01. [Sebesta, 2000] Quais são as questões de projeto referentes a nomes?

02. [Sebesta, 2000] Qual é o perigo potencial dos nomes que fazem distinção entre maiúsculas e minúsculas?

03. [Sebesta, 2000] De que maneira as palavras reservadas são melhores do que as palavras-chave?

04. [Sebesta, 2000] O que é um *alias*?

05. [Sebesta, 2000] Quais categorias de variáveis de referência do C++ são sempre *aliases*?

06. [Sebesta, 2000] Qual é o *valor-l* de uma variável? Qual é o *valor-r*?

07. [Sebesta, 2000] Defina vinculação e tempo de vinculação?

08. [Sebesta, 2000] Depois do projeto e da implementação da linguagem, quais são os quatro momentos em que vinculações podem desenvolver-se em um programa?

09. [Sebesta, 2000] Defina vinculação estática e vinculação dinâmica.

10. [Sebesta, 2000] Quais são as vantagens e as desvantagens das declarações implícitas?

11. [Sebesta, 2000] Quais são as vantagens e as desvantagens da vinculação dinâmica de tipos?

12. [Sebesta, 2000] Defina variáveis estáticas, *stack*-dinâmicas, *heap*-dinâmicas explícitas e *heap*-dinâmicas implícitas. Quais são as vantagens e as desvantagens de cada uma?

13. [Sebesta, 2000] Defina coerção, erro de tipo, verificação de tipos e tipificação forte.

14. [Sebesta, 2000] Defina compatibilidade de tipo de nome e compatibilidade de tipo de estrutura. Quais são os méritos relativos de ambas?

15. [Sebesta, 2000] Qual é a diferença entre um tipo derivado Ada e um subtipo Ada?

16. [Sebesta, 2000] Defina tempo de vida, escopo, escopo estático e escopo dinâmica.

17. [Sebesta, 2000] Como uma referência a uma variável não-local em um programa de escopo estático é vinculada à sua definição?

18. [Sebesta, 2000] Qual é o problema geral com o escopo estático?

19. [Sebesta, 2000] Qual é o ambiente de referenciamento de uma instrução?

20. [Sebesta, 2000] O que é um ancestral estático de um subprograma? O que é um ancestral dinâmico de um subprograma?

21. [Sebesta, 2000] O que é um bloco?

22. [Sebesta, 2000] Quais são as vantagens e as desvantagens do escopo dinâmico?

23. [Sebesta, 2000] Quais são as vantagens das constantes nomeadas?

24. [Sebesta, 2000] Decida qual das seguintes formas de identificador é a mais legível e sustente essa decisão:

a) `SomaDeVendas`

b) `soma_de_vendas`

c) `SOMADEVENDAS`

25. [Sebesta, 2000] Algumas linguagens de programação são sem tipo. Quais são as vantagens e as desvantagens evidentes de não se ter tipos em uma linguagem?

26. [Sebesta, 2000] Uma utilização comum de `EQUIVALENCE` do FORTRAN é a seguinte: um array grande de valores numéricos é colocado à disposição de um subprograma como um parâmetro. O array contém muitas variáveis diferentes não-relacionadas, em vez de uma coleção de repetições da mesma variável. Ele é representado como um array para reduzir o número de nomes que precisam ser passados como parâmetros. Dentro de um subprograma, uma instrução `EQUIVALENCE` extensa é usada para criar nomes conotativos como *aliases* para os vários elementos do array, o que aumenta a legibilidade do código do subprograma. Essa é uma boa ideia ou não? Quais alternativas para o uso de *aliases* estão disponíveis?

27. [Sebesta, 2000] Escreva uma instrução de atribuição simples com um operador aritmético em alguma linguagem de programação que você conheça. Para cada componente da instrução, liste as várias vinculações que são necessárias para determinar a semântica quando a instrução é executada. Para cada vinculação, indique o tempo de vinculação usado para a linguagem.

28. [Sebesta, 2000] A vinculação dinâmica de tipos está estreitamente relacionada com as variáveis heap-dinâmicas. Explique essa relação.

29. [Sebesta, 2000] Descreva a situação em que uma variável sensível à história em um subprograma é útil.

30. [Sebesta, 2000] Pesquise a definição de fortemente tipificada apresentada em Gehani (1983) e compare-a com a definição apresentada por Sebesta (2000). De que maneira elas diferem?

31. [Sebesta, 2000] Considere o seguinte programa Pascal esquemático.

```
program main;
  var x : integer;
  procedure sub3; forward;
  procedure sub1;
    var x : integer;
    procedure sub2;
      begin { sub2 }
      ...
      end; { sub2 }
    begin { sub1 }
    ...
    end; { sub1 }
  procedure sub3;
    begin { sub3 }
    ...
    end; { sub3 }
  begin { main }
  ...
  end. { main }
```

Suponhamos que a execução desse programa seja na seguinte ordem unitária:

`main` chama `sub1`
`sub1` chama `sub2`
`sub2` chama `sub3`

a) Supondo que esteja presente o escopo estático, qual declaração de x é a correta para a referência a x em:

i. sub1
ii. sub2
iii. sub3

b) Repita a parte a, mas suponha a presença do escopo dinâmico.

32. [Sebesta, 2000] Suponha que o seguinte programa tenha compilado e executado usando regras de escopo estático. Qual valor de `x` é impresso no procedimento `sub1`? De acordo com as regras de escopo dinâmico, qual valor de `x` é impresso no procedimento `sub1`?

```
program main;
  var x : integer;
  procedure sub1;
    begin { sub1 }
      writeln('x = ', x);
    end; { sub1 }
  procedure sub2;
    var x : integer;
    begin { sub2 }
      x := 10;
      sub1;
    end; { sub2 }
  begin { main }
    x := 5;
    sub2;
  end. { main }
```

33. [Sebesta, 2000] Considere o seguinte programa:

```
program main;
  var x, y, z : integer;
  procedure sub1;
    var a, y, z : integer;
    procedure sub2;
      var a, b, z : integer;
      begin { sub2 }
      ...
      end; { sub2 }
    begin { sub1 }
    ...
    end; { sub1 }
  procedure sub3;
    var a, x, w : integer;
    begin { sub3 }
    ...
    end; { sub3 }
  begin { main }
  ...
  end. { main }
```

Liste todas as variáveis, juntamente com as unidades de programa onde elas foram declaradas que são visíveis nos corpos de `sub1`, `sub2` e `sub3`, supondo que seja usado o escopo estático.

34. [Sebesta, 2000] Considere o seguinte programa:

```
program main;
  var x, y, z : integer;
  procedure sub1;
    var a, y, z : integer;
    begin { sub1 }
    ...
    end; { sub1 }
  procedure sub2;
    var a, x, w : integer;
    procedure sub3;
      var a, b, z : integer;
      begin { sub3 }
```

```
      ...
      end; { sub3 }
    begin { sub2 }
    ...
    end; { sub2 }
  begin { main }
  ...
  end. { main }
```

Liste todas as variáveis, juntamente com as unidades de programa onde elas estão declaradas, que são visíveis nos corpos de `sub1`, `sub2` e `sub3`, supondo que seja usado o escopo estático.

35. [Sebesta, 2000] Considere o seguinte programa em C:

```
void fun(void) {
  int a, b, c; /* definição 1 */
  ...
  while(...) {
    int b, c, d; /* definição 2 */
    ... (1)
    while(...) {
      int c, d, e; /* definição 3 */
      ... (2)
    }
    ... (3)
  }
  ... (4)
}
```

Para cada um dos quatro pontos marcados nessa função, liste cada variável visível, juntamente com o número da instrução de definição que a define.

36. [Sebesta, 2000] Considere o seguinte programa C esquemático:

```
void fun1(void); /* protótipo */
void fun2(void); /* protótipo */
void fun3(void); /* protótipo */
void main() {
  int a, b, c;
  ...
}
void fun1(void) {
  int b, c, d;
  ...
}
void fun2(void) {
  int c, d, e;
  ...
}
void fun3(void) {
  int d, e, f;
  ...
}
```

Dada a sequência de chamada a seguir e supondo-se que seja usado o escopo dinâmico, quais variáveis são visíveis durante a execução da última função chamada? Inclua, em cada variável visível, o nome da função em que ela foi definida.

a) `main` chama `fun1`; `fun1` chama `fun2`; `fun2` chama `fun3`.

b) `main` chama `fun1`; `fun1` chama `fun3`.

c) `main` chama `fun2`; `fun2` chama `fun3`; `fun3` chama `fun1`.

d) `main` chama `fun3`; `fun3` chama `fun1`.

e) `main` chama `fun1`; `fun1` chama `fun3`; `fun3` chama `fun2`.

f) `main` chama `fun3`; `fun3` chama `fun2`; `fun2` chama `fun1`.

37. [Sebesta, 2000] Considere o seguinte programa:

```
program main;
  var x, y, z : integer;
  procedure sub1;
    var a, y, z : integer;
    begin { sub1 }
    ...
    end; { sub1 }
  procedure sub2;
    var a, b, z : integer;
    begin { sub2 }
    ...
    end; { sub2 }
  procedure sub3;
    var a, x, w : integer;
    begin { sub3 }
    ...
    end; { sub3 }
  begin { main }
  ...
  end. { main }
```

Dada as seguintes sequências de chamada e supondo-se que seja usado o escopo dinâmico, quais variáveis são visíveis durante a execução do último subprograma ativado? Inclua, em cada variável visível, o nome da unidade em que ela foi declarada.

a) `main` chama `sub1`; `sub1` chama `sub2`; `sub2` chama `sub3`.

b) `main` chama `sub1`; `sub1` chama `sub3`.

c) `main` chama `sub2`; `sub2` chama `sub3`; `sub3` chama `sub1`.

d) `main` chama `sub3`; `sub3` chama `sub1`.

e) `main` chama `sub1`; `sub1` chama `sub3`; `sub3` chama `sub2`.

f) `main` chama `sub3`; `sub3` chama `sub2`; `sub2` chama `sub1`.

38. [Sebesta, 2000] Considere o seguinte programa C esquemático:

```
void fun1(void); /* protótipo */
void fun2(void); /* protótipo */
void fun3(void); /* protótipo */
void main() {
  int x, y, z;
  ...
}
void fun1(void) {
  int y, z, a;
```

```
  ...
}
void fun2(void) {
  int z, b, c;
  ...
}
void fun3(void) {
  int a, b, c;
  ...
}
```

Dada a sequência de chamada a seguir e supondo-se que seja usado o escopo dinâmico, quais variáveis são visíveis durante a execução da última função chamada? Inclua, em cada variável visível, o nome da função em que ela foi definida.

a) `main` chama `fun1`; `fun1` chama `fun2`; `fun2` chama `fun3`.

b) `main` chama `fun2`; `fun2` chama `fun3`; `fun3` chama `fun1`.

c) `main` chama `fun1`; `fun1` chama `fun3`; `fun3` chama `fun2`.

d) `main` chama `fun3`; `fun3` chama `fun2`; `fun2` chama `fun1`.