01. [Sebesta, 2000] O que é um descritor?

02. [Sebesta, 2000] Quais são as vantagens e as desvantagens dos tipos de dados decimais?

03. [Sebesta, 2000] Quais são as questões de projeto relativas aos tipos *string* de caracteres?

04. [Sebesta, 2000] Defina as três opções de tamanho de *string*.

05. [Sebesta, 2000] Defina o que são tipos *ordinal*, *enumeração* e *subfaixa*.

06. [Sebesta, 2000] Quais são as vantagens dos tipos *enumeração*, definidos pelo usuário?

07. [Sebesta, 2000] Quais são as questões de projeto relativas aos *arrays*?

08. [Sebesta, 2000] Defina o que são *arrays estáticos*, *stack-dinâmicos fixos*, *stack-dinâmicos* e *dinâmicos*. Quais as vantagens de cada um?

09. [Sebesta, 2000] Qual recurso de inicialização de *arrays* está disponível na Ada, mas não está disponível em outras linguagens imperativas comuns?

10. [Sebesta, 2000] O que é uma constante agregada?

11. [Sebesta, 2000] Quais operações de *array* são oferecidas especificamente para *arrays* unidimensionais na Ada?

12. [Sebesta, 2000] Quais são as diferenças entre as fatias (*slices*) do FORTRAN 90 e as da Ada?

13. [Sebesta, 2000] Defina *ordem de linha maior* e *ordem de coluna maior*.

14. [Sebesta, 2000] O que é uma função de acesso para um *array*?

15. [Sebesta, 2000] Quais são as entradas exigidas em um descritor de *array* Pascal, e quando elas devem ser armazenadas (no tempo de compilação ou no de execução)?

16. [Sebesta, 2000] Qual é o propósito dos números de nível nos registros COBOL?

17. [Sebesta, 2000] Defina referências *amplamente qualificadas* e *referências elípticas* a campos em registros.

18. [Sebesta, 2000] Defina *união*, *união livre* e *união discriminada.*

19. [Sebesta, 2000] Quais são as questões de projeto relativas às uniões?

20. [Sebesta, 2000] Quais são os dois problemas com as uniões do Pascal?

21. [Sebesta, 2000] De que maneira as uniões da Ada são mais seguras do que as do Pascal?

22. [Sebesta, 2000] Por que normalmente há severas restrições quanto ao tamanho dos conjuntos nas implementações Pascal?

23. [Sebesta, 2000] Quais são as questões de projeto relativas aos tipos ponteiro?

24. [Sebesta, 2000] Quais são os dois problemas comuns com os ponteiros?

25. [Sebesta, 2000] Quais são as duas maneiras segundo as quais os ponteiros da Ada são mais seguros do que os do Pascal?

26. [Sebesta, 2000] Porque os ponteiros da maioria das linguagens restringem-se a apontar para um objeto de tipo único?

27. [Sebesta, 2000] O que é um tipo referência C++ e qual é seu uso comum?

28. [Sebesta, 2000] Por que as variáveis de referência no C++ são melhores do que os ponteiros para parâmetros formais?

29. [Sebesta, 2000] Quais vantagens as variáveis de tipo referência do Java têm sobre os ponteiros de outras linguagens?

30. [Sebesta, 2000] Descreva as abordagens preguiçosa e ansiosa de reivindicar o lixo.

31. [Sebesta, 2000] Quais são as diferenças entre as variáveis de referência do C++ e do Java?

32. [Sebesta, 2000] Por que a aritmética em referências Java não faria sentido?

33. [Sebesta, 2000] Quais são os argumentos favoráveis e contrários à representação dos valores booleanos como bits únicos na memória?

34. [Sebesta, 2000] Por que um valor decimal perde espaço de memória?

35. [Sebesta, 2000] O COBOL usa diversos métodos diferentes para armazenar números decimais. Explique o formato e o propósito de cada um.

36. [Sebesta, 2000] Os microcomputadores VAX usam um formato para números de ponto-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?

37. [Sebesta, 2000] Compare os métodos das *tombstones* e das chaves-e-fechaduras (l*ocks-and-keys*) para evitar ponteiros oscilantes, do ponto de vista da segurança e do custo de implementação.

38. [Sebesta, 2000] 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.

39. [Sebesta, 2000] Quais desvantagens há na valoração implícita de ponteiros, mas somente em certos contextos? Por exemplo, considere a valoração implícita de um ponteiro para um registro em Ada quando ele é usado para referenciar um campo de registro.

40. [Sebesta, 2000] Qual justificativa importante há para o operador `->` no C e no C++?

41. [Sebesta, 2000] Determine qual opção de implementação descrita na Seção 5.9.4 é usada por algum compilador Pascal ao qual você tem acesso?

42. [Sebesta, 2000] 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?

43. [Sebesta, 2000] Determine se algum compilador Pascal que você tem acesso implementa o procedimento `dispose`.

44. [Sebesta, 2000] Determine se algum compilador C ao qual você tem acesso implementa a função `free`.

45. [Sebesta, 2000] Suponhamos que uma linguagem inclua tipos-enumeração definidos pelo usuário e que valores de enumeração possam ser sobrecarregados (*overloaded*), 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.

46. [Sebesta, 2000] *Arrays* 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 *arrays* tridimensionais.

47. [Sebesta, 2000] Na linguagem Burroughs Extended ALGOL, as matrizes são armazenadas como um *array* unidimensional de ponteiros para as linhas da matriz, que são tratados como *arrays* unidimensionais de valores. Quais são as vantagens e as desvantagens desse esquema?

48. [Sebesta, 2000] 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 *assembly* 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.

49. [Sebesta, 2000] Escreva um programa Pascal que inclua as seguintes declarações:

```
var
  A, B : array [1..10] of integer;
  C : array [1..10] of integer;
  D : array [1..10] of integer;
```

Inclua no programa o código que determina, para cada *array*, qual dos outros três *arrays* são compatíveis com ele.

50. [Sebesta, 2000] Se você tiver acesso a um compilador em que o usuário pode especificar se a verificação de faixa de subscrito é desejada, escreva um programa que faça um grande número de acessos a matrizes e cronometre a execução deles. Rode o programa com verificação de faixa de subscrito e sem ela e compare os tempos.

51. [Sebesta, 2000] Analise e escreva uma comparação das funções `malloc` e `free` do C com os operadores `new` e `delete` do C++. Use a segurança como a principal consideração na comparação.

52. [Sebesta, 2000] Analise e escreva uma comparação do uso de ponteiros C++ e variáveis de referência Java para referir-se a variáveis *heap*-dinâmicas. Use a segurança e a conveniência como as principais considerações na comparação.

53. [Sebesta, 2000] Escreva uma breve discussão daquilo que foi ganho e daquilo que foi perdido na decisão dos projetistas do Java de não incluírem os ponteiros do C++.

54. [Sebesta, 2000] Quais são os argumentos favoráveis e os contrários à recuperação de armazenamento de *heap* implícito do Java, em comparação com a recuperação de armazenamento de *heap* explícito, exigido no C++.

55. [Sierra, 2004] Dado o código a seguir,

```
1.  public class {
2.  public static void main(String[] args) {
3.  X x = new X();
4.  X x2 = m1(x);
5.  X x4 = new X();
6.  x2 = x4;
7.  doComplexStuff();
8.  }
9.  static X mi(X mx) {
10. mx = new X();
11. return mx;
12. }
13. }
```

Depois que a linha 6 for executada, quantos objetos estarão qualificados para a coleta de lixo?

a) nenhum

b) 1

c) 2

d) 3

e) 4

56. [Sierra, 2004] Que declaração é verdadeira?

a) todos os objetos que estiverem qualificados para a coleta de lixo serão eliminados pelo coletor.

b) os objetos que tenham pelo menos uma referência nunca serão coletados como lixo.

c) os objetos de uma classe com o método `finalize()` substituído nunca serão coletados como lixo.

d) os objetos instanciados dentro de classes internas anônimas serão inseridos no *heap* de coleta de lixo.

e) depois que um método `finalize()` substituído for chamado, não haverá como tornar esse objeto qualificado para a coleta de lixo.

57. [Sierra, 2004] Dado o código a seguir,

```
1.  class X2 {
2.  public X2 x;
3.  public static void main(String[] args) {
4.  X2 x2 = new X2();
5.  X2 x3 = new X2();
6.  x2.x = x3;
7.  x3.x = x2;
8.  x2 = new X2();
9.  x3 = x2;
10. doComplexStuff();
11. }
12. }
```

Depois que a linha 9 for executada, quantos objetos estarão qualificados para a coleta de lixo?

a) nenhum

b) 1

c) 2

d) 3

e) 4

58. [Sierra, 2004] Que declaração é verdadeira?

a) chamar `Runtime.gc()` fará com que os objetos qualificados sejam coletados como lixo.

b) o coletor de lixo usa um algoritmo de marcação e eliminação.

c) se um objeto puder ser acessado por um segmento ativo, não poderá ser coletado como lixo.

d) se o objeto 1 referenciar o objeto 2, então, o objeto 2 não poderá ser coletado como lixo.

59. [Sierra, 2004] Dado o código a seguir,

```
12. X3 x2 = new X3();
13. X3 x3 = new X3();
14. X3 x5 = x3;
15. x3 = x2;
16. X3 x4 = x3;
17. x2 = null;
18. // insert code
```

Dois trechos de código, inseridos independentemente na linha 18, tornarão um objeto qualificado para a coleta de lixo. Quais (selecione dois)?

a) `x3 = null;`

b) `x4 = null;`

c) `x5 = null;`

d) `x3 = x4;`

e) `x5 = x4;`

60. [Sierra, 2004] Dado o código a seguir,

```
12. void doStuff3() {
13. X x = new X();
14. X y = doStuff(x);
15. y = null;
16. x = null;
17. }
18. X doStuff(X mx) {
19. return doStuff2(mx);
20. }
```

Em que ponto o objeto criado na linha 13 estará qualificado para a coleta de lixo?

a) depois que a linha 15 for executada.

b) depois que a linha 16 for executada.

c) depois que a linha 17 for executada.

d) o objeto não será qualificado.

e) não é possível saber com certeza.