## QUESTÕES DE REVISÃO

1. O que é um descritor?
2. Quais são as vantagens e as desvantagens dos tipos de dados decimais?
3. Quais são as questões de projeto relativas aos tipos cadeia de caracteres?
4. Defina as três opções de tamanho de cadeia.
5. Defina o que são tipos *ordinal, enumeração* e *subfaixa*.
6. Quais são as vantagens dos tipos enumeração, definidos pelo usuário?





7. Quais são as questões de projeto relativas às matrizes?
8. Defina o que são matrizes estáticas, fixas dinâmicas na pilha, dinâmicas na piha e dinâmicas. Quais as vantagens de cada uma?
9. Qual recurso de inicialização de matrizes está disponível na Ada, mas não está disponível em outras linguagens imperativas comuns?
10. O que é uma constante agregada?
11. Quais operações de matrizes são oferecidas especificamente para matrizes unidimensionais na Ada?
12. Quais são as diferenças entre as fatias (*slices*) do FORTRAN 90 e as da Ada?
13. Defina *ordem de linha maior* e *ordem de coluna maior.*
14. O que é uma função de acesso para uma matriz?
15. Quais são as entradas exigidas em um descritor de matriz Pascal, e quando elas devem ser armazenadas (no tempo de compilação ou no de execução)?
16. Qual é o propósito dos números de nível nos registros COBOL?
17. Defina *referências amplamente qualificadas* e *referências elípticas* a campos em registros.
18. Defina *união, união livre* e *união driscriminada.*
19. Quais são as questões de projeto relativas às uniões?
20. Quais são os dois problemas com as uniões do Pascal?
21. De que maneira as uniões da Ada são mais seguras do que as do Pascal?
22. Por que normalmente há severas restrições quanto ao tamanho dos conjuntos nas implementações Pascal?
23. Quais são as questões de projeto relativas aos tipos ponteiro?
24. Quais são os dois problemas comuns com os ponteiros?
25. Quais são as duas maneiras segundo as quais os ponteiros da Ada são mais seguros do que os do Pascal?
26. Por que os ponteiros da maioria das linguagens restringem-se a apontar para um objeto de tipo único?
27. O que é um tipo referência C++ e qual é seu uso comum?
28. Por que as variáveis de referência no C++ são melhores do que os ponteiros para parâmetros formais?
29. Quais vantagens as variáveis de tipo referência do Java têm sobre os ponteiros de outras linguagens?
30. Descreva as abordagens preguiçosa e ansiosa de reivindicar o lixo.
31. Quais são as diferenças entre as variáveis de referência do C++ e do Java?
32. Por que a aritmética em referências Java não faria sentido?

1. Quais são os argumentos favoráveis e contrários à representação dos valores booleanos como bits únicos na memória?
2. Por que um valor decimal gasta tanto espaço de memória?
3. O COBOL usa diversos métodos diferentes para armazenar números decimais. Explique o formato e o propósito de cada um.
4. Os microcomputadores VAX usam um formato para números de vírgula-flutuante que não é o mesmo que o padrão IEEE. Qual é esse formato e por que ele foi escolhido pelos projetistas dos computadores VAX? Uma referência para as representações do vírgula-flutuante VAX encontra-se em Sebesta (1991).
5. Compare os métodos das lápides e das chaves-e-fechaduras (*locks-and-keys*) para evitar ponteiros pendurados, do ponto de vista da segurança e do custo de implementação.
6. Projete um conjunto de programas de teste simples para determinar as regras de compatibilidade de tipos de um compilador Pascal ou C ao qual você tem acesso. Escreva um relatório de suas descobertas.
7. Quais desvantagens há no desreferenciamento implícito de ponteiros, mas somente em certos contextos? Por exemplo, considere o desreferecimento implícito de um ponteiro para um registro em Ada quando ele é usado para referenciar um campo de registro.





8. Qual justificativa importante há para o operador -> no C e no C++?
9. Determine qual opção de implementação descrita na Seção 6.9.4 é usada por algum compilador Pascal ao qual você tem acesso?
10. As uniões no C e no C++ são separadas dos registros dessas linguagens, em vez de serem combinadas como são no Pascal e na Ada. Quais são as vantagens e as desvantagens dessas duas opções?
11. Determine se algum compilador Pascal que você tem acesso implementa o procedimento `dispose`.
12. Determine se algum compilador C ao qual você tem acesso implementa a função `free`.
13. Suponhamos que uma linguagem inclua tipos-enumeração definidos pelo usuário e que valores de enumeração possam ser sobrecarregados; ou seja, o mesmo valor literal poderia aparecer em dois tipos-enumeração diferentes, como em:

```
type
  cores = (vermelho, blue, verde);
  humor = (feliz, brabo, blue);
```

O uso da constante `blue` não pode ser verificado quanto ao tipo. Proponha um método para permitir essa verificação de tipos sem desativar completamente essa sobrecarga.
14. Matrizes multidimensionais podem ser armazenados em ordem de linha maior, como no Pascal, ou em ordem de coluna maior, como no FORTRAN. Desenvolva as funções de acesso para ambos os arranjos para matrizes tridimensionais.
15. Na linguagem Burroughs Extended ALGOL, as matrizes são armazenadas como um vetor unidimensional de ponteiros para as linhas da matriz, que são tratados como vetores unidimensionais de valores. Quais são as vantagens e as desvantagens desse esquema?
16. Escreva um programa que faz a multiplicação de matrizes em alguma linguagem que faça verificação da faixa de subscrito e para a qual você possa obter uma versão de linguagem de montagem ou de linguagem de máquina do compilador. Determine o número de instruções necessárias para a verificação da faixa de subscrito e compare-o com o número total de instruções para o processo de multiplicação de matrizes.
17. Escreva um programa Pascal que inclua as seguintes declarações:

```
var
```

## QUESTÕES DE REVISÃO

1. Defina precedência de operadores e associatividade de operadores.
2. Defina efeito colateral funcional.
3. O que é uma coerção?
4. O que é uma expressão condicional?
5. O que é um operador sobrecarregado?
6. Defina conversões de estreitamento e de alargamento.
7. O que é uma expressão de modo-misto?
8. Como a ordem de avaliação de operandos interage com efeitos colaterais funcionais?
9. O que é avaliação curto-circuito?
10. Cite uma linguagem que sempre faça avaliação curto-circuito de expressões booleanas. Cite uma que nunca a faça. Cite uma em que seja permitido ao programador escolher.
11. Como o C suporta expressões relacionais e booleanas?
12. Qual é o propósito de um operador de atribuição composto?
13. Qual é a associatividade dos operadores aritméticos unários do C?
14. Qual é uma das possíveis desvantagens de tratar o operador de atribuição como se ele fosse um operador aritmético?
15. Quais atribuições de modo misto são permitidas na Ada?
16. Quais atribuições de modo misto são permitidas no Java?





## PROBLEMAS

1. Quando você poderia querer que o compilador ignorasse diferenças de tipos em uma expressão?
2. Apresente seus próprios argumentos contrários e favoráveis a permitir expressões aritméticas de modo misto.
3. Você acha que a eliminação de operadores sobrecarregados de sua linguagem favorita seria benéfica? Por que sim e por que não?
4. Seria uma boa idéia eliminar todas as regras de precedência de operadores e exigir parênteses para mostrar a precedência desejada nas expressões? Por que sim e por que não?
5. As operações de atribuição do C (por exemplo, +=) devem ser incluídas em outras linguagens? Por que sim e por que não?
6. As formas de atribuição de único operando do C (por exemplo, `++cont`) devem ser incluídas em outras linguagens? Por que sim e por que não?
7. Descreva uma situação em que o operador de adição em uma linguagem de programação não seria comutativo.
8. Descreva uma situação em que o operador de adição em uma linguagem de programação não seria associativo.
9. Escreva um segmento de programa Pascal, usando uma construção **`while`** para procurar em um vetor de números inteiros um número inteiro particular, que funcionaria mesmo que uma avaliação curto-circuito de expressões booleanas não fosse feita.

Suponha as seguintes regras de associatividade e precedência para expressões:

| | | |
|---|---|---|
| *Precedência:* | *Mais alta* | `*, /, not` |
| | | `+, -,&, mod` |
| | | `-` (unário) |
| | | `=, /=, <, <=, >=, >` |
| | | `and` |
| | *Mais baixa* | `or, xor` |
| *Associatividade* | *Da esquerda para a direita* | |

## QUESTÕES DE REVISÃO

1. Qual é a definição de estrutura de controle?
2. Qual é a definição de bloco?
3. Quais são as questões de projeto relativas às estruturas de seleção?
4. Quais são as soluções comuns para o problema do aninhamento de seletores bidirecionais?
5. Quais são as questões de projeto referentes às instruções de seleção múltipla?
6. Qual é a base para as instruções de controle do FORTRAN I?
7. O que está errado com a instrução `IF` aritmética do FORTRAN?
8. O que é incomum a respeito da instrução de seleção múltipla do C? Qual compromisso de projeto foi feito?
9. Quais são as questões de projeto referentes às instruções de laço controladas por contador?
10. O que é uma instrução de laço pré-teste? E uma instrução pós-teste?
11. Qual é a mudança mais significativa na instrução `DO` do FORTRAN IV e do FORTRAN 90?
12. Qual característica da instrução **`for`** do ALGOL 60 torna os programas que a usam difíceis de serem lidos?
13. Qual é a diferença entre a instrução **`for`** do C++ e a do Java?
14. Quais são as questões de projeto referentes às instruções de laço controladas logicamente?
15. Qual é a principal razão para que instruções de controle de laço localizadas pelo usuário tenham sido inventadas?
16. Que vantagem a instrução **`exit`** da Ada tem sobre a instrução **`break`** do C?
17. Quais são as diferenças entre a instrução **`break`** do C++ e a do Java?
18. O que é um controle de iteração definido pelo usuário?
19. Quais são as duas desvantagens das variáveis de rótulos da PL/I?
20. Quais linguagens de programação comuns tomam emprestado parte de seu projeto dos comandos protegidos de Dijkstra?

## PROBLEMAS

1. Redija uma defesa da afirmação segundo a qual a instrução de seleção tridirecional do FORTRAN I foi a melhor opção, dadas as circunstâncias da época.
2. Imagine uma situação na qual a variável de rótulo da PL/I seria uma grande vantagem.
3. Descreva três situações em que uma construção de laço de contagem e lógico combinados seja necessária.
4. Compare a `GO TO` computada do FORTRAN com a instrução **`case`** Pascal, especialmente em termos de legibilidade e de confiabilidade.
5. Quais são as possíveis razões para que o Pascal tenha um **`laço`** pós-teste lógico, enquanto o ALGOL 60 não?





**d.** Expressões **`until`** são avaliadas uma vez na entrada do laço, e expressões **`step`** são avaliadas antes de cada iteração, logo depois que o contador de laços é incrementado.

Em todos os casos, quando mais de uma expressão é avaliada ao mesmo tempo, elas são avaliadas na ordem esquerda para a direita. Além disso, a atribuição é sempre feita assim que seu LD é avaliado.

14. Use a Science Citation Index para encontrar um artigo que se refira a Knuth (1974). Leia o artigo

## QUESTÕES DE REVISÃO

1. Quais são as três características gerais dos subprogramas?
2. O que significa um subprograma estar ativo?
3. O que é um perfil de parâmetro? O que é um protocolo de subprograma?
4. O que são parâmetros formais? O que são parâmetros reais?
5. Quais são as vantagens e as desvantagens dos parâmetros de palavra-chave?
6. Quais são as questões de projeto referentes aos subprogramas?
7. Quais são as vantagens e as desvantagens das variáveis locais dinâmicas?
8. Quais são os três modelos semânticos de passagem de parâmetros?
9. Quais são os modos, os modelos conceituais de transferência, as vantagens e as desvantagens dos métodos de passagem de parâmetros por valor, por valor-resultado, por referência e por nome?
10. De quais maneiras podem ocorrer apelidos com parâmetros passados por referência?
11. Qual é a diferença na maneira pela qual o C e o ANSI C lidam com um parâmetro real cujo tipo não é idêntico ao do formal correspondente?
12. Qual é o problema com a política da Ada de permitir que os implementadores decidam quais parâmetros passar pela referência e quais passar pelo valor-resultado?
13. Quais são as duas considerações de projeto fundamentais referentes aos métodos de passagem de parâmetros?
14. Quais são as duas questões que surgem quando nomes de subprograma são parâmetros?
15. Defina vinculação rasa e vinculação profunda para ambientes de referenciamento de subprogramas passados como parâmetros.
16. O que é um subprograma sobrecarregado?
17. O que é polimorfismo paramétrico?
18. O que faz uma função modelo C++ ser instanciada?
19. Defina compilação separada e compilação independente.
20. Quais são as questões de projeto referentes às funções?
21. De quais maneiras as co-rotinas são diferentes dos subprogramas convencionais?

## PROBLEMAS

1. Quais são os argumentos favoráveis e quais os contrários a que um programa crie definições adicionais para operadores existentes, como pode ser feito em Ada e em C++? Você acha que essa sobrecarga de operador definida pelo usuário é boa ou má? Sustente sua resposta.
2. Na maioria das implementações FORTRAN IV, os parâmetros eram passados pela referência usando somente a transmissão do caminho de acesso. Exponha tanto as vantagens como as desvantagens dessa opção de projeto.
3. Argumente em defesa da decisão dos projetistas da Ada 83 de permitirem que o implementador escolha entre implementar parâmetros no modo **entrada/saída** por cópia ou por referência.
4. O FORTRAN tem dois tipos ligeiramente diferentes de `COMMON`: em branco e nomeados. Uma diferença entre eles é que os blocos `COMMON` em branco não podem ser inicializados durante a compilação. Veja se você pode determinar a razão pela qual o `COMMON` em branco foi projetado dessa maneira. *Dica*: essa decisão de projeto foi tomada no início do desenvolvimento do FORTRAN, quando as memórias dos computadores eram bem pequenas.
5. Suponhamos que você deseje escrever um subprograma que imprima um cabeçalho em uma nova página de saída, juntamente com um número de página que é 1 na primeira ativação e que aumenta 1 em cada ativação subseqüente. Isso pode ser feito sem parâmetros e sem referência a variáveis não-locais em Pascal? Pode ser feito em FORTRAN? Pode ser feito em C?
6. O FORTRAN permite que um subprograma tenha múltiplas entradas. Por que isso, às vezes, é uma capacidade valiosa?
7. Escreva um procedimento Pascal `SOMADOR` que faça a adição de dois vetores de números inteiros. Ele deve ter somente dois parâmetros, os quais são os vetores a serem adicionados. O