01. [Sebesta, 2000] Quais são as três características gerais dos subprogramas?

02. [Sebesta, 2000] O que significa um subprograma estar ativo?

03. [Sebesta, 2000] O que é um perfil de parâmetro? O que é um protocolo de parâmetro?

04. [Sebesta, 2000] O que são parâmetros formais? O que são parâmetros reais?

05. [Sebesta, 2000] Quais são as vantagens e as desvantagens dos parâmetros nomeados?

06. [Sebesta, 2000] Quais são as questões de projeto referentes aos subprogramas?

07. [Sebesta, 2000] Quais são as vantagens e as desvantagens das variáveis locais dinâmicas?

08. [Sebesta, 2000] Quais são os três modelos semânticos de passagem de parâmetros?

09. [Sebesta, 2000] 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. [Sebesta, 2000] De que maneiras podem ocorrer *aliases* com parâmetros de passagem por referência?

11. [Sebesta, 2000] 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. [Sebesta, 2000] 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. [Sebesta, 2000] Quais são as duas considerações de projeto fundamentais referentes aos métodos de passagem de parâmetros?

14. [Sebesta, 2000] Quais são as duas questões que surgem quando nomes de subprograma são parâmetros?

15. [Sebesta, 2000] Defina vinculação rasa e vinculação profunda para ambientes de referenciamento de subprogramas passados como parâmetros.

16. [Sebesta, 2000] O que é um subprograma sobrecarregado?

17. [Sebesta, 2000] O que é polimorfismo paramétrico?

18. [Sebesta, 2000] O que faz uma função modelo C++ ser instanciada?

19. [Sebesta, 2000] Defina compilação separada e compilação independente.

20. [Sebesta, 2000] Quais são as questões de projeto referentes às funções?

21. [Sebesta, 2000] De que maneiras as co-rotinas são diferentes dos subprogramas convencionais?

22. [Sebesta, 2000] Quais são os argumentos favoráveis e quais são 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.

23. [Sebesta, 2000] 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.

24. [Sebesta, 2000] Argumente em defesa da decisão dos projetistas da Ada 83 de permitirem que o implementador escolha entre implementar parâmetros no modo **entrada/saída (*in/out*)** por cópia ou por referência.

25. [Sebesta, 2000] 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.

26. [Sebesta, 2000] Suponhamos que você deseje escrever um subprograma que imprima em um cabeçalho uma nova página de saída, juntamente com uma página que é 1 na primeira ativação e que aumenta 1 em cada ativação subsequente. 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?

27. [Sebesta, 2000] O FORTRAN permite que um subprograma tenha múltiplas entradas. Por que isso, às vezes, é uma capacidade valiosa?

28. [Sebesta, 2000] Escreva um procedimento Pascal `SOMADOR` que faz a adição de dois *arrays* de números inteiros. Ele deve ter somente dois parâmetros, os quais têm dois *arrays* a serem adicionados. O segundo *array* também conterá o *array* da soma na saída. Ambos os parâmetros devem ser passados por referência. Teste `SOMADOR` com a chamada `SOMADOR (A, A)` em que `A` é um *array* a ser adicionado a si mesmo. Explique os resultados de executar esse programa.

29. [Sebesta, 2000] Repita o Problema 28 usando C.

30. [Sebesta, 2000] Considere o procedimento `BIGSUB` apresentado a seguir.

```
procedure BIGSUB;
  integer GLOBAL;
  integer array LIST[1:2];
  procedure SUB(PARAM);
    integer PARAM;
    begin
      PARAM := 3;
      GLOBAL := GLOBAL + 1;
      PARAM := 5;
    end;
  begin
    LIST[1] := 2;
    LIST[2] := 2;
    GLOBAL := 1;
    SUB(LIST[GLOBAL]);
  end;
```

Mude as duas atribuições do *array* `LIST` para

```
LIST[1] := 3;
LIST[2] := 1;
```

Execute manualmente o novo programa sob as mesmas pressuposições e compare os valores resultantes do *array* `LIST` em `BIGSUB` depois do retorno de `SUB`:

a) Parâmetros são passados por valor

b) Parâmetros são passados por referência

c) Parâmetros são passados por nome

d) Parâmetros são passados por valor-resultado

31. [Sebesta, 2000] Considere o seguinte programa escrito na sintaxe C:

```
void main() {
  int valor = 2, lista[5] = {1, 3, 5, 7, 9};
  troca(valor, lista[0]);
  troca(lista[0], lista[1]);
  troca(valor, lista[valor]);
}
void troca(int a, int b) {
  int temp;
  temp = a;
  a = b;
  b = temp;
}
```

Para cada um dos métodos de passagem de parâmetros seguintes, quais são todos os valores das variáveis `valor` e `lista` depois de cada uma das três chamadas a `troca`?

a) Passados por valor

b) Passados por referência

c) Passados por nome

d) Passador por valor-resultado

32. [Sebesta, 2000] Apresente um argumento contrário a oferecer tanto variáveis estáticas como dinâmicas em subprogramas.

33. [Sebesta, 2000] Argumente contrariamente ao fato do *design* do C oferecer somente subprogramas de função.

34. [Sebesta, 2000] Em um livro didático sobre o FORTRAN, aprenda a sintaxe e a semântica das funções de instrução. Justifique a existência das mesmas no FORTRAN.

35. [Sebesta, 2000] Estude os métodos de sobrecarga de operador definida pelo usuário no C++ e na Ada e escreva um relatório comparando os dois, usando nossos critérios para avaliar linguagens.

36. [Sebesta, 2000] Considere o seguinte procedimento ALGOL 60, chamado Dispositivo de Jensen, em homenagem a J. Jensen, do *Regnecentralen* de Copenhague, que o projetou em 1960:

```
real procedure SOMA(SOMADOR, INDICE, COMPRIMENTO);
  value COMPRIMENTO
  real SOMADOR;
  integer SOMATEMP;
  begin
    real SOMATEMP;
    SOMATEMP := 0.0;
    for INDICE := 1 step 1 until COMPRIMENTO do
      SOMATEMP := SOMATEMP + SOMADOR;
   SOMA := SOMATEMP;
  end;
```

O que é retornado para cada uma das chamadas seguintes a `SOMA`, lembrando que os parâmetros são passados por nome e observando que o valor de retorno é definido ao atribuí-lo ao nome do subprograma?

a) `SOMA(A, I, 100)`, em que `A` é um escalar.

b) `SOMA(A[I]*A[I], I, 100)`, em que `A` é um *array* de 100 elementos.

c) `SOMA(A[I]*B[I], I, 100)`, em que `A` e `B` são *arrays* de 100 elementos.

37. Considere o seguinte programa escrito na sintaxe Pascal:

```
program SUB1;
var x, y: integer;
   procedure SUB2;
   begin
      writeln('x = ', x);
      writeln('y = ', y);
   end;
   procedure SUB3;
   var x, y: integer;
   begin
      x := 3;
      y := 6;
      SUB4(SUB2);
   end;
   procedure SUB4(procedure SUBX)
   var x: integer;
   begin
      x := 4;
      SUBX;
   end;
begin
   x := 1;
   y := 3;
   SUB3;
end.
```

Um aspecto interessante dos nomes de subprogramas passados como parâmetros é a questão referente ao ambiente de referenciamento correto para executar o subprograma passado. Quais serão os valores impressos no caso de ser utilizada a:

a) Vinculação rasa

b) Vinculação profunda

c) Vinculação *ad hoc*

38. [Sebesta, 2000] Considere o seguinte programa escrito na sintaxe C:

```
void main() {
  int a = 3;
  add(a, a + 2);
  printf("%d", a);
}
void add(int x, int y) {
  for (int i = 0; i < 3; i++) {
    x = x + y;
  }
}
```

Para cada um dos métodos de passagem de parâmetros a seguir, quais são os valores impressos pelo programa.

a) Passados por valor

b) Passados por referência

c) Passados por nome

d) Passador por valor-resultado

39. [Sebesta, 2000] Considere o seguinte programa escrito na sintaxe C:

```
void xpto(int a, int b, int c){
  a = b + c;
  b = a * c;
}

void main(){
  int x = 5;
  int y = 6;

  xpto(x, y, x + 2);

  printf("%d", x);
  printf("%d", y);
}
```

Para cada um dos métodos de passagem de parâmetros a seguir, quais são os valores impressos pelo programa.

a) Passados por valor

b) Passados por referência

c) Passados por nome

d) Passador por valor-resultado