UNIVERSIDADE FEDERAL DE SÃO CARLOS - DEPARTAMENTO DE COMPUTAÇÃO

PARADIGMAS DE LINGUAGENS DE PROGRAMAÇÃO – 3ª LISTA DE EXERCÍCIOS

Programação Estruturada - 2009

1. Considere o seguinte programa em C:

```
main ( )
{     int a, b ;
      a = 10;
      b = 2;
      soma ( a, b) ;
}
soma (int x, int y)
{     int z;
      z = x + y ;
      printf("%d \n", z) ;
}
```

Determine o momento em que ocorrem as alocações e liberações de memória, as amarrações de tipo e de valor das variáveis a, b e z, de acordo com as fases abaixo.

- Fase 1 - Compilação
- Fase 2 - Do início da execução do programa até a chamada do procedimento soma;
- Fase 3 – Durante a elaboração do procedimento soma;
- Fase 4 - Durante a execução do procedimento soma;
- Fase 5 - Do final da execução do procedimento soma até o final da execução do programa.

2. Considere o seguinte programa, escrito em uma linguagem tipo PASCAL:

```
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  }
```

Para a sequência de chamadas: main chama sub1; sub1 chama sub2; sub2 chama sub3, diga quais variáveis são visíveis durante a execução de sub3, considerando:

a) amarração de escopo dinâmica
b) amarração de escopo estática

3. Considere o seguinte programa em Pascal. Supondo regras de escopo estático, qual valor de x é impresso no procedimento sub1? E para regras de escopo dinâmico?

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 }
```

4. Considere o seguinte programa C esquemático:

```
void fun1 (void);
void fun2 (void);
void fun3 (void);
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;
    ...
}
```

Dadas as seguintes seqüências de chamadas e supondo-se que seja usado o escopo dinâmico, quais variáveis são visíveis durante a execução da última função chamada? Diga, para cada variável visível na última função, o nome da função em que ela foi declarada.

- main chama fun1; fun1 chama fun2; fun2 chama fun3.
- main chama fun2; fun2 chama fun3; fun3 chama fun1.

5. Defina amarração e tempo de amarração.

6. Diga o que são alocação de memória estática e dinâmica. Explique como ocorrem as alocações de memória em C, C++ e JAVA.

7. Explique as diferenças de passagem de parâmetros entre C++ e JAVA.

8. Diga o que são amarração de tipo estática e dinâmica. Explique como ocorrem as amarrações de tipo em C e JAVA.

9. Considere o seguinte programa em C:

```
main ( )
{     int a, b ;
      a = 10;
      b = 2;
      soma1 ( a, b) ;                    2
      soma2 (&a, &b);
}
soma1 (int x, int y)
{     int z;
      z = x + y ;
      printf("%d \n", z) ;               1
}
soma2 (int *x, int *y)
{     int z;                             3
      z = *x + *y ;
      printf("%d \n", z) ;
}
```

Mostre o conteúdo da pilha de execução nos pontos 1, 2 e 3 indicados.

10. Considere o seguinte programa escrito em uma linguagem semelhante ao 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, descreva os passos para realizar essa passagem e diga quais são os valores das variáveis valor e lista antes e depois de cada uma das três chamadas a troca. Mostre o conteúdo da pilha de execução para cada um desses casos.

- passados por valor
- passados por referência
- passados por nome
- passados por valor-resultado

11. Mostre a pilha com todas as instâncias do registro de ativação, incluindo encadeamentos estáticos e dinâmicos, quando a execução atingir a posição 1 no programa esquemático a seguir. Suponha que BIGSUB está no nível 1.

```
procedure BIGSUB;
  procedure A;
    procedure B;
      begin { B }
      .....
      end; { B }                         1
    procedure C;
      begin { C }
```

```
      .....
      B;
      ....
      end; { C }
    begin { A }
      .....
      C;
      ....
    end; { A }
  begin { BIGSUB }
  ...
  A;
  ...
  end; { BIGSUB }
```

12. Mostre a pilha com todas as instâncias do registro de ativação, incluindo encadeamentos estáticos e dinâmicos, quando a execução atingir a posição 1 no programa esquemático seguinte. Suponha que BIGSUB esteja no nível 1.

```
procedure BIGSUB;
  procedure C; forward;
  procedure A (flag: boolean);
    procedure B;
      begin { B }
      ....
      A(false)
      end { B };
    begin { A }
    if flag
      then B
      else C
    ...
    end ; { A }
  procedure C;
    procedure D;
      .....                    <-------  1
      end { D }
    begin { C }
    ...
    D
    end; { C }
  A(true);
  ....
  end; { BIGSUB }
```

A sequência de chamada desse programa para que a execução atinja D é:
BIGSUB chama A
A chama B
B chama A
A chama C
C chama D

13. Para cada um dos quatro pontos indicados, (1, 2, 3 e 4) diga quais variáveis de quais procedimentos estão sendo referenciadas. Mostre a situação da pilha de execução nos pontos 1, 2, 3 e 4.

```
program MAIN;
  var A, B, C : integer;
  procedure SUB1 (X : integer) ;
    var A, D : integer;
    procedure SUB4;
      begin {SUB4 }
        .....
        A : = D / 2;                    <-- 1
        ......
      end; { SUB4 }
    begin { SUB1}
    .....
    D := X + 1;
    ......                              <-- 2
    SUB4;
    .....
    B : = A + X;
    .....
    end; { SUB1}
  procedure SUB2;
    var B, E  : integer;
    procedure SUB3;
      var C, E : integer;
      begin { SUB3 }
      B := 0;                           <-- 3
      SUB1( 5);
      .......
      E := B + A;
      end ; {SUB3}
    begin { SUB2 }
      .....
      SUB3;                             <-- 4
      ......
      A := B + 1;
      ....
    end; { SUB2 }
  begin { MAIN }
      A := 100;
      .....
      SUB2;
      ....
  end; { MAIN }
```