

### EIF

Executa o código dentro do bloco posterior se a expressão ‘iff’ acima for falsa, mas a expressão neste for verdadeira. Expressões vazias são inválidas. Um bloco posterior deve ser definido usando chaves. Exemplo:

iff (A < B){

} eif (A == B){

}

## EBNF

|  |  |
| --- | --- |
| <cond\_stmt> → | iff’(‘<lexp>’)’’{‘<stmt>’}’’[‘eif’(‘<lexp>’)’’{‘<stmt>’}’’]’’[‘else ‘{‘<stmt>’}’’]’ | gto <lbel>; |

# Estruturas de repetição

## for

Executa o código dentro do bloco posterior de acordo com as condições iniciais, final e de alteração. Um bloco posterior deve ser definido usando chaves. Exemplo:

for(A = 0; A < 10; A = A + 1){

}

## whl

Executa o código no bloco posterior enquanto a expressão for válida. Expressões vazias são inválidas. Um bloco posterior deve ser definido usando chaves. Exemplo:

whl(A == 0){

}

## EBNF

|  |  |
| --- | --- |
| <rept\_stmt> → | for’(‘<atrb>; <lexp>; <atrb>’)’’{‘<stmt>’}’ | whl’(‘<lexp>’)’’{‘<stmt>’}’ |

# Definições

## Palavras reservadas

Todos os comandos são palavras reservadas. As palavras A00 a A99 (de 00 a 99 sucedendo a letra A maiúscula) são palavras reservadas.

## Nomes de variáveis

Os nomes de variáveis podem ter até 255 caracteres e devem iniciar com uma letra. Os nomes de variáveis são sensíveis a alteração de letras maiúsculas e minúsculas.

### EBNF

|  |  |
| --- | --- |
| <digi>→ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| <letr>→ | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z |
| <id> → | <letr>{<letr> | <digi>} |

# EBNF

## Expressões Lógicas

|  |  |
| --- | --- |
| <lexp> → | (<id> | <int> | <chr> | <flt> | <bln>) (== | != | > | < | <= | >=) (<id> | <int> | <chr> | <flt> | <bln>) NÃO TERMINADO! |

Expressões aritméticas, Operadores sobrecarregados, Conversões de tipos, Expressões relacionais e booleanas e Avaliação em curto-circuito. (Os slides do Cap07 está junto na pasta das aulas de LP no Dropbox)