Subprogramas

Guilherme, Gustavo, Sean e Vinícius

Fundamentos

Ambiente de re-ferênciamento locais

Aninhamento de subprogramas

Métodos de Passagem de Parâmetros

Subprogramas Como Parâmetro

Chamar Subprogramas Indiretamente

Sobrecarga de Subprogramas

Subprogramas

Guilherme, Gustavo, Sean e Vinícius

Universidade Estadual de Londrina

October 9, 2013

Sumário

1 Fundamentos

2 Ambiente de referênciamento locais

3 Aninhamento de subprogramas

4 Métodos de Passagem de Parâmetros

5 Subprogramas Como Parâmetro

6 Chamar Subprogramas Indiretamente

7 Sobrecarga de Subprogramas

8 Suprogramas Genéricos

9 Questões de projetos referente a funções

10 Sobrecarga de operadores definidos pelo usuário

11 Closure

12 Co-rotinas

Introdução

Subprogramas

Guilherme, Gustavo, Sean e Vinícius

Fundamentos

Ambiente de re-ferênciamento locais

Aninhamento de subprogramas

Métodos de Passagem de Parâmetros

Subprogramas Como Parâmetro

Chamar Subprogramas Indiretamente

Sobrecarga de Subprogramas

- Conceito muito importante nas linguagens de programação.
- Máquina análitica de Babbage
- Reuso, economia de tempo e abstração.
- Métodos nas linguagens orientadas a objeto também são subprogramas

Características comuns

Subprogramas

Guilherme, Gustavo, Sean e Vinícius

Fundamentos

Ambiente de referênciamento locais

Aninhamento de subprogramas

Métodos de Passagem de Parâmetros

Subprogramas Como Parâmetro

Chamar Subprogramas Indiretamente

Sobrecarga de Subprogramas

- Cada subprograma tem um único ponto de entrada.
- Há apenas um subprograma em execução em um dado momento.
- O controle sempre retorna para a estrutura que chamou quando a execução do subprograma termina.

Cabeçalho de subprograma

Fornece um nome para o subprograma e especifica uma lista de parâmetros.

- Ruby e Python

  ```
  def funcao (parametros)
  ```

- C-based

  ```
  void funcao (parametros)
  ```

Corpo dos subprogramas

Os corpos dos subprogramas definem as computações.

- *C-based.* Delimitadas por chaves { }
- *Python.* Identação
- *Ruby.* Palavra-chave **end**

Peculiaridade de Ruby

Subprogramas

Guilherme, Gustavo, Sean e Vinícius

Fundamentos

Ambiente de referênciamento locais

Aninhamento de subprogramas

Métodos de Passagem de Parâmetros

Subprogramas Como Parâmetro

Chamar Subprogramas Indiretamente

Sobrecarga de Subprogramas

```
1  class Exemplo
2    def invocar_subprograma
3      puts self.method(:invocar_subprograma).owner #
           Exemplo
4      subprograma
5    end
6  end
7
8  def subprograma
9    puts self.method(:subprograma).owner # Object
10 end
11
12 exemplo = Exemplo.new
13 exemplo.invocar_subprograma
```

Parâmetros

Subprogramas

Guilherme, Gustavo, Sean e Vinícius

Fundamentos

Ambiente de re-ferênciamento locais

Aninhamento de subprogramas

Métodos de Passagem de Parâmetros

Subprogramas Como Parâmetro

Chamar Subprogramas Indiretamente

Sobrecarga de Subprogramas

- Parâmetros Reais (Argumentos)
- Parâmetros Formais (Parâmetros)

Exemplos de parâmetros por palavras-chave

Subprogramas

Guilherme, Gustavo, Sean e Vinícius

Fundamentos

Ambiente de referênciamento locais

Aninhamento de subprogramas

Métodos de Passagem de Parâmetros

Subprogramas Como Parâmetro

Chamar Subprogramas Indiretamente

Sobrecarga de Subprogramas

- Parâmetros Formais (Parâmetros)

```
funcao(20, 10)
```

- Parâmetros Reais (Argumentos)

```
void funcao(int param1, int param2)
```

Exemplos de parâmetros por palavras-chave

Subprogramas

Guilherme, Gustavo, Sean e Vinícius

Fundamentos

Ambiente de referênciamento locais

Aninhamento de subprogramas

Métodos de Passagem de Parâmetros

Subprogramas Como Parâmetro

Chamar Subprogramas Indiretamente

Sobrecarga de Subprogramas

Ada, Fortran 95+ e Python

```
1 def subprograma(param1, param2, param3):
2     print param1 # 30
3     print param2 # 20
4     print param3 # 10
5
6 subprograma(param3 = 10, param2 = 20, param1 = 30)
```

Desvantagem: Cliente precisa saber o nome dos parâmetros

Exemplos de parâmetros com valores padrão

Subprogramas

Guilherme, Gustavo, Sean e Vinícius

Fundamentos

Ambiente de referênciamento locais

Aninhamento de subprogramas

Métodos de Passagem de Parâmetros

Subprogramas Como Parâmetro

Chamar Subprogramas Indiretamente

Sobrecarga de Subprogramas

Python, Ruby, C++, Fortran 95+, Ada e PHP

```
1 def subprograma(param1, param2 = 20, param3 = 30):
2     print param1 # 10
3     print param2 # 20
4     print param3 # 30
5
6 subprograma(10)
```

Passagem de hash e listas como parâmetros

Subprogramas

Guilherme, Gustavo, Sean e Vinícius

Fundamentos

Ambiente de referênciamento locais

Aninhamento de subprogramas

Métodos de Passagem de Parâmetros

Subprogramas Como Parâmetro

Chamar Subprogramas Indiretamente

Sobrecarga de Subprogramas

```
1  class Conta
2    def transfere(argumentos, *valores)
3      destino = argumentos[:para]
4      data = argumentos[:em]
5
6    end
7  end
8
9  conta = Conta.new
10 conta.transfere({:para => :escola, :em => Time.now},
       [20.0, 30.0])
```

Ambiente de referênciamento locais

Subprogramas

Guilherme, Gustavo, Sean e Vinícius

Fundamentos

Ambiente de re-ferênciamento locais

Aninhamento de subprogramas

Métodos de Passagem de Parâmetros

Subprogramas Como Parâmetro

Chamar Subprogramas Indiretamente

Sobrecarga de Subprogramas

Variáveis locais estáticas

São vinculadas ao armazenamento antes da execução do programa e continuam até seu término.

- + Endereçamento direto na memória.
- + Não causam sobrecarga na alocação e desalocação.
- - Não se comportam bem em programas recursivos.
- - Representam um estado global.

Ambiente de referênciamento locais

Subprogramas

Guilherme, Gustavo, Sean e Vinícius

Fundamentos

Ambiente de re-ferênciamento locais

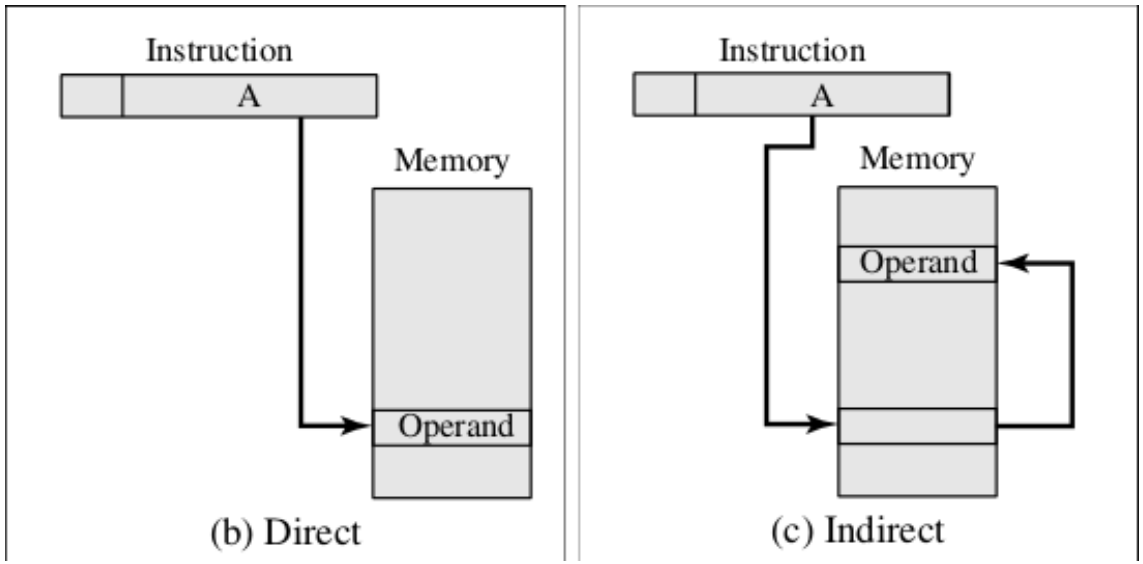
Aninhamento de subprogramas

Métodos de Passagem de Parâmetros

Subprogramas Como Parâmetro

Chamar Subprogramas Indiretamente

Sobrecarga de Subprogramas



Ambiente de referênciamento locais

Subprogramas

Guilherme, Gustavo, Sean e Vinícius

Fundamentos

Ambiente de re-ferênciamento locais

Aninhamento de subprogramas

Métodos de Passagem de Parâmetros

Subprogramas Como Parâmetro

Chamar Subprogramas Indiretamente

Sobrecarga de Subprogramas

```
1  int sum (int arr[], int n)
2  {
3      static int result = 0;
4      if (n == 0)
5          return result ;
6      else {
7          result += arr[n - 1];
8          sum(arr, n - 1);
9       }
10 }
11
12 int main(void) {
13     int array[5] = {1,2,3,4,5};
14     printf("%d\n", sum(array, 3));   // 6
15     printf("%d\n", sum(array, 3));   // 12
16     printf("%d\n", sum(array, 3));   // 18
17     return 0;
18 }
```

Ambiente de referênciamento locais

Subprogramas

Guilherme, Gustavo, Sean e Vinícius

Fundamentos

Ambiente de re-ferênciamento locais

Aninhamento de subprogramas

Métodos de Passagem de Parâmetros

Subprogramas Como Parâmetro

Chamar Subprogramas Indiretamente

Sobrecarga de Subprogramas

Variáveis locais dinâmicas na pilha

Variáveis dinâmicas na pilha, são vinculadas ao armazenamento quando o subprograma inicia sua execução e desvinculadas do armazenamento quando ele se encerra.

- + Maior flexibilidade (programas recursivos).
- - Sobrecarga na alocação e desalocação.
- - Endereçamento indireto.

Ambiente de referênciamento locais

Subprogramas

Guilherme, Gustavo, Sean e Vinícius

Fundamentos

Ambiente de re-ferênciamento locais

Aninhamento de subprogramas

Métodos de Passagem de Parâmetros

Subprogramas Como Parâmetro

Chamar Subprogramas Indiretamente

Sobrecarga de Subprogramas

Exemplos

- ALGOL 60 e suas linguagens descendentes, possuem variáveis locais dinâmicas na pilha.
- Funções em C possuem variáveis são dinâmicas na pilha a menos que sejam especificamente declaradas como static.
- Subprogramas Pascal e Ada e métodos em C++, Java, C# têm somente variaveis dinâmicas na pilha.

Aninhamento de subprogramas

Subprogramas

Guilherme, Gustavo, Sean e Vinícius

Fundamentos

Ambiente de re-ferênciamento locais

Aninhamento de subprogramas

Métodos de Passagem de Parâmetros

Subprogramas Como Parâmetro

Chamar Subprogramas Indiretamente

Sobrecarga de Subprogramas

Linguagens como ALGOL 68, Pascal e Ada, JavaScript, Python e Lua permitem aninhamento de subprogramas. Linguagens descententes de C não permitem aninhamento.

```
function hipotenusa(a, b) {
    function quadrado(x) {
        return x * x;
    }
    return Math.sqrt(quadrado(a) + quadrado(b));
}
```

Métodos de Passagem de Parâmetros

Subprogramas

Guilherme, Gustavo, Sean e Vinícius

Fundamentos

Ambiente de re-ferênciamento locais

Aninhamento de subprogramas

Métodos de Passagem de Parâmetros

Subprogramas Como Parâmetro

Chamar Subprogramas Indiretamente

Sobrecarga de Subprogramas

- Acesso a dados
- Apresentar e recuperar valores
- Parâmetros formais e parâmetros reais

Exemplo de parâmetros reais e parâmetros formais em C++

```
1 void soma(int a, int b) {
2     cout << "Soma = " << a+b;
3 }
4 int main() {
5     int x = 2;
6     int y = 5;
7     soma(x, y);
8     return 0;
9 }
```

Modelos semânticos de passagem de parâmetros

modo entrada (in mode)

Os parâmetros formais recebem dados do parâmetro real.

modo saída (out mode)

Os parâmetros formais transmititem dados para o parâmetro real.

modo entrada/saída (inout mode)

Podem fazer ambos.

Passagem por valor

Subprogramas

Guilherme, Gustavo, Sean e Vinícius

Fundamentos

Ambiente de re-ferênciamento locais

Aninhamento de subprogramas

Métodos de Passagem de Parâmetros

Subprogramas Como Parâmetro

Chamar Subprogramas Indiretamente

Sobrecarga de Subprogramas

- Implementação para parâmetros em modo entrada
- O valor do parâmetro real é utilizado para inicializar o parâmetro formal que atua como uma variável local no subprograma.
- A transferência dos dados pode ser feita por cópia dos valores ou pela transmissão de um caminho de acesso (ponteiro ou referência) para o valor do parâmetro real.

Passagem por valor

- O método de passagem por valor é rápida na vinculação e no tempo de acesso.
- Caminho de acesso: Proteção de célula contra escrita.
- Cópia: espaço adicional para armazenamento e as operações de transferência podem ser custosas se o parâmetro for grande.

Passagem por resultado

- Implementação para parâmetros em modo saída
- Nenhum valor é transmitido na chamada do subprograma.
- Antes que o controle retorne para o chamador o valor do parâmetro formal é copiado para o parâmetro real.
- Dificuldade esta de garantir que o valor inicial do parâmetro real não seja utilizado no subprograma chamado.

Passagem por resultado

Subprogramas

Guilherme, Gustavo, Sean e Vinícius

Fundamentos

Ambiente de re-ferênciamento locais

Aninhamento de subprogramas

Métodos de Passagem de Parâmetros

Subprogramas Como Parâmetro

Chamar Subprogramas Indiretamente

Sobrecarga de Subprogramas

- Colisão de parâmetros reais: Qual será o valor retornado para a?

Exemplo em C#

```
1 void atribuicao(out int x, out int y) {
2     x = 29;
3     y = 15;
4 }
5 ...
6 f.atribuicao(out a, out a);
```

Passagem por resultado

Subprogramas

Guilherme, Gustavo, Sean e Vinícius

Fundamentos

Ambiente de re-ferênciamento locais

Aninhamento de subprogramas

Métodos de Passagem de Parâmetros

Subprogramas Como Parâmetro

Chamar Subprogramas Indiretamente

Sobrecarga de Subprogramas

- Tempo para avaliar os endereços dos parâmetros reais.
- Chamada list[21] ou no retorno list[5]?

Exemplo em C#

```
1 void subprograma(out int x, int index) {
2     x = 23;
3     index = 5;
4 }
5 ...
6 sub = 21;
7 f.subprograma(list[sub], sub);
```

Passagem por valor-resultado

Subprogramas

Guilherme, Gustavo, Sean e Vinícius

Fundamentos

Ambiente de re-ferênciamento locais

Aninhamento de subprogramas

Métodos de Passagem de Parâmetros

Subprogramas Como Parâmetro

Chamar Subprogramas Indiretamente

Sobrecarga de Subprogramas

- Implementação para parâmetros em modo entrada-saída.
- Combinação da passagem por valor com a passagem por resultado.
- O valor do parâmetro real é usado para inicializar o parâmetro formal que atua como variável local.
- No termino do subprograma o valor do parâmetro formal é transmitido de volta para o parâmetro real.
- Partilha dos mesmos problemas da passagem por valor e passagem por resultado.

Passagem por referência

- Implementação para parâmetros em modo entrada-saída.
- Transmissão de um caminho de acesso.
- Vantagem em relação a passagem por valor-resultado: não é necessário espaço duplicado e nem operações de cópia.
- Desvantagem: o acesso ao parâmetro formal é mais lento do que a passagem por valor devido ao endereçamento indireto.

Passagem por nome

Subprogramas

Guilherme, Gustavo, Sean e Vinícius

Fundamentos

Ambiente de re-ferênciamento locais

Aninhamento de subprogramas

Métodos de Passagem de Parâmetros

Subprogramas Como Parâmetro

Chamar Subprogramas Indiretamente

Sobrecarga de Subprogramas

- Implementação para parâmetros em modo entrada-saída.
- O parâmetro real é textualmente substituído pelo parâmetro formal em todas as suas ocorrências no subprograma.
- O parâmetro formal é vinculado a valores ou a endereços reais.
- A vinculação real é retardada até o momento que o parâmetro formal seja atribuído ou referenciado.
- Usado em tempo de compilação para parâmetros genéricos de subprogramas genéricos em C++, Java 5.0 e C# 2005.

Pilha em tempo de execução

Subprogramas

Guilherme, Gustavo, Sean e Vinícius

Fundamentos

Ambiente de referênciamento locais

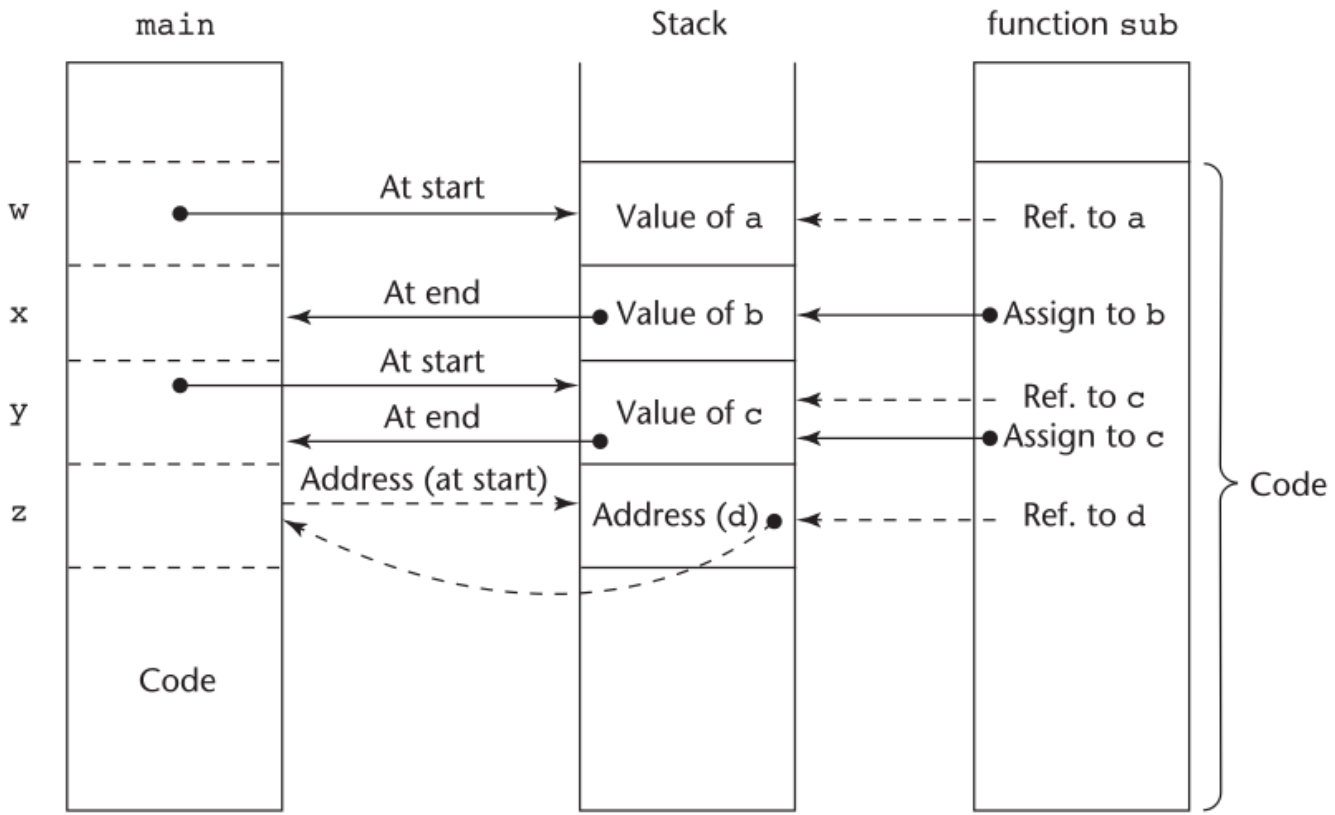
Aninhamento de subprogramas

Métodos de Passagem de Parâmetros

Subprogramas Como Parâmetro

Chamar Subprogramas Indiretamente

Sobrecarga de Subprogramas



Function header: **void** sub (**int** a, **int** b, **int** c, **int** d)
Function call in main: sub (w,x,y,z)
(pass w by value, x by result, y by value-result, z by reference)

Métodos de passagem de parâmetros das principais linguagens

Subprogramas

Guilherme, Gustavo, Sean e Vinícius

Fundamentos

Ambiente de re-ferênciamento locais

Aninhamento de subprogramas

Métodos de Passagem de Parâmetros

Subprogramas Como Parâmetro

Chamar Subprogramas Indiretamente

Sobrecarga de Subprogramas

- O C usa passagem por valor, porém obtêm a semântica da passagem por referência utilizando ponteiros (copiado do ALGOL 68).
- O C++ utiliza também a passagem por valor e garante a passagem por referência com ponteiros.
- O C++ ainda possui um tipo especial de ponteiro chamado tipo de referência que após sua inicialização não pode referenciar outra variável.

```
1 void fun(const int &p1, int p2, int &p3) { . . . }
```

Métodos de passagem de parâmetros das principais linguagens

Subprogramas

Guilherme, Gustavo, Sean e Vinícius

Fundamentos

Ambiente de re-ferênciamento locais

Aninhamento de subprogramas

Métodos de Passagem de Parâmetros

Subprogramas Como Parâmetro

Chamar Subprogramas Indiretamente

Sobrecarga de Subprogramas

- Em Java os parâmetros também são passados por valor, porém como os objetos são apenas acessados por variáveis de referência os parâmetros são passados com a semântica de referência.
- Ada e Fortran 95+ permitem ao programador especificar o modo de cada parâmetro formal (entrada, saída e entrada-saída).

Métodos de passagem de parâmetros das principais linguagens

Subprogramas

Guilherme, Gustavo, Sean e Vinícius

Fundamentos

Ambiente de re-ferênciamento locais

Aninhamento de subprogramas

Métodos de Passagem de Parâmetros

Subprogramas Como Parâmetro

Chamar Subprogramas Indiretamente

Sobrecarga de Subprogramas

- O C# utiliza a passagem por valor como padrão, mas também permite ao programador utilizar passagem por refreferência se o prefixo **ref** for utilizado antes dos dois parâmetros (real e formal).
- Também suporta passagem de parâmetro em modo saída, passado por referência, com o modificador out antes do parâmetro formal.

```
1 void sumer(ref int oldSum, int newOne) { . . . }
2 ...
3 sumer(ref sum, newValue);
```

Métodos de passagem de parâmetros das principais linguagens

Subprogramas

Guilherme, Gustavo, Sean e Vinícius

Fundamentos

Ambiente de re-ferênciamento locais

Aninhamento de subprogramas

Métodos de Passagem de Parâmetros

Subprogramas Como Parâmetro

Chamar Subprogramas Indiretamente

Sobrecarga de Subprogramas

- Em Python e Ruby é utilizado a passagem por atribuição em que todos os valores de dados são objetos.
- Se uma variável referenciada é acrescida de uma unidade então é criado um novo objeto com o valor da variável mais 1 e a variável referencia o novo objeto.
- No caso de vetor passado como parâmetro se houver uma atribuição ao parâmetro formal que referencia o vetor então não tem efeito no chamador.
- Se houve uma atribuição à um elemento do vetor passado então o correspondente parâmetro real será modificado.

Matrizes multidimencionais como parâmetro

Subprogramas

Guilherme, Gustavo, Sean e Vinícius

Fundamentos

Ambiente de referênciamento locais

Aninhamento de subprogramas

Métodos de Passagem de Parâmetros

Subprogramas Como Parâmetro

Chamar Subprogramas Indiretamente

Sobrecarga de Subprogramas

- Quando uma matriz é passado como parâmetro o compilador deve ser capaz de construir uma função de mapeamento.
- Uma função de mapeamento simples mapeia valores inteiros (índices de elementos na matriz) para os endereços dos elementos da matriz.

Matrizes multidimencionais como parâmetro

Subprogramas

Guilherme, Gustavo, Sean e Vinícius

Fundamentos

Ambiente de re-ferênciamento locais

Aninhamento de subprogramas

Métodos de Passagem de Parâmetros

Subprogramas Como Parâmetro

Chamar Subprogramas Indiretamente

Sobrecarga de Subprogramas

Exemplo em C

```
1 void procedure(int *mat, int rows, int cols) {
2       ...
3 }
4 void main() {
5       int mat[2][3];
6       ...
7       procedure(&mat, 2, 3); //ou procedure(mat[0][0],
              2, 3);
8       ...
9 }
```

Matrizes multidimencionais como parâmetro

Subprogramas

Guilherme, Gustavo, Sean e Vinícius

Fundamentos

Ambiente de re-ferênciamento locais

Aninhamento de subprogramas

Métodos de Passagem de Parâmetros

Subprogramas Como Parâmetro

Chamar Subprogramas Indiretamente

Sobrecarga de Subprogramas

Exemplo em Ada

```
1 type Mat_Type is array (Integer range <>, Integer
      range <>) of Float;
2
3 Mat_1 : Mat_Type(1..5, 1..30);
4
5 function Sumer(Mat : in Mat_Type) return Float is
6     ...
7 end Sumer;
```

Subprogramas Como Parâmetro

Subprogramas

Guilherme, Gustavo, Sean e Vinícius

Fundamentos

Ambiente de re-ferênciamento locais

Aninhamento de subprogramas

Métodos de Passagem de Parâmetros

Subprogramas Como Parâmetro

Chamar Subprogramas Indiretamente

Sobrecarga de Subprogramas

- Ideia simples, mas gera complicações.
- *Type checking*.
- referencing environment.

Referencing Environment

Subprogramas

Guilherme, Gustavo, Sean e Vinícius

Fundamentos

Ambiente de re-ferênciamento locais

Aninhamento de subprogramas

Métodos de Passagem de Parâmetros

Subprogramas Como Parâmetro

Chamar Subprogramas Indiretamente

Sobrecarga de Subprogramas

- Linguagens que permitem subprogramas aninhados.
- Shallow Binding
- Deep Binding
- Ad Hoc Binding

Exemplo

Subprogramas

Guilherme, Gustavo, Sean e Vinícius

Fundamentos

Ambiente de re-ferênciamento locais

Aninhamento de subprogramas

Métodos de Passagem de Parâmetros

Subprogramas Como Parâmetro

Chamar Subprogramas Indiretamente

Sobrecarga de Subprogramas

```
1  function sub1() {
2    var x;
3    function sub2() {
4      alert(x);
5    };
6    function sub3() {
7      var x;
8      x = 3;
9      sub4(sub2);
10   };
11   function sub4(subx) {
12     var x;
13     x = 4;
14     subx();
15   };
16   x = 1;
17   sub3();
18 };
```

Shallow Binding

Subprogramas

Guilherme, Gustavo, Sean e Vinícius

Fundamentos

Ambiente de re-ferênciamento locais

Aninhamento de subprogramas

Métodos de Passagem de Parâmetros

Subprogramas Como Parâmetro

Chamar Subprogramas Indiretamente

Sobrecarga de Subprogramas

O ambiente é o local onde o subprograma é chamado.

Shallow Binding

Subprogramas

Guilherme, Gustavo, Sean e Vinícius

Fundamentos

Ambiente de re-ferênciamento locais

Aninhamento de subprogramas

Métodos de Passagem de Parâmetros

Subprogramas Como Parâmetro

Chamar Subprogramas Indiretamente

Sobrecarga de Subprogramas

```
1  function sub1() {
2    var x;
3    function sub2() {
4      alert(x);
5    };
6    function sub3() {
7      var x;
8      x = 3;
9      sub4(sub2);
10   };
11   function sub4(subx) {
12     var x;
13     x = 4;
14     subx();
15   };
16   x = 1;
17   sub3();
18 };
```

Deep Binding

Subprogramas

Guilherme, Gustavo, Sean e Vinícius

Fundamentos

Ambiente de re-ferênciamento locais

Aninhamento de subprogramas

Métodos de Passagem de Parâmetros

Subprogramas Como Parâmetro

Chamar Subprogramas Indiretamente

Sobrecarga de Subprogramas

O ambiente refere-se onde o subprograma foi definido.

Deep Binding

Subprogramas

Guilherme, Gustavo, Sean e Vinícius

Fundamentos

Ambiente de re-ferênciamento locais

Aninhamento de subprogramas

Métodos de Passagem de Parâmetros

Subprogramas Como Parâmetro

Chamar Subprogramas Indiretamente

Sobrecarga de Subprogramas

```
1  function sub1() {
2    var x;
3    function sub2() {
4      alert(x);
5    };
6    function sub3() {
7      var x;
8      x = 3;
9      sub4(sub2);
10   };
11   function sub4(subx) {
12     var x;
13     x = 4;
14     subx();
15   };
16   x = 1;
17   sub3();
18 };
```

Ad Hoc Binding

Subprogramas

Guilherme, Gustavo, Sean e Vinícius

Fundamentos

Ambiente de re-ferênciamento locais

Aninhamento de subprogramas

Métodos de Passagem de Parâmetros

Subprogramas Como Parâmetro

Chamar Subprogramas Indiretamente

Sobrecarga de Subprogramas

O ambiente condiz com o local que o subprograma foi passado por parâmetro. Nunca implementado.

Ad Hoc Binding

Subprogramas

Guilherme, Gustavo, Sean e Vinícius

Fundamentos

Ambiente de re-ferênciamento locais

Aninhamento de subprogramas

Métodos de Passagem de Parâmetros

Subprogramas Como Parâmetro

Chamar Subprogramas Indiretamente

Sobrecarga de Subprogramas

```
1  function sub1() {
2    var x;
3    function sub2() {
4      alert(x);
5    };
6    function sub3() {
7      var x;
8      x = 3;
9      sub4(sub2);
10   };
11   function sub4(subx) {
12     var x;
13     x = 4;
14     subx();
15   };
16   x = 1;
17   sub3();
18 };
```

Chamar Subprogramas Indiretamente

Subprogramas

Guilherme, Gustavo, Sean e Vinícius

Fundamentos

Ambiente de re-ferênciamento locais

Aninhamento de subprogramas

Métodos de Passagem de Parâmetros

Subprogramas Como Parâmetro

Chamar Subprogramas Indiretamente

Sobrecarga de Subprogramas

- Subprograma conhecido em tempo de execução.
- GUI e callback.
- C/C++ ponteiro para função.
- C# Delegate.

C/C++ - Ponteiro Para Função

Subprogramas

Guilherme, Gustavo, Sean e Vinícius

Fundamentos

Ambiente de referênciamento locais

Aninhamento de subprogramas

Métodos de Passagem de Parâmetros

Subprogramas Como Parâmetro

Chamar Subprogramas Indiretamente

Sobrecarga de Subprogramas

```
1  //declaracao da funcao
2  int sum(int a, int b)
3  {
4  return a + b;
5  }
6
7  //ponteiro para a funcao
8  int (*sum_pointer)(int, int);
9  sum_pointer = &sum;
10
11 //chamar a funcao
12 (*sum_pointer)(1,2);
```

C# - Delegate

Subprogramas

Guilherme, Gustavo, Sean e Vinícius

Fundamentos

Ambiente de re-ferênciamento locais

Aninhamento de subprogramas

Métodos de Passagem de Parâmetros

Subprogramas Como Parâmetro

Chamar Subprogramas Indiretamente

Sobrecarga de Subprogramas

```
1 //declarar um delegate
2 public delegate int SumDelegate(int a, int b);
3 ...
4 //instanciar um delegate (funcao sum tem a mesma
      assinatura)
5 SumDelegate sumDelegate = new SumDelegate(sum);
6 //executar
7 sumDelegate(2,3);
```

Sobrecarga de Subprogramas

Subprogramas

Guilherme, Gustavo, Sean e Vinícius

Fundamentos

Ambiente de referênciamento locais

Aninhamento de subprogramas

Métodos de Passagem de Parâmetros

Subprogramas Como Parâmetro

Chamar Subprogramas Indiretamente

Sobrecarga de Subprogramas

- Subprogramas (diferentes) com o mesmo nome.
- Parâmetros diferentes.
- Subprogramas relacionados.
- Exemplo: Sobrecarga de construtor.
- Ada, Java, C++, C# e F#.

Suprogramas Genéricos

Subprogramas

Guilherme, Gustavo, Sean e Vinícius

Fundamentos

Ambiente de re-ferênciamento locais

Aninhamento de subprogramas

Métodos de Passagem de Parâmetros

Subprogramas Como Parâmetro

Chamar Subprogramas Indiretamente

Sobrecarga de Subprogramas

- Reuso de software é algo importante.
- Subprogramas com tipos genéricos.
- Exemplo: Ordenação independente de tipo.
- C++ - Templates
- Java e C# - Generics

C++ - Templates

Subprogramas

Guilherme, Gustavo, Sean e Vinícius

Fundamentos

Ambiente de referênciamento locais

Aninhamento de subprogramas

Métodos de Passagem de Parâmetros

Subprogramas Como Parâmetro

Chamar Subprogramas Indiretamente

Sobrecarga de Subprogramas

```
1  //declarar funcao template
2  template <class myType>
3  myType GetMax (myType a, myType b) {
4  return (a>b?a:b);
5  }
6  ...
7  //exemplo de chamada para inteiro
8  GetMax<int> (1,2);
9  ...
10 //exemplo de chamada para float
11 GetMax<float> (1,2);
```

Java - Generics

Subprogramas

Guilherme, Gustavo, Sean e Vinícius

Fundamentos

Ambiente de re-ferênciamento locais

Aninhamento de subprogramas

Métodos de Passagem de Parâmetros

Subprogramas Como Parâmetro

Chamar Subprogramas Indiretamente

Sobrecarga de Subprogramas

```
1  //declarar um metodo generico.
2  public static <T> T dolt(T[] list) {
3  ...
4  }
5  ...
6  //chamar o metodo para String
7  dolt<String>(myList);
8
9  ...
10 //chamar o metodo para Integer
11 dolt<Integer>(myList);
12
13 ...
14 //isso causaria um erro (tipo primitivo)
15 dolt<int>(myList);
```

Questões de projetos referente a funções

Considerações

- Efeitos colaterais
- Tipos de valores retornados
- Quantidade de valores retornados

Efeitos colaterais

Subprogramas

Guilherme, Gustavo, Sean e Vinícius

Fundamentos

Ambiente de re-ferênciamento locais

Aninhamento de subprogramas

Métodos de Passagem de Parâmetros

Subprogramas Como Parâmetro

Chamar Subprogramas Indiretamente

Sobrecarga de Subprogramas

Exemplo de aliasing

```
int x = 3;
...
... // se int* y = &x;
*y = 9;
```

Tipos de valores retornados

Alguns exemplos

- C permite qualquer tipo ser retornado por suas funções exceto vetores e funções.
- C++ permite tipos definidos pelo usuário ou classes serem retornados.
- Java e C#, qualquer tipo ou classe podem ser retornados por seus métodos.

Quantidade de valores retornados

Subprogramas

Guilherme, Gustavo, Sean e Vinícius

Fundamentos

Ambiente de referênciamento locais

Aninhamento de subprogramas

Métodos de Passagem de Parâmetros

Subprogramas Como Parâmetro

Chamar Subprogramas Indiretamente

Sobrecarga de Subprogramas

Linguagem Lua

Lua permite o retorno de múltiplos valores de suas funções. Por exemplo, a chamada da função:

```
a, b, c = fun()
```

Recebe três valores de retorno da função func():

```
return 3, sum, index
```

Sobrecarga de operadores

Subprogramas

Guilherme, Gustavo, Sean e Vinícius

Fundamentos

Ambiente de re-ferênciamento locais

Aninhamento de subprogramas

Métodos de Passagem de Parâmetros

Subprogramas Como Parâmetro

Chamar Subprogramas Indiretamente

Sobrecarga de Subprogramas

Linguagens como Ada, Python, Ruby e C++ suportam sobrecarga de operadores.

```
1  CVector CVector::operator+ (CVector param) {
2    CVector temp;
3    temp.x = x + param.x;
4    temp.y = y + param.y;
5    return (temp);
6  }
7
8  int main () {
9    CVector a (3,1);
10   CVector b (1,2);
11   CVector c;
12   c = a + b;
13   cout << c.x << "," << c.y;
14   return 0;
15 }
```

Closure

Subprogramas

Guilherme, Gustavo, Sean e Vinícius

Fundamentos

Ambiente de re-ferênciamento locais

Aninhamento de subprogramas

Métodos de Passagem de Parâmetros

Subprogramas Como Parâmetro

Chamar Subprogramas Indiretamente

Sobrecarga de Subprogramas

Closure é uma variável local em uma função que é mantida viva (não é desalocada) após o retorno dessa função. Linguagens como C# e JavaScript possuem closure.

```
1 function foo(x) {
2   var tmp = 3;
3   return function (y) {
4     alert(x + y + (++tmp));
5   }
6 }
7
8 var bar = foo(2);
9 bar(10);
```

Co-rotinas

Subprogramas

Guilherme, Gustavo, Sean e Vinícius

Fundamentos

Ambiente de re-ferênciamento locais

Aninhamento de subprogramas

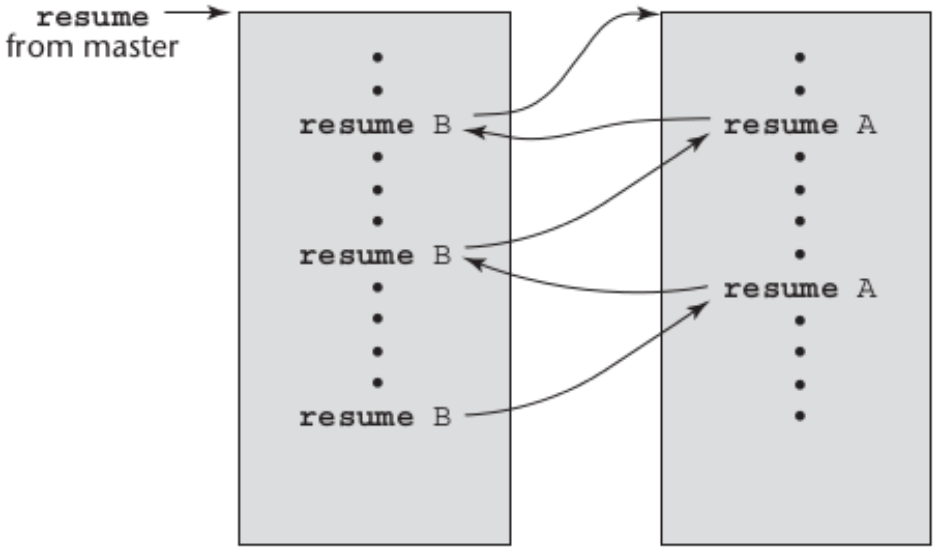
Métodos de Passagem de Parâmetros

Subprogramas Como Parâmetro

Chamar Subprogramas Indiretamente

Sobrecarga de Subprogramas

Co-rotinas são um tipo especial de subprogramas. A linguagem Lua é uma das linguagens que possui co-rotinas. Geralmente, corrotinas são criadas pela aplicação por uma unidade chamada de unidade mestre.



Co-rotinas

Subprogramas

Guilherme, Gustavo, Sean e Vinícius

Fundamentos

Ambiente de re-ferênciamento locais

Aninhamento de subprogramas

Métodos de Passagem de Parâmetros

Subprogramas Como Parâmetro

Chamar Subprogramas Indiretamente

Sobrecarga de Subprogramas