PARADIGMAS DE PROGRAMAÇÃO
EXERCÍCIOS LISTA 05: **EXPRESSÕES E SENTENÇAS DE ATRIBUIÇÃO**
Prof.ª Lucília Ribeiro

01 Defina:

a) precedência de operador
b) associatividade de operador
c) efeito colateral funcional
d) coerção
e) operador sobrecarregado
f) conversões de alargamento e de estreitamento
g) cast

02 O que é avaliação em curto-circuito? Exemplifique.

03 Como a linguagem C oferece suporte para expressões relacionais e booleanas?

04 Assuma as seguintes regras de associatividade e de precedência para expressões:

Precedência	Mais alta	*, /, **not**
		+, – &, **mod**
		– (unário)
		=, /=, <, <=, >=, >
		and
	Mais baixa	**or**, **xor**
Associatividade	Esquerda para a direita	

Mostre a ordem de avaliação das seguintes expressões por meio do uso de parênteses em todas as subexpressões e colocando um expoente no parêntese direito para indicar a ordem.

Por exemplo, para a expressão
`a + b * c + d`, a ordem de avaliação seria representada como
`( (a + (b * c)1 )2 + d)3`

```
a) a * b – 1 + c
b) a * (b – 1) / c mod d
c) (a – b) / c & (d * e / a – 3)
d) –a or c = d and e
e) a > b xor c or d <= 17
f) –a + b
```

05 Mostre a ordem de avaliações das expressões do exercício anterior, assumindo que não existem regras de precedência e todos os operadores são associativos da direita para a esquerda.

```
a) a * b - 1 + c
b) a * (b - 1) / c mod d
c) (a - b) / c & (d * e / a - 3)
d) -a or c = d and e
e) a > b xor c or d <= 17
f) -a + b
```

06 Escreva uma descrição em BNF para as regras de precedência e de associatividade definidas para as expressões do exercício 4. Assuma que os únicos operandos são os nomes a, b, c, d e.

```
<expr>→ <expr> or <e1> | <expr> xor <e1> | <e1>
<e1>→ <e1> and <e2> | <e2>
<e2>→ <e2> = <e3> | <e2> /= <e3> | <e2> < <e3>
      | <e2> <= <e3> | <e2> > <e3> | <e2> >= <e3> | <e3>
<e3>→ <e4>
<e4>→ <e4> + <e5> | <e4> - <e5> | <e4> & <e5>
      | <e4> mod <e5> | <e5>
<e5>→ <e5> * <e6> | <e5> / <e6> | not <e5> | <e6>
<e6>→ a | b | c | d | e | const | (<expr>)
```

07 Usando a gramática do exercício 6, desenhe árvores de análise sintática para as expressões do exercício 4.

08 Rode o código a seguir em C para determinar os valores de soma1 e soma2. Explique os resultados.

```
#include <stdio.h>

int fun(int *k){
 *k += 4;
 return 3 * (*k) - 1;
}

int main(){
  int i = 10, j = 10, soma1, soma2;
  soma1 = (i / 2) + fun(&i);
  soma2 = fun(&j) + (j / 2);
  printf("\nsoma1 = %d\n", soma1);
  printf("\nsoma2 = %d\n", soma2);
}
```

09 Reescreva o programa do exercício 8 em C++ e Java. Execute-os e compare os resultados.

10 Escreva um programa em linguagem C que tenha as seguintes sentenças:

```
int a, b;
a = 10;
b = a + fun();
printf("Com a chamada a função pela direita, ");
printf("b = %d\n", b);
a = 10;
b = fun() + a;
printf("Com a chamada a função pela esquerda, ");
printf("b = %d\n", b);
```

11 Escreva um programa em Java ou C++ que realize um grande número de operações de ponto flutuante e um número igual de operações sobre inteiro e compare o tempo necessário.