**Lista de Exercícios de Linguagens de Programação IV**

**Universidade Federal do Amazonas**

**Departamento de Ciência da Computação**

**Marco Cristo**

**Alunos: Adilson R. A. De Vilar Junior 20510539**

**Crisley Prestes Linhares 20810161**

**Procedural: Nomes, Ligações, Escopos e Procedimentos**

1. O que é um apelido (alias)? Quais as vantagens e desvantagens relacionadas com o uso de apelidos?

Um apelido é usado para chamar um bloco de código ou uma função que se usa muito.Facilita o trabalho do programador.

1. O que é escopo?

Escopo é um contexto delimitado em que os valores e expressões estão associadas. Várias linguagens de programação têm vários tipos de escopos. O tipo de escopo determina o tipo de entidades que podem conter e como isso afeta eles-ou sua [semântica](http://pt.wikipedia.org/wiki/Semântica_formal). Normalmente, o escopo é utilizado para definir o grau de ocultação da informação, isto é, a visibilidade e acessibilidade das variáveis em diferentes partes do programa. Escopos podem:

* Conter [declarações](http://pt.wikipedia.org/wiki/Declaração_(informática)) ou definições de [identificadores](http://pt.wikipedia.org/wiki/Identificador);
* Conter [instruções](http://pt.wikipedia.org/w/index.php?title=Instrução_(computação)&action=edit&redlink=1) e/ou [expressões](http://pt.wikipedia.org/wiki/Expressão_(computação)), que definem um [algoritmo](http://pt.wikipedia.org/wiki/Algoritmo) executável ou parte dele;
* aninhar ou ser aninhados.

1. O que é um bloco? Qual a relação comum entre bloco e escopo, introduzida pelo Algol e válida para a maioria das linguagens até hoje?

Geralmente, alguns [blocos](http://pt.wikipedia.org/w/index.php?title=Bloco_(computação)&action=edit&redlink=1) são definidos para criar vinculações cujo tempo de vida útil é o tempo de execução do bloco. Cada definição de bloco tem seu próprio escopo local. Uma variável declarada dentro de um bloco tem escopo de bloco e não está acessível de fora do bloco.

1. Quais as vantagens e desvantagens de ligações estáticas?

Alguns atributos de um objeto, tal como seu tipo, são limitados quando o objeto é criado e não podem serer mudados. Isto é chamado ligação estática.

1. Quais são as (mais importantes) vantagens e desvantagens de linguagens tipadas?

Normalmente, linguagens tipadas tem desempenho melhor porém dão mais trabalho para programar. Os programas porém são mais robustos, sólidos e apresentam menos erros. As linguagens menos tipadas são mais fáceis de programar porém normalmente são mais lentas pois exigem mais esforço em tempo de execução para manipular as variáveis. Além disso, é mais fácil gerar um código instável em linguagens não tipadas.

1. **PROJETO**: Escreva um programa em C com três funções. Uma declara um vetor de cem milhões de entradas na pilha, outro no *heap* e outro estaticamente. Chame cada uma das funções cem mil vezes e mostre o tempo necessário para cada caso. Como você explica este resultado?
2. O que é um procedimento (sub-programa)?

Os procedimentos são subprogramas que se distinguem das funções pelo facto de não apresentarem qualquer valor de retorno. Os procedimentos, tal como as funções, podem apresentar ou não uma lista de parâmetros que definem a forma como o procedimento irá ser executado. Tal como nas funções, é necessário declarar os parâmetros do procedimento no caso de estes existirem

1. Qual a diferença entre parâmetros formais e reais?

Parâmetros Formais são os nomes simbólicos usados na definição dos parâmetros de um subalgoritmo.

Parâmetros Reais são aqueles que substituem os parâmetros formais quando da chamada de um subalgoritmo.

1. Qual a diferença entre funções sobrecarregadas, polimórficas e genéricas?

As funções polimórficas são as que funcionam com argumentos de vários tipos.

1. Qual a vantagem de operadores sobrecarregados?

A vantagem é que um operador sobrecarregado é global.Se o modificarmos, não precisamos trabalhar com o resto separadamente o que é uma vantagem.

1. Considere o seguinte programa em sintaxe C:

void swap(int a, int b) {

int temp; temp = a; a = b; b = temp;

}

void main() {

int value = 2, list[5] = {1, 3, 5, 7, 9};

swap(value, list[0]);

swap(list[0], list[1]);

swap(value, list[value]);

}

Quais são os valores das variáveis *value* e *list* depois de cada instrução *swap*, considerando que a passagem de parâmetros é feita por (a) valor, (b) referência, (c) valor-resultado?

1. O código abaixo representa uma passagem de parâmetro por referência em C++. Que modificações você faria em foo para conseguir uma semântica de passagem de parâmetros de acordo com o modelo valor-resultado:

ftype\_t foo(ptype\_t &parameter) {

... foo statements using parameter ...

}

1. Considere o seguinte pseudo-código.

1. procedure main

2. a: integer := 1

3. b: integer := 2

4. procedure middle

5. b: integer := a

6. procedure inner

7. print a, b

8. a: integer := 3

9. –– body of middle

10. inner()

11. print a, b

12. –– body of main

13. middle()

14. print a, b

Suponha que este código foi feito em uma linguagem que possui as mesmas regras de ordem de declaração de C (mas com procedimentos aninhados, como em Pascal) – isto é, nomes devem ser declarados antes de serem usados e o escopo de um nome se estende de sua declaração até o fim do bloco. Para cada *print*, indique que declarações de a e b estão no ambiente de referência. O que o programa imprime? Ou o compilador irá detectar erros semânticos estáticos? Repita o exercício considerando as regras de ordem de declaração do C# (nomes devem ser declarados antes de usados, mas o escopo do nome é o bloco inteiro em que é declarado) e Modula-3 (nomes podem ser declarados em qualquer ordem e escopo é o bloco inteiro em que foi declarado).

1. Considere o seguinte fragmento de código C:

{ int a, b, c;

...

{

int d, e;

...

{

int f;

...

}

...

}

...

{

int g, h, i;

...

}

...

}

Assuma que cada variável inteira ocupe 4 bytes. Qual o espaço total necessário para as variáveis neste código?

1. Considere o seguinte pseudo-código:

x : integer –– global

procedure set x(n : integer)

x := n

procedure print x

write integer(x)

procedure foo(S, P : function; n : integer)

x : integer := 5

if n in {1, 3}

set x(n)

else

S(n)

if n in {1, 2}

print x

else

P

set x(0); foo(set x, print x, 1); print x

set x(0); foo(set x, print x, 2); print x

set x(0); foo(set x, print x, 3); print x

set x(0); foo(set x, print x, 4); print x

Assuma que a linguagem usa escopo dinâmico. O que o programa imprime se a linguagem usa ligação superficial? E se usa ligação profunda? Porquê?

1. Quais as diferenças entre funções matemáticas e funções em linguagem procedurais?

Do ponto de vista da programação, funções matemáticas são funções que retornam valores sem necessariamente escrever um código em determinada linguagem de programação.

Funções em linguagem procedurais, são algoritmos codificados por um programador que através deste tem um valor de retorno ou não (procedimento).