**Lista de Exercícios de Linguagens de Programação III**

**Universidade Federal do Amazonas**

**Departamento de Ciência da Computação**

**Marco Cristo**

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

**Crisley Prestes Linhares 20810161**

**Procedural: Tipos de Dados**

1. O que acontece quando um elemento não existente de um vetor é referenciado em C?

Na linguagem C, devemos ter cuidado com os limites de um vetor. Embora na sua declaração, tenhamos definido o tamanho de um vetor, o C não faz nenhum teste de verificação de acesso a um elemento dentro do vetor ou não.

1. Quais os tipos comuns de problemas com ponteiros?

Devem sempre apontar para algum endereço de memória. Portanto, todo ponteiro deve ser inicializado antes de ser utilizado.

Não guardam valores, apenas endereços de memória, ou seja, se atribuímos um valor ao ponteiro, esse valor será um novo endereço de memória.

Inicializar um ponteiro com o valor de uma dereferência.

1. Por que em várias linguagens um ponteiro é restrito a apontar para dados do mesmo tipo?

Para realizar aritmética de ponteiros em um mesmo tipo de dados.

1. Qual o uso mais comum de referências em C++? Por que o seu uso é aconselhado em lugar de ponteiros?

Variáveis de referência podem ser criadas para dar um nome diferente para as variáveis que já existem no programa, ou para passar a variável para dentro do corpo de uma função.

Em **C++** podemos criar variáveis que podem ser uma alternativa para os ponteiros em algumas situações. A vantagem de não usar diretamente o endereço (valor de ponteiro) em situações onde não precisamos lidar diretamente com valores de memória torna a programação mais segura e simplificada. Podemos deixar as operações com ponteiros apenas para quando for estritamente necessário.

1. O que é uma linguagem fortemente tipada?

São aquelas em que a declaração de tipo é obrigatória. Todas as variáveis tem um tipo específico que tem de ser explicitado no código.

1. Quais as principais vantagens e desvantagens na representação de tipos booleanos com um único bit?

Vantagens: Permite o acompanhamento das etapas lógicas de forma rápida e repetidamente.

Desvantagens: Só é possível chegar a resultados diretos, não os implícitos.

1. Considerando segurança e custo de implementação, o que é melhor: *tombstone* ou *lock-and-key*?
2. Quais os ganhos e perdas de Java ao decidir não incluir ponteiros, como em C++?

Java utiliza no luga dos ponteiros as passagens de parâmetro por valor, cuja grande vantagem é o aumento da flexibilidade do local onde os objetos são armazenados, alocados, passados e usados entre diferentes áreas de código. Desde que se possa acessar uma referência para o dado, não é necessário mover o dado em si. É mais simples também compartilhar dados entre diferentes áreas de código, desde que cada área mantenha uma referência ao dado.

Entretanto, as variáveis são alteradas se manipuladas dentro de um método, mas a passagem continua sendo por valor, ou seja, uma cópia do valor da referência é realizada.

1. Suponha que estamos compilando para uma máquina com caracteres de 1 byte, shorts de 2 bytes, inteiros de 4 bytes, reais de 8 bytes e com regras de alinhamento que requerem que cada dado primitive seja um múltiplo par do tamanho do elemento. Além disso, suponha que o compilador não reordena os campos. Quanto espaço será consumido pelo vetor abaixo. Explique.

a: array[0..9] of tuple

s: short

c: char

t: short

d: char

r: real

i: integer

**Procedural: Fluxo de Controle**

1. O que é incomum na forma como Python define blocos?

Em python os blocos são definidos pela identação, que deve seguir um padrão em todos os programas. Um bloco indica um trecho de código que está dentro de um trecho de código.

1. O que é incomum no comando de múltipla seleção do C (*switch*)?

Não há um switch-case em python, mas há outras alternativas, como as instruções if-elif-else para conseguir o mesmo efeito. Outra alternativa é usando um dicionário, em que as chaves são as opções e os valores são funções com as instruções a serem executadas para cada instrução.

1. Qual o equivalente em Python da instrução C: for (i = 0; i < n; i ++) …

for i in [1, .. , n]

1. Por que linguagens contemporâneas não incluem goto?

A maioria das linguagens de programação contemporânea não utiliza mais o goto em detrimento da programação estruturada. Com a utilização de outras instruções com a mesma utilidade do goto.

1. Quais as vantagens e desvantagens de usar palavras reservadas únicas para terminar instruções compostas? (ex: if...end if; while... end while; em ADA)

A aparência do programa, e desse modo, a sua legibilidade são fortemente influenciadas pelas formas das palavras especiais de uma linguagem. Especialmente é importante o método para formar instruções compostas ou grupos de instruções principalmente em instruções de controle. Isto gera em teoria um código mais legível a quem o lê, porém se estas palavras reservadas puderem ser utilizadas como nomes de variáveis de programas, os códigos podem ser muito confusos, tornando o mesmo código difícil de ser deduzido e mantido.

1. Descreva uma situação em que a cláusula *else* do *for* em Python seria conveniente:

for i in ['a', 'b', 'c', 'd', 'e', 'f', 'g']:

if i == 'd':

break

print i

else:

print 'd'

Usando o else no final, podemos controlar o que será executado ao final de um execução completa do for e o que será executado se o laço for quebrado por um break. Quando o break quebra o laço, a execução segue pra dentro do eles.

Neste exemplo, serão mostradas sempre as letras a, b, c e d, mas d não será impresso dentro do for, mas dentro do eles.

1. Reescreva o código abaixo em C e Pyhton, usando uma instrução de múltipla seleção:

if ((k = 1) or (k = 2)) j := 2 \* k - 1

if ((k = 3) or (k = 5)) j := 3 \* k + 1

if (k = 4) j := 4 \* k - 1

if ((k = 6) or (k = 7) or (k = 8)) j := k - 2

1. Reescreva o código abaixo em C e Pyhton, eliminando breaks sem usar gotos:

j = -3

for (i = 0; i < 10; i ++) {

switch (j + 2) {

case 3:

case 2: j--; break;

case 0: j += 2; break;

default: j = 0;

}

if (j > 0)

break;

j = 3 – i;

}

1. Considere o código escrito em C, que evita o processamento de linhas em branco:

for (;;) {

linha = leia\_linha();

if (em\_branco(linha))

break;

processe\_linha(linha);

}

Mostre como realizar a mesma tarefa usando laços *while* ou *do…while*, sem usar o *break* no meio do código. Como estas alternativas se comparam ao método apresentado, usando *for* com *break*?